This work was presented to the faculty of the Graduate School of Yale University in candidacy for the degree of Doctor of Philosophy.

Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations

Howard C. Elman

Research Report #229

April, 1982

This work was supported in part by ONR Grant N00014-76-C-0277, FSU-ONR Grant F1.N00014-80-C-0076, DOE Grant DE-AC02-77ET53053, and NSF Grant MCS-8104874.

ABSTRACT

Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations

Howard C. Elman

Yale University, 1982

In this dissertation, we consider iterative methods for solving large, sparse, nonsingular, nonsymmetric systems of linear equations. Such systems occur frequently in scientific computing, as in the discretization of non-self-adjoint elliptic partial differential equations. Until recently, few iterative methods were of practical use for solving nonsymmetric systems. We give an overview of recent developments in iterative methods for solving such systems, present new convergence results for a subset of these methods, and examine the performance of the methods in a set of numerical experiments.

Most of the techniques that we consider are similar in form to the conjugate gradient method for symmetric, positive-definite problems. They choose solution iterates from a Krylov space based on the coefficient matrix, and require no a priori estimates of scalar parameters. We show how these methods have been developed from the minimization and orthogonality properties exhibited by the conjugate gradient method, and we present new theoretical results that show that some of them are convergent for problems where the coefficient matrix has positive-definite symmetric part. We compare these methods with several alternatives, including the conjugate gradient method applied to the normal equations and the nonsymmetric Chebyshev algorithm, and we introduce a hybrid gradient/Chebyshev method.

We also consider the use of preconditioning in conjunction with these techniques. We discuss several preconditionings based on the incomplete factorization of the coefficient matrix. For two-cyclic problems, we examine the construction of a reduced linear system. For discretized non-self-adjoint elliptic problems, we consider fast direct methods as preconditionings and show that the convergence of several iterative methods with these preconditionings is independent of mesh size.

Finally, we examine the performance of various combinations of iterative methods and preconditionings in computing the numerical solution of some non-self-adjoint elliptic partial differential equations.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

In this dissertation, we consider iterative methods for solving linear systems of the form

$$A x = f , \tag{1.1}$$

where A is a large, sparse, nonsingnlar, nonsymmetric matrix of order N. Such systems arise often in scientific compnting, and their solution frequently requires a large amount of numerical compntation.

1.1 Background

Techniques for solving linear systems are classified as either direct methods or iterative methods. In the absence of roundoff error, direct methods compute the exact solution $x := A^{-1}f$ with a finite number of numerical operations. Iterative methods compute a seqnence of approximate solutions that converge to the exact solution.

Direct methods generally use some form of Ganssian elimination [41]. When the rows of the matrix A are ordered appropriately, A is factored into the product

$$A = L U ,$$

where L is a lower triangular matrix and U is an upper triangular matrix. The solution is then computed by solving successively

$$L y = f \quad \text{and} \quad U x = y .$$

These methods are most suitable for solving dense systems and densely packed sparse systems (snch as banded systems), but they have drawbacks that limit their usefulness for general sparse systems.

The main difficulties stem from the fact that the factors L and U tend to have many more nonzeros than the coefficient matrix A. Thus, more storage is required for the factors than for the original matrix, and the storage reqnirements may exceed the resources of any given computing environment. The number of arithmetic operations needed to compnte the factorization, while not as serions a limitation on the size of problems, can also become larger than is desirable. Although progress has been made in the development of orderings for the unknowns that decrease the complexity of direct methods for solving sparse problems [32, 58, 59, 66], many large sparse problems cannot be solved by direct methods on present-day computers.

Iterative methods [38, 73, 78] compute a sequence of approximate solutions $\{x_i\}$ to x by an algorithm of the form

$$x_{i+1} = F_i(x_0,\ldots,x_i) ,$$

where x_0 is an arbitrary initial guess. F_i may be linear or nonlinear; typically, it consists of a small number of matrix–vector products, scalar–vector products, inner products, and vector additions. Since matrix–vector products are relatively inexpensive for sparse problems, iterative methods tend to have low computational cost per iteration. Usually just a small number (independent of i) of vectors of length N must be stored in order to compute x_{i+1}. Thus, the storage requirements depend essentially on the number of nonzeros in A, and are lower than those of direct methods.

In order to be effective, an iterative method must converge rapidly. Many iterative methods require the estimate of scalar parameters (for example, the extreme eigenvalues of A) for fast convergence. Such methods include the successive over–relaxation (SOR), Chebyshev, and alternating direction implicit (ADI) methods (see [73, 78]), and the strongly implicit procedure (SIP) [68]. The parameters can sometimes be estimated dynamically during the early stages of the iterative procedure [38, 78], but in general the need for parameter estimates is a drawback of iterative methods. Moreover, this problem is more difficult for nonsymmetric problems than for symmetric, positive–definite ones [78].

In contrast, the conjugate gradient method is an iterative procedure for solving symmetric, positive–definite systems that requires

no estimates of scalar parameters, and in exact arithmetic converges in at most N steps. Moreover, CG is in some sense optimal over a class of iterative methods, in that the approximate solution x_i computed at each step minimizes a certain norm of the error over a translate of an i–dimensional subspace. At the same time, CG is relatively inexpensive per step. These properties make CG more robust, easier to implement, and more rapidly convergent than other iterative methods for solving symmetric, positive–definite problems [4, 12, 15, 40, 56].

The convergence of some of these methods (most notably, the Chebyshev method and CG) can be speeded by preconditioning techniques. Roughly speaking, preconditioning consists of solving a problem

$$Q^{-1}A\,x = Q^{-1}f \tag{1.2}$$

that is equivalent to (1.1), where Q is an approximation of A so that (1.2) is in some sense "easier" to solve than (1.1). Preconditionings that have been effective for symmetric, positive–definite systems include the incomplete factorization of A [50, 51], the modified incomplete factorization of A [19, 35], the SSOR preconditioning [78], and, for linear systems arising from elliptic partial differential equations, fast direct methods [6, 7, 14, 18, 70] and reduced systems [12, 57].

1.2 Iterative Methods for Nonsymmetric Problems

Although any iterative method can be formally applied to nonsymmetric problems, in most cases there is no guarantee that the sequence $\{x_i\}$ will converge to the solution. We now discuss the recent progress made in methods that are rigorously applicable to nonsymmetric problems.

Until recently, the only iterative method known to converge for general nonsymmetric problems was the conjugate gradient method applied to the normal equations [40]

$$A^T A x = A^T f .$$

The drawback of this technique is that, while the coefficient matrix is symmetric and positive-definite, the convergence rate depends on $A^T A$ rather than A. When A is symmetric and positive-definite, convergence tends to be significantly slower than when CG is applied directly to $A x = f$.

The first gradient method for nonsymmetric problems that avoided the use of the normal equations was the generalized conjugate gradient method (GCG), introduced by Concus and Golub [13] and Widlund [75]. This method is applicable to matrices of the form $A = I - R$, where R is skew-symmetric, although it can be used with preconditioning to solve more general problems.

A large collection of CG-like methods for more general problems

that share a common heuristic has been developed by Axelsson [2], Eisenstat, Elman, and Schultz [22], Jea [43], Saad [60], Vinsome [74], and Young and Jea [80, 81]. Each of these methods originates from a technique that computes x_i so that some condition (such as norm minimization or orthogonality) relative to an i-dimensional subspace is imposed. Such conditions are imposed inexpensively by CG in the symmetric, positive-definite case, but the cost for nonsymmetric problems increases with each iteration. The heuristic for cutting expenses is to relax the condition, forcing it to hold only with respect to a k-dimensional space (for some fixed k) at each step. The resulting methods have fixed computational cost per step. We have shown that some of these "truncated" methods are convergent when the symmetric part of A is positive-definite [22]. We present these results in Chapters 5 - 6.

Another generalization of CG is the biconjugate gradient method, proposed by Fletcher [28]. This method is related to the Lanczos biorthogonalization method for nonsymmetric eigenvalue problems [76] and has been examined by several authors (see [42, 53, 61, 77, 80]).

Foremost among the methods not inspired by the conjugate gradient method is the adaptive Chebyshev algorithm developed by Manteuffel [46, 48, 49]. Like the Chebyshev method for symmetric, positive-definite problems, it requires estimates of the extreme eigenvalues of the coefficient matrix. The problem of computing these estimates is more difficult for nonsymmetric matrices, in part because

the eigenvalues may be complex. Manteuffel's method includes an adaptive procedure for computing such estimates based on information acquired during the iteration.

Gay [31] has considered Broyden's method [10], a quasi-Newton method for solving nonlinear equations, as a technique for solving linear systems. He showed that this method computes the exact solution to nonsingular linear problems in at most 2N steps. Further analysis of this method has been done by Gerber and Luk [33]. In its usual form, the method builds a sequence of approximations to the inverse of A. These matrices are dense, so that the usual formulation is not suitable for sparse problems. However, Engleman, Strang, and Bathe [26] have observed that these matrices can effectively be reconstructed at each step by a sequence of inner products.

The efficiency of these iterative methods can also enhanced by preconditioning techniques. Most preconditioning techniques for symmetric, positive-definite problems, including incomplete factorizations [23, 24, 36], fast direct methods, and reduced systems, extend naturally to nonsymmetric problems. We discuss these ideas in Chapter 10.

1.3 Outline of the Dissertation

In this dissertation, we survey most of these methods for solving nonsymmetric problems, present new theoretical results on convergence properties and interrelationships among the methods, and describe their behavior in a set of numerical tests. The emphasis is on CG-like methods and preconditionings that can be used with them.

In Chapter 2, we establish definitions and conventions of notation and describe a sample problem used to test the methods presented.

In Chapter 3, we review the conjugate gradient method and a related technique, the conjugate residual method, for symmetric, positive-definite problems. We discuss error bounds for these methods and highlight their properties of minimization and orthogonality and their relationship to the symmetric Lanczos algorithm [76] as three qualities that have been generalized to CG-like methods for nonsymmetric problems.

In Chapter 4, we discuss the conjugate gradient method applied to the normal equations. We present an error bound that illustrates the limitation of this technique and give a bound for the condition number of A^TA that introduces a class of problems for which it may be suitable.

In Chapters 5 and 6, we discuss a class of CG-like methods for nonsymmetric problems that generalize the minimization properties of CG. These include Orthomin [74], and methods introduced by Axelsson [2] and Young and Jea [80, 81]. We present error bounds that show that most of

these methods are convergent for problems with positive-definite symmetric part. In Chapter 7, we discuss a set of CG-like methods for nonsymmetric problems based on the orthogonality property and the relationship with the Lanczos method. These include GCG [13, 75], Orthores [81], and Saad's projection methods [60]. We give error bounds for the "untruncated" versions of these techniques.

In Chapter 8, we describe several techniques for sparse problems that were not inspired by the conjugate gradient method. These include Manteuffel's adaptive Chebyshev method [46, 48, 49] and the version of Broyden's method developed for linear problems with sparse coefficient matrices [26]. The adaptive Chebyshev technique requires an estimate of the eigenvalues of the coefficient matrix, and it may converge slowly (or even diverge) until its adaptive procedure provides good estimates. To overcome this difficulty, we introduce several hybrid methods in which some of the CG-like methods are used to compute estimates of the eigenvalues of A prior to execution of the Chebyshev method.

In Chapter 9, we discuss the issues associated with preconditioning and present preconditioned versions of most of the iterative methods discussed. In Chapter 10, we survey some preconditioning techniques. We describe the incomplete [50, 51] and modified incomplete [19, 35, 36] factorizations of the coefficient matrix and the SSOR preconditioning [78]. For discretized elliptic problems, we consider the use of separable approximations of the coefficient matrix as

preconditionings for which fast direct methods can be used [18, 70]; we show that the convergence of several iterative methods with these preconditionings is independent of mesh size. For two-cyclic problems, we consider the use of cyclic reduction to produce an alternative linear system of smaller order (a reduced system), to which any of the previously described solution techniques can be applied.

Finally, in Chapter 11, we describe the performance of the various iterative methods and preconditionings in computing the numerical solution of some non-self-adjoint elliptic partial differential equations, and in Chapter 12, we summarize our observations and discuss issues for further research.

CHAPTER 2

Preliminaries

2.1 Introduction

In this chapter, we discuss some conventions used throughout the dissertation. In Section 2.2, we describe the mathematical notation and some conventions that we use for describing algorithms. In Section 2.3, we describe a model problem used to test the numerical methods presented.

2.2 Notation

Given a square matrix A, let $\sigma(A)$ denote the set of eigenvalues of A, and let $\lambda(A)$ denote any eigenvalue of A. The eigenvalue $\lambda(A)$ with smallest (respectively largest) absolute value is denoted by $\lambda_{min}(A)$ (respectively $\lambda_{max}(A)$). The spectral radius of A, $\rho(A)$, is defined to be $|\lambda_{max}(A)|$. If A is nonsingular, then the condition number of A is defined to be

$$K(A) := \|A\|_2 \|A^{-1}\|_2 .$$

If A is symmetric, then $K(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|}$.

The symmetric part of A is given by $M := \frac{A+A^T}{2}$, and the skew-symmetric part by $R := -\frac{A-A^T}{2}$. Thus, $A = M - R$. The Jordan canonical form for A is denoted by $J := T^{-1}AT$.

If A is symmetric and positive-definite, then the A-norm of a vector v is defined to be

$$\|v\|_A := (v,Av)^{1/2} .$$

Given a set of vectors $S = \{v_1,\ldots,v_i\}$, let $\langle v_1,\ldots,v_i\rangle$ denote the space spanned by S. If v is some vector and i is a nonnegative integer, then

$$\langle v,Av,\ldots,A^i v\rangle$$

is said to be a Krylov space based on A.

The unit vectors in $\mathbb{R}^m$ are denoted by e_i, where

$$[e_i]_j := \delta_{ij} , \quad 1 \le i,j \le m ,$$

and δ_{ij} is the Dirac delta function. We will also have occasion to allow the indices of e_i to begin at i = 0 (so that, for example, $\{e_i\}_{i=0}^m$ span $\mathbb{R}^{m+1}$).

The algorithms that we consider in this dissertation are iterative methods for solving linear systems of the form

$$A x = f ,$$

where A is a square matrix. Whenever we present an algorithm, we will describe its work per step and storage requirements. We measure the work by the number of real (i.e. floating point) multiplications and divisions, with both operations referred to as "multiplications." Most of the methods require matrix-vector products of the form Av. Since the number of multiplications in this operation depends on the number of nonzeros in A, we count it separately. We denote the cost of a matrix-vector product by mv. All of the algorithms depend explicitly on A and f, so we omit any storage required for these objects from the storage count.

For all the algorithms that we consider, the iterate x_i has the form

$$x_i = x_0 + s_i(A)(x-x_0) ,$$

where s_i is a real polynomial of degree at most i such that $s_i(0) = 0$. Equivalently, the residual $r_i := f - Ax_i$ has the form

$$r_i = q_i(A)r_0 , \qquad (2.1)$$

where q_i is a real polynomial of degree at most i and $q_i(0) = 1$. We denote the set of these i-degree polynomials $\{q_i\}$ by P_i, and we call any algorithm that satisfies (2.1) a polynomial-based method.

2.3 A Sample Problem

An important application of the methods of this dissertation is the numerical solution of elliptic partial differential equations. A prototypical problem in two dimensions is the following:

$$\begin{aligned} &-(Bu_x)_x - (Cu_y)_y + Du_x + (Du)_x + Eu_y + (Eu)_y + Fu = G \\ &\qquad \text{in } \Omega \subset \mathbb{R}^2 , \qquad (2.2) \\ &u = H \quad \text{on } \partial\Omega , \end{aligned}$$

where Ω is a rectangular domain, B(x,y), C(x,y), D(x,y), E(x,y), F(x,y), and G(x,y) are functions defined on Ω, and $B, C > 0$, $F \geq 0$ on Ω.

If (2.2) is discretized by the five-point operator on a uniform n x n grid, then the result is a system of linear equations

$$A z = f$$

of order $N = n^2$. If $D(x,y) \equiv E(x,y) \equiv 0$, then (2.2) is self-adjoint and A is a symmetric matrix. Otherwise, (2.2) is non-self-adjoint and A is nonsymmetric.

If

$$B = B(x) , \quad C = C(y) , \quad D = D(x) ,$$
$$E = E(y) , \quad F = F_1(x) + F_2(y) ,$$

then (2.2) is said to be a separable problem. We will say that the discrete operator A is separable if the corresponding continuous operator is separable.

CHAPTER 3

The Conjugate Gradient and Conjugate Residual Methods

3.1 Introduction

Consider the system of linear equations

$$A x = f , \tag{3.1}$$

where A is a symmetric, positive-definite matrix of order N. In this chapter, we review the conjugate gradient (CG) and conjugate residual (CR) methods for solving (3.1). These methods are known to be effective for solving large sparse problems [4, 12, 15, 40, 56] and have motivated efforts to develop similar methods that are applicable to nonsymmetric systems. Our purpose here is to survey their most important properties.

In Section 3.2, we present several formulations of the conjugate gradient method and outline some of its properties and error bounds. In Section 3.3, we present the conjugate residual method.

3.2 The Conjugate Gradient Method

The conjugate gradient method [40] is an iterative procedure that computes a sequence of approximate solutions $\{x_i\}$ to (3.1), starting with an arbitrary initial guess x_0. In the absence of roundoff error, the exact solution $x = A^{-1}f$ is obtained in at most N steps, so that CG can be viewed as a direct method. In practice, though, a sufficiently accurate solution is usually obtained in far fewer steps, so that CG is treated as an iterative method [56].

CG is a polynomial-based algorithm with a strong minimization property. Let $\{r_i := f - Ax_i\}$ denote the residuals of the CG iterates, and consider the i-dimensional Krylov space

$$S_i := \langle r_0, Ar_0, \ldots, A^{i-1}r_0 \rangle . \tag{3.2}$$

At each step, CG computes the point $x_i \in x_0 + S_i$ that minimizes the error functional

$$E_1(x_i) := (x-x_i, A(x-x_i))^{1/2} = \|x-x_i\|_A .$$

Because of the form of S_i, r_i can be expressed as

$$r_i = q_i(A)r_0 ,$$

where $q_i \in P_i$. CG is thus the polynomial-based algorithm that is optimal with respect to E_1. Hence, it converges at least as rapidly as other polynomial-based methods, such as Richardson's method and the Chebyshev method [73, 78]. Moreover, unlike most other iterative

methods (see [73, 78]), CG does not require the estimation of scalar parameters for fast convergence.

There are several mathematically equivalent formulations of CG (see [12, 25, 54, 56]). The most efficient one with respect to number of operations per step is the following:

Algorithm 3.1: The conjugate gradient method.

Choose x_0 .

Set $r_0 = f - Ax_0$.

Set $p_0 = r_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(r_i, r_i)}{(p_i, Ap_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i Ap_i$$

$$b_i = \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)}$$

$$p_{i+1} = r_{i+1} + b_i p_i \ .$$

The work per iteration is 5N multiplications plus one matrix-vector product. Like most iterative methods, CG has modest storage requirements. Storage is needed for four vectors of length N: x, r, p, and Ap. Also, CG does not actually require an explicit representation of A or f. The only reference to A is in the form of a matrix-vector product Av, so that a routine that performs this

operation is sufficient; the right-hand side f is used only to compute r_0.

The steplength a_i in Algorithm 3.1 minimizes $E_1(x_{i+1})$ as a function of a_i, and the directions $\{p_i\}$ are "A-orthogonal," that is

$$(p_i, Ap_j) = 0 \ , \quad i \neq j \ .$$

Because of this particular choice of directions, the one-dimensional minimization actually produces the minimum for E_1 over $x_0 + \langle p_0, \ldots, p_i \rangle$ [12, 40], from which the optimality of CG follows.

The optimality of CG is the basis for error bounds. Using the i^{th} Chebyshev polynomial [30] as a particular choice for q_i, one can derive the following bound on the error at the i^{th} step [4, 16]:

$$E_1(x_i) \leq 2 \left[\frac{1 - 1/\sqrt{K(A)}}{1 + 1/\sqrt{K(A)}} \right]^i E_1(x_0) \ . \tag{3.3}$$

Then an approximate upper bound on the number of iterations required to make the relative error $E_1(x_i)/E_1(x_0) \leq \varepsilon$ is [4]

$$i = \left[\frac{1}{2} \ln \frac{2}{\varepsilon} \right] \sqrt{K(A)} \ . \tag{3.4}$$

Moreover, CG automatically takes advantage of special distributions of the eigenvalues of A. Stronger bounds are applicable, for example, if most of the eigenvalues lie within an interval $[\alpha, \beta]$ but a small number of eigenvalues are much greater than β (see [4]).

In addition to optimality, CG satisfies the Galerkin condition

$$(Ax_i, v) = (f, v) \quad \text{for all } v \in S_i .$$

This is equivalent to the orthogonality relation for the residuals

$$(r_i, r_j) = 0 , \quad i \neq j , \tag{3.5}$$

which determines a second formulation of CG that relates it to other iterative methods. Consider the iteration

$$x_{i+1} = x_{i-1} + \omega_{i+1}(\alpha_i r_i + x_i - x_{i-1}) , \tag{3.6}$$

where $x_{-1} = 0$, and α_i and ω_{i+1} are real scalars. This is a general form for several iterative methods, including the Chebyshev semi-iterative method and the Richardson second-order method [73, 78]. For the particular choices $\alpha_i = \dfrac{(r_i, r_i)}{(r_i, Ar_i)}$, $\omega_1 = 1$, and

$$\omega_{i+1} = \left[1 - \frac{\alpha_i}{\alpha_{i-1}} \frac{\|r_i\|_2^2}{\|r_{i-1}\|_2^2} \frac{}{\omega_i}\right]^{-1} , \quad \text{for } i \geq 1 ,$$

the residuals generated by (3.6) satisfy (3.5), and the resulting algorithm is equivalent to CG [25]. This formulation, while more expensive than Algorithm 3.1 [56], is of use in the development of methods for nonsymmetric problems (see Chapter 7).

Both the optimality property and the Galerkin condition characterize CG in the class of polynomial-based methods. We state this formally as follows (see [79], Sections 5 and 7, for a proof):

Theorem 3.2: The iterates generated by the conjugate gradient method are uniquely determined by the following combinations of requirements:

$$r_i = q_i(A)r_0 , \quad q_i \in P_i , \quad \text{(polynomial-based)}$$

and either

$$x_i \text{ minimizes } E_1 , \quad \text{(optimality)}$$

or

$$(r_i, r_j) = 0 , \quad i \neq j \quad \text{(orthogonality)} .$$

A third formulation of CG shows its relation to the symmetric Lanczos algorithm (see [54]). Let v_1 be a vector such that $\|v_1\|_2 = 1$, and let $v_0 = 0$. The Lanczos algorithm, which is of use in the computation of eigenvalues of positive-definite matrices [76], constructs an orthonormal basis for the Krylov space $\langle v_1, Av_1, \ldots, A^{i-1}v_1 \rangle$ as follows:

$$\beta_{j+1}v_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1} , \quad j < i , \tag{3.7}$$

where $\alpha_j = (v_j, Av_j)$ and β_{j+1} is chosen so that $\|v_{j+1}\|_2 = 1$.

Suppose now that $v_1 = \dfrac{r_0}{\|r_0\|_2}$, where r_0 is the residual of some guess x_0 for the solution to (3.1). Then $\langle v_1, \ldots, v_i \rangle$ is the Krylov space S_i of (3.2). Hence, by Theorem 3.2, the point in $x_0 + \langle v_1, \ldots, v_i \rangle$ that minimizes E_1 is the CG iterate x_i. The coefficients of $\{v_j\}_{j=1}^{i}$ that determine x_i can be obtained by solving a symmetric tridiagonal linear system of order i with coefficient matrix

$$T_i := [\ \beta_i \quad \alpha_i \quad \beta_{i+1}\ ] \ .$$

This observation is the basis for SYMMLQ, a generalization of CG applicable to symmetric indefinite problems [54]. A similar idea has been used to develop methods for nonsymmetric problems (see Chapter 7, [60, 61]).

In summary, the conjugate gradient method is a polynomial-based iterative method for solving symmetric, positive-definite linear problems that has the three properties of optimality, orthogonality, and a connection to the Lanczos algorithm, without requiring parameter estimates.

3.3 The Conjugate Residual Method

The conjugate residual method [67] is closely related to the conjugate gradient method, differing mainly in the inner product and error functional associated with it. CR is an iterative, polynomial-based algorithm whose iterates $\{x_i\}$ minimize the error functional

$$E_2(x_i) := (A(x-x_i),A(x-x_i))^{1/2} = \|f-Ax_i\|_2$$

over the translated i-dimensional Krylov space

$$x_0 + \langle r_0, Ar_0, \ldots, A^{i-1}r_0 \rangle \ .$$

Like CG, CR computes x in at most N steps and requires no parameter estimates.

The most efficient implementation of CR with respect to number of operations per step is as follows:

Algorithm 3.3: The conjugate residual method.

Choose x_0 .

Set $r_0 = f - Ax_0$.

Set $p_0 = r_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(r_i,Ar_i)}{(Ap_i,Ap_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i Ap_i$$

$$b_i = \frac{(r_{i+1},Ar_{i+1})}{(r_i,Ar_i)}$$

$$p_{i+1} = r_{i+1} + b_i p_i$$

$$Ap_{i+1} = Ar_{i+1} + b_i Ap_i \ .$$

CR is slightly more expensive than CG. The work per loop is 6N multiplications and one matrix-vector product. It requires 5N storage for x, r, p, Ap, and Ar.

The CR steplength a_i minimizes $E_2(x_{i+1})$ as a function of a_i, and the directions satisfy

$$(Ap_i,Ap_j) = 0 \ , \qquad i \neq j \ .$$

This choice of directions forces x_{i+1} to minimize E_2 over x_0 +

$\langle p_0,\ldots,p_i\rangle$ [12]. Alternative formulations of CR based on (3.6) with a Galerkin condition [25] and the Lanczos process (3.7) [54] also exist.

Error bounds for the conjugate residual method are analogous to those for CG. For example, the analogue of (3.3) is

$$E_2(x_i) \leq 2\left[\frac{1 - 1/\sqrt{K(A)}}{1 + 1/\sqrt{K(A)}}\right]^i E_2(x_0) ,$$

so that

$$\left[\frac{1}{2}\ln\frac{2}{\varepsilon}\right]\sqrt{K(A)}$$

is an approximate upper bound on the number of steps needed to make $E_2(x_i)/E_2(x_0) \leq \varepsilon$.

CHAPTER 4

The Conjugate Gradient Method Applied to the Normal Equations

Consider the system of linear equations

$$A x = f ,$$

where A is a nonsingular, nonsymmetric matrix of order N. This problem is equivalent to the normal equations

$$A^T A x = A^T f , \tag{4.1}$$

and to the related system

$$AA^T y = f , \quad x = A^T y . \tag{4.2}$$

Since the coefficient matrices of (4.1) and (4.2) are symmetric and positive-definite, a natural way to use CG to solve nonsymmetric problems is to apply it to either of these two problems. In this chapter, we consider the advantages and disadvantages of this approach.

When CG is used to solve (4.1), the iterate x_i minimizes the residual norm $\|r_i\|_2$ over the translated Krylov space

$$x_0 + \langle A^T r_0, (A^T A) A^T r_0, \ldots, (A^T A)^{i-1} A^T r_0 \rangle .$$

We denote this method by CGNR, the conjugate gradient method applied to the normal equations with minimum residnal [40]. Although $A^T A$ figures in the development and analysis of this method, it need not be formed explicitly.

Algorithm 4.1: The conjugate gradient method applied to the uormal equations (CGNR).

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $p_0 = A^T r_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(A^T r_i, A^T r_i)}{(Ap_i, Ap_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i A p_i$$

$$b_i = \frac{(A^T r_{i+1}, A^T r_{i+1})}{(A^T r_i, A^T r_i)}$$

$$p_{i+1} = A^T r_{i+1} + b_i p_i .$$

CG can be implemented to solve (4.2) without reference to y or the approximations $\{y_i\}$ of y. This implementation was proposed by Hestenes [39] and is also known as Craig's method [27]. The iterate x_i minimizes the error norm $\|x - x_i\|_2$ over the translated Krylov space

$$x_0 + \langle r_0, (AA^T)r_0, \ldots, (AA^T)^{i-1}r_0 \rangle \ .$$

We denote this method by CGNE, the conjugate gradient method applied to the normal equations with minimum error. Again, AA^T need not be formed explicitly.

Algorithm 4.2: Craig's method (CGNE).

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $p_0 = A^T r_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(r_i, r_i)}{(p_i, p_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i A p_i$$

$$b_i = \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)}$$

$$p_{i+1} = A^T r_{i+1} + b_i p_i \ .$$

We refer to CGNR and CGNE together as CGN, the conjugate gradient method applied to the normal equations. For both versions of CGN, the work per loop is 5N multiplications plus two matrix-vector products. In both cases, 4N storage is required for the vectors x, r, p, and Ap. $A^T r$ can share storage with Ap.

Since $\sigma(A^TA) = \sigma(AA^T)$, the convergence properties of the two

algorithms are essentially the same. The only important difference is in the norm minimized. Using (3.3) and the fact that $K(A^TA) = K(AA^T) = K(A)^2$, an upper bound for the error at the i^{th} step is

$$E(x_i) \leq 2 \left[\frac{1 - 1/K(A)}{1 + 1/K(A)} \right]^i E(x_0) \ , \tag{4.3}$$

where

$$E(x_i) := \begin{cases} \|b - Ax_i\|_2 & \text{for CGNR} \ , \\ \|x - x_i\|_2 & \text{for CGNE} \ . \end{cases}$$

Hence, an approximate upper bound on the number of iterations required to make the relative error $E(x_i)/E(x_0) \leq \varepsilon$ is

$$i = \left[\frac{1}{2} \ln \frac{2}{\varepsilon} \right] K(A) \ . \tag{4.4}$$

These bounds illustrate the main drawback of CGN. The upper bound (4.4) is larger by a factor of $\sqrt{K(A)}$ than the analogous number for CG applied directly to a problem with symmetric, positive-definite coefficient matrix (see (3.4)). This suggests that if A is poorly conditioned, then the convergence of CGN could be slow. Indeed, this difficulty motivates our examination of the conjugate gradient-like methods described in subsequent chapters.

Nevertheless, CGN may be suitable for some problems. The following result bounds K(A) in terms of the extreme eigenvalues of the symmetric and skew-symmetric parts of A, and thereby suggests a class of problems for which CGN may be suitable.

Theorem 4.3: If the symmetric part M of A is positive-definite, then

$$\lambda_{min}(A^TA) \geq \lambda_{min}(M)^2 ;$$

$$\lambda_{max}(A^TA) \leq [\lambda_{max}(M) + \rho(R)]^2 .$$

Hence,

$$K(A) \leq K(M) + \frac{\rho(R)}{\lambda_{min}(M)} . \tag{4.5}$$

Proof: Let S denote the unique symmetric positive-definite square root of M, i.e., $S^2 = M$. Then

$$\begin{aligned}(A^TAx,x) &= (Ax,Ax) = (S(S - S^{-1}R)x,S(S - S^{-1}R)x) \\ &= (M(S - S^{-1}R)x,(S - S^{-1}R)x) .\end{aligned}$$

But for any real y,

$$(My,y) \geq \lambda_{min}(M)(y,y)$$

and

$$(Ry,y) = 0 ,$$

so that

$$\begin{aligned}(A^TAx,x) &\geq \lambda_{min}(M)\ ((S - S^{-1}R)x,(S - S^{-1}R)x) \\ &= \lambda_{min}(M)\ [(Sx,Sx) - 2\ (Rx,x) + (S^{-1}Rx,S^{-1}Rx)] \\ &= \lambda_{min}(M)\ [(Mx,x) + (S^{-1}Rx,S^{-1}Rx)]\end{aligned}$$

$$\geq \lambda_{min}(M)^2\ (x,x) .$$

Therefore,

$$\lambda_{min}(A^TA) = \min_{x \neq 0} \frac{(A^TAx,x)}{(x,x)} \geq \lambda_{min}(M)^2 .$$

For the upper bound on $\lambda_{max}(A^TA)$,

$$\lambda_{max}(A^TA) = \|A\|_2^2 \leq [\|M\|_2 + \|R\|_2]^2 = [\lambda_{max}(M) + \rho(R)]^2 ,$$

where we have used the fact that $\|R\|_2 = \rho(R)$ since R is skew-symmetric and hence normal [73].

Finally, inequality (4.5) follows from the fact that

$$K(A) = \sqrt{K(A^TA)} = \sqrt{\frac{\lambda_{max}(A^TA)}{\lambda_{min}(A^TA)}} .$$

Q.E.D.

Thus, if M is not ill-conditioned and $\frac{\rho(R)}{\lambda_{min}(M)}$ is small, then K(A) is small and CGN will converge rapidly. We will return to this observation in Section 10.4.

CHAPTER 5

Generalizations of the Conjugate Residual Method I

5.1 Introduction

Consider the system of linear equations

$$A x = f , \tag{5.1}$$

where A is a nonsymmetric matrix of order N with positive-definite symmetric part. In this chapter we present a class of methods for solving (5.1) that are modelled after the conjugate residual method and that exhibit minimization properties like those of CG and CR.

Recall that for symmetric positive-definite problems, the conjugate residual method computes a seqnence of approximate solutions by an iteration of the form

$$x_{i+1} = x_i + a_i p_i ,$$

and the steplength a_i minimizes $E(x_{i+1}) := \|f-A(x_i+a_i p_i)\|_2$ as a function of a_i. The direction vectors $\{p_i\}$ are computed by a two-term recurrence of the form

$$p_{i+1} = r_{i+1} + b_i p_i , \tag{5.2}$$

and they satisfy

$$(Ap_i, Ap_j) = 0 , \quad i \neq j . \tag{5.3}$$

As a result of the A^TA-orthogonality (5.3), the choice of a_i minimizes $\|r_{i+1}\|_2$ over the translated Krylov space $x_0 + \langle r_0, Ar_0, \ldots, A^i r_0 \rangle$.

Note that the bilinear form $\langle v,w \rangle := (Av,Aw)$ induces a norm even if A is not symmetric. Thus, this type of iteration is a candidate for a descent method for nonsymmetric problems. In this chapter, we consider a class of descent methods for nonsymmetric problems that combine the CR solution npdate with a modification of (5.2). The approximate solution obtained at each step minimizes the residual norm over some subspace of a Krylov space based on A.

In Section 5.2, we present the methods and give an overview of their properties. In Sections 5.3 and 5.4, we present convergence proofs and error bounds.

5.2 Four Descent Methods

We consider four methods that have the general form given in Algorithm 5.1.

The choice of a_i minimizes $\|r_{i+1}\|_2$ at each step, so that the

Algorithm 5.1: Prototype for variational methods.

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Set $p_0 = r_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(r_i, Ap_i)}{(Ap_i, Ap_i)} \tag{5.4}$$

$$x_{i+1} = x_i + a_i p_i \tag{5.5}$$

$$r_{i+1} = r_i - a_i Ap_i \tag{5.6}$$

Compute p_{i+1} . (5.7)

---

residual norms comprise a nonincreasing sequence.* We wish to compute directions $\{p_i\}$ that produce significant decreases in $\{\|r_i\|_2\}$, with as little expense as possible.

In the nonsymmetric (positive-definite) case, a set of directions that satisfy (5.3) can be computed for use in Algorithm 5.1 as follows:

$$p_{i+1} = r_{i+1} + \sum_{j=0}^{i} b_j^{(i)} p_j \ , \tag{5.8}$$

*The two expressions for a_i in Algorithms 3.3 and 5.1 are equivalent (see Theorem 5.2), but we have found (5.4) to be less sensitive to roundoff error.

where

$$b_j^{(i)} = - \frac{(Ar_{i+1}, Ap_j)}{(Ap_j, Ap_j)} \ , \qquad j \leq i \ . \tag{5.9}$$

We refer to the method given by Algorithm 5.1 and (5.8) - (5.9) as the generalized conjugate residual method (GCR). Again, $\|r_{i+1}\|_2$ is minimized over $x_0 + \langle r_0, Ar_0, \ldots, A^i r_0 \rangle$, and GCR gives the exact solution to (5.1) in at most N iterations (see Section 5.3).

The work per step and storage requirements of GCR may be prohibitively high when N is large. A modification of GCR that is significantly less expensive per step is derived by limiting the number of direction vectors used to compute p_{i+1}, allowing only k (≥ 0) directions. We consider the use of the most recently computed directions $\{p_j\}_{j=i-k+1}^{i}$, with p_{i+1} chosen to be A^TA-orthogonal to these vectors:

$$p_{i+1} = r_{i+1} + \sum_{j=i-k+1}^{i} b_j^{(i)} p_j \ , \tag{5.10}$$

$\{b_j^{(i)}\}$ defined as in (5.9).* We refer to this method, due to Vinsome [74], as Orthomin(k) (see also [80, 81]). Only k direction vectors need be saved. Both GCR and Orthomin(k) for $k \geq 1$ are

*The first k directions $\{p_j\}_{j=0}^{k-1}$ are computed by (5.8), as in GCR.

mathematically equivalent to the conjugate residual method when A is symmetric and positive-definite.

Another alternative is to restart GCR periodically: every k+1 iterations, the current iterate $x_{j(k+1)}$ is taken as the new starting guess.* At most k direction vectors have to be saved, so that the storage costs are the same as for Orthomin(k). However, the cost per iteration is lower, since in general fewer than k direction vectors are used to compute p_{i+1}. We refer to this restarted method as GCR(k).

For the special case k = 0, Orthomin(k) and GCR(k) are identical, with

$$p_{i+1} = r_{i+1} .$$

This method, which we refer to as the minimum residual method (MR), has very modest work and storage requirements, and in the symmetric case resembles the method of steepest descent (see [45]). Because of its simplicity, we consider it separately from Orthomin(k) and GCR(k).

In Table 5-1, we summarize the work and storage costs (excluding storage for A and f) of performing one loop of each of the methods. We

*Here j is a counter for the number of restarts. The j^{th} cycle of GCR(k) produces the sequence of approximate solutions $\{x_j\}_{i=(j-1)(k+1)+1}^{j(k+1)}$.

	GCR	Orthomin(k)	GCR(k)	MR
Work/ Loop	(3(i+1)+4)N + 1 mv	(3k+4)N + 1 mv	((3/2)k+4)N + 1 mv	4N + 1 mv
Storage	(2(i+2)+2)N	(2k+3)N	(2k+3)N	3N

Table 5-1: Work and storage requirements of one loop of GCR and variants.

assume that Ap is updated by

$$Ap_{i+1} = Ar_{i+1} + \sum_{j=j_i}^{i} b_j^{(i)} Ap_j ,$$

where $j_i = 0$ for GCR and $j_i = \max(0, i-k+1)$ for Orthomin(k). The storage cost includes space for the vectors x, r, Ar, $\{p_j\}_{j=0}^{i+1}$, and $\{Ap_j\}_{j=0}^{i+1}$. For GCR, Ar can share storage with Ap_{i+1}. The entries for Orthomin(k) correspond to the requirements after the k^{th} iteration. The work given for GCR(k) is the average over k+1 iterations. The cost of MR is the same as the cost of Orthomin(0) or GCR(0).*

*Several other implementations are possible. In GCR and in Orthomin(k) and GCR(k) with large k, it may be cheaper to compute Ap_{i+1} by a matrix-vector product. With a third matrix-vector product, $b_j^{(i)}$ can be computed as $-(A^T Ar_{i+1}, p_j)/(Ap_j, Ap_j)$, and the previous $\{Ap_j\}$ need not be saved.

5.3 Convergence of GCR and GCR(k)

In this section, we give convergence proofs for GCR and GCR(k). We show that GCR is optimal among polynomial-based algorithms with respect to the norm of the residual and that GCR gives the exact solution in at most N iterations, and we present error bounds for GCR and GCR(k).

We first establish a set of relations among the vectors generated by GCR. (See [40] for an analogous result for the conjugate gradient method.)

Theorem 5.2: If $\{x_i\}$, $\{r_i\}$, and $\{p_i\}$ are the iterates generated by GCR in solving the linear system (5.1), then the following relations hold:

$$(Ap_i,Ap_j) = 0 , \qquad i \neq j ; \tag{5.11}$$

$$(r_i,Ap_j) = 0 , \qquad i > j ; \tag{5.12}$$

$$(r_i,Ap_i) = (r_i,Ar_i) ; \tag{5.13}$$

$$(r_i,Ar_j) = 0 , \qquad i > j ; \tag{5.14}$$

$$(Ap_i,Ar_j) = 0 , \qquad i > j ; \tag{5.15}$$

$$(Ap_i,Ar_i) = (Ap_i,Ap_i) ; \tag{5.16}$$

$$(r_j,Ap_i) = (r_0,Ap_i) , \quad j \leq i ; \tag{5.17}$$

$$\langle p_0,\ldots,p_i\rangle = \langle r_0,Ar_0,\ldots,A^i r_0\rangle = \langle r_0,\ldots,r_i\rangle ; \tag{5.18}$$

$$\text{if } r_i \neq 0, \text{ then } p_i \neq 0 ; \tag{5.19}$$

x_{i+1} minimizes $E(w) = \|f-Aw\|_2$ over the affine space

$$x_0 + \langle p_0,\ldots,p_i\rangle . \tag{5.20}$$

Proof: The directions $\{p_i\}$ are chosen so that (5.11) holds.

Relation (5.12) is proved by induction on i. It is vacuously true for $i = 0$. Assume that it holds for $i \leq t$. Then, using (5.6) and taking the inner product with Ap_j,

$$(r_{t+1},Ap_j) = (r_t,Ap_j) - a_t(Ap_t,Ap_j) .$$

If $j < t$, then the terms on the right-hand side are zero by the induction hypothesis and (5.11). If $j = t$, then the right-hand side is zero by the definition of a_t. Hence (5.12) holds for $i = t+1$.

For (5.13), by premultiplying (5.8) by A and taking the inner product with r_i,

$$(r_i,Ap_i) = (r_i,Ar_i) + \sum_{j=0}^{i-1} b_j^{(i-1)}(r_i,Ap_j) = (r_i,Ar_i) ,$$

since all the terms in the sum are zero by (5.12).

To prove (5.14), we rewrite (5.8) as

$$r_j = p_j - \sum_{t=0}^{j-1} b_t^{(j-1)} p_t .$$

Premultiplying by A and taking the inner product with r_i ($i > j$),

$$(r_i,Ar_j) = (r_i,Ap_j) - \sum_{t=0}^{j-1} b_t^{(j-1)}(r_i,Ap_t) = 0 ,$$

by (5.12).

Similarly, for (5.15) and (5.16),

$$(Ap_i, Ar_j) = (Ap_i, Ap_j) - \sum_{t=0}^{j-1} b_t^{(j-1)} (Ap_i, Ap_t)$$

$$= \begin{cases} (Ap_i, Ap_i) , & \text{for } j = i \\ 0 , & \text{for } j < i , \end{cases}$$

by (5.11).

Relation (5.17) is proved by induction on j, for $j \leq i$. It is trivially true when $j = 0$. Assume that it holds for $j = t < i$. Using (5.6),

$$(r_{t+1}, Ap_i) = (r_t, Ap_i) - a_t (Ap_t, Ap_i) = (r_0, Ap_i) ,$$

by the induction hypothesis and (5.11).

Relation (5.18) is proved by induction on i. The three spaces are identical when $i = 0$. Assume that they are identical for $i \leq t$. Then $\{p_j\}_{j=0}^{t} \subset \langle r_0, \ldots, r_{t+1} \rangle$. But by (5.8),

$$p_{t+1} = r_{t+1} + \sum_{j=0}^{t} b_j^{(t)} p_j ,$$

so that $\langle p_0, \ldots, p_{t+1} \rangle$ is a subspace of $\langle r_0, \ldots, r_{t+1} \rangle$. By (5.11), the vectors $\{p_j\}_{j=0}^{t+1}$ are linearly independent. Hence, the dimension of $\langle r_0, \ldots, r_{t+1} \rangle$ is greater than or equal to t+1, which implies that $\{r_j\}_{j=0}^{t+1}$ are linearly independent and $\langle p_0, \ldots, p_{t+1} \rangle = \langle r_0, \ldots, r_{t+1} \rangle$. Similarly, by (5.6) and (5.8),

$$p_{t+1} = r_t - a_t Ap_t + \sum_{j=0}^{t} b_j^{(t)} p_j .$$

By the induction hypothesis, r_t, Ap_t, and $\{p_j\}_{j=0}^{t} \in \langle r_0, Ar_0, \ldots, A^{t+1} r_0 \rangle$, so that $\langle p_0, \ldots, p_{t+1} \rangle$ is a subspace of $\langle r_0, Ar_0, \ldots, A^{t+1} r_0 \rangle$. Again, the two spaces are equal because the $\{p_j\}$ are linearly independent.

The proof of (5.19) depends on the fact that the symmetric part M of A is positive-definite. If $r_i \neq 0$, then by (5.13),

$$(r_i, Ap_i) = (r_i, Ar_i) = (r_i, Mr_i) > 0 ,$$

so that $(r_i, Ap_i) \neq 0$, whence $p_i \neq 0$.

For the proof of (5.20), note that

$$x_{i+1} = x_0 + \sum_{j=0}^{i} a_j p_j .$$

Thus, $E(x_{i+1})^2$ is a quadratic functional in $\underline{a} = (a_0, \ldots, a_i)^T$. Indeed, using (5.11) to simplify the quadratic term,

$$E(x_{i+1})^2 = \| r_0 - \sum_{j=0}^{i} a_j Ap_j \|_2^2$$

$$= (r_0, r_0) - 2 \sum_{j=0}^{i} a_j (r_0, Ap_j) + \sum_{j=0}^{i} a_j^2 (Ap_j, Ap_j) .$$

Thus, E(w) is minimized over $x_0 + \langle p_0, \ldots, p_i \rangle$ when

$$a_j = \frac{(r_0, Ap_j)}{(Ap_j, Ap_j)} = \frac{(r_j, Ap_j)}{(Ap_j, Ap_j)} ,$$

by (5.17).

Q.E.D.

Corollary 5.3: GCR gives the exact solution to (5.1) in at most N iterations.

Proof: If $r_i = 0$ for some $i \leq N-1$, then $Ax_i = f$ and the assertion is proved. If $r_i \neq 0$ for all $i \leq N-1$, then $p_i \neq 0$ for all $i \leq N-1$ by (5.19). By (5.11), $\{p_i\}_{i=0}^{N-1}$ are linearly independent, so that $\langle p_0,\ldots,p_{N-1}\rangle = R^N$. Hence, by (5.20), x_N minimizes the functional E over R^N, i.e., x_N is the solution to the system.

Q.E.D.

This result does not give any insight into how close x_i is to the solution of (5.1) for $i < N$. We now derive an error bound for GCR that proves that GCR converges as an iterative method. Recall that $J = T^{-1}AT$ is the Jordan canonical form of A.

Theorem 5.4: If $\{r_i\}$ is the sequence of residuals generated by GCR, then

$$\|r_i\|_2 \leq \min_{q_i \in P_i} \|q_i(A)\|_2 \, \|r_0\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{i/2} \|r_0\|_2 \,. \tag{5.21}$$

Hence, GCR converges. If A has a complete set of eigenvectors, then

$$\|r_i\|_2 \leq K(T)\, M_i \, \|r_0\|_2 \,, \tag{5.22}$$

where

$$M_i := \min_{q_i \in P_i} \max_{\lambda \in \sigma(A)} |q_i(\lambda)| \,. \tag{5.23}$$

Moreover, if A is normal, then

$$\|r_i\|_2 \leq M_i \, \|r_0\|_2 \,. \tag{5.24}$$

Proof: By (5.18), the residuals $\{r_i\}$ generated by GCR are of the form $r_i = q_i(A)r_0$ for some $q_i \in P_i$. By (5.20),

$$\|r_i\|_2 = \min_{q_i \in P_i} \|q_i(A)r_0\|_2 \,. \tag{5.25}$$

The first inequality of (5.21) is an immediate consequence of (5.25). To prove the second inequality note that $q_1(z) := 1 + \alpha z \in P_1$, and

$$\min_{q_i \in P_i} \|q_i(A)\|_2 \leq \|q_1(A)^i\|_2 \leq \|q_1(A)\|_2^i \,.$$

But

$$\|q_1(A)\|_2^2 = \max_{x \neq 0} \frac{((I+\alpha A)x,(I+\alpha A)x)}{(x,x)}$$

$$= \max_{x \neq 0} \left[1 + 2\alpha \frac{(x,Ax)}{(x,x)} + \alpha^2 \frac{(Ax,Ax)}{(x,x)}\right] .$$

Moreover,

$$\frac{(Ax,Ax)}{(x,x)} = \frac{(x,A^TAx)}{(x,x)} \leq \lambda_{max}(A^TA) \,,$$

and, using the positive-definiteness of M,

$$\frac{(x,Ax)}{(x,x)} = \frac{(x,Mx)}{(x,x)} \geq \lambda_{min}(M) > 0 .$$

Hence, if $\alpha < 0$,

$$\|q_1(A)\|_2^2 \leq 1 + 2\lambda_{min}(M)\alpha + \lambda_{max}(A^TA)\alpha^2 .$$

This expression is minimized by $\alpha = -\frac{\lambda_{min}(M)}{\lambda_{max}(A^TA)}$, and with this choice of α,

$$\|q_1(A)\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{1/2} ,$$

which concludes the proof of (5.21).

To prove (5.22), we rewrite (5.25) as

$$\|r_i\|_2 = \min_{q_i \in P_i} \|T\, q_i(J)\, T^{-1} r_0\|_2$$

$$\leq \|T\|_2 \, \|T^{-1}\|_2 \min_{q_i \in P_i} \|q_i(J)\|_2 \, \|r_0\|_2 .$$

Since A has a complete set of eigenvectors, J is diagonal, so that

$$\min_{q_i \in P_i} \|q_i(J)\|_2 = \min_{q_i \in P_i} \max_{\lambda \in \sigma(A)} |q_i(\lambda)| ,$$

whence (5.22) follows.

If A is normal, then T can be chosen to be an orthonormal matrix, which proves (5.24).

Q.E.D.

Since the symmetric part of A is positive-definite, the spectrum of A lies in the open right half of the complex plane (see [41]). Thus, the analysis of Manteuffel [49] shows that $\min_{q_i \in P_i} \|q_i(A)\|_2$ and M_i approach zero as i goes to infinity, which also implies that GCR converges.

Theorem 5.4 can also be used to establish an error bound for GCR(k).

Corollary 5.5: If $\{r_i\}$ is the sequence of residuals generated by GCR(k), then

$$\|r_{j(k+1)}\|_2 \leq \left[\min_{q_{k+1} \in P_{k+1}} \|q_{k+1}(A)\|_2\right]^j \|r_0\|_2 , \tag{5.26}$$

so that

$$\|r_i\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{i/2} \|r_0\|_2 . \tag{5.27}$$

Hence, GCR(k) converges. Moreover, if A has a complete set of eigenvectors, then

$$\|r_{j(k+1)}\|_2 \leq [K(T)\, M_{k+1}]^j \|r_0\|_2 , \tag{5.28}$$

and if A is normal, then

$$\|r_{j(k+1)}\|_2 \leq (M_{k+1})^j \|r_0\|_2 . \tag{5.29}$$

Proof: Assertions (5.26), (5.28), and (5.29) follow from Theorem 5.4.

To prove (5.27), let $i = jk + t$ where $0 \leq t < k$. Then

$$\|r_{jk+t}\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{t/2} \|r_{jk}\|_2$$

by (5.21), and

$$\|r_{jk}\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{jk/2} \|r_0\|_2 ,$$

by (5.26) and the second inequality of (5.21).

Q.E.D.

Finally, note that the requirement that M be positive-definite is necessary. If M is indefinite, then for some $i < N$, $\{p_j\}_{j=0}^{i}$ may be linearly dependent while $\{A^j r_0\}_{j=0}^{i}$ are independent. The result is that $r_i \neq 0$, but $a_i = 0$ and $p_{i+1} = 0$, so that GCR breaks down. This occurs, for example, in the following symmetric indefinite system [81]:

$$\begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} x = \begin{vmatrix} 3 \\ 1 \end{vmatrix} . \tag{5.30}$$

With initial guess $x_0 = (1, 2)^T$, GCR breaks down on the first iteration.

5.4 Convergence of Orthomin(k)

In this section, we present convergence results for Orthomin(k) and an alternative error bound for GCR and GCR(k). We also present an analysis of Orthomin(k) in the special case when the symmetric part of A is the identity matrix.

The vectors generated by Orthomin(k) satisfy a set of relations analogous to Theorem 5.2.

Theorem 5.6: The iterates $\{x_i\}$, $\{r_i\}$, and $\{p_i\}$ generated by Orthomin(k) satisfy the relations:

$$(Ap_i, Ap_j) = 0, \quad j = i-k,\ldots,i-1 , \quad i \geq k ; \tag{5.31}$$

$$(r_i, Ap_j) = 0, \quad j = i-k-1,\ldots,i-1 , \quad i \geq k+1 ; \tag{5.32}$$

$$(r_i, Ap_i) = (r_i, Ar_i) ; \tag{5.33}$$

$$(r_i, Ar_{i-1}) = 0 ; \tag{5.34}$$

$$(Ap_i, Ar_i) = (Ap_i, Ap_i) ; \tag{5.35}$$

$$(r_j, Ap_i) = (r_{i-k}, Ap_i) , \quad j = i-k,\ldots,i , \quad i \geq k ; \tag{5.36}$$

$$\text{if } r_i \neq 0, \text{ then } p_i \neq 0 ; \tag{5.37}$$

$$\text{for } i \geq k, \ x_{i+1} \text{ minimizes } E(w) \text{ over the affine space } x_{i-k} + \langle p_{i-k},\ldots,p_i \rangle . \tag{5.38}$$

We now prove that Orthomin(k) converges. Since the analysis applies as well to GCR, GCR(k), and MR, we state the results for all four methods. We first prove two preliminary results:

Lemma 5.7: The direction vectors $\{p_i\}$ and the residuals $\{r_i\}$ generated by GCR, Orthomin(k), GCR(k), and MR satisfy

$$(Ap_i, Ap_i) \leq (Ar_i, Ar_i) \ . \tag{5.39}$$

Proof: The direction vectors are given by

$$p_i = r_i + \sum b_j^{(i-1)} p_j \ ,$$

where the limits of the sum are the same as in (5.8) for GCR and GCR(k), and (5.10) for Orthomin(k). Therefore, by the A^TA-orthogonality of the $\{p_i\}$ and the definition of $b_j^{(i-1)}$,

$$\begin{aligned}(Ap_i, Ap_i) &= (Ar_i, Ar_i) + 2 \sum b_j^{(i-1)} (Ar_i, Ap_j) + \sum (b_j^{(i-1)})^2 (Ap_j, Ap_j) \\ &= (Ar_i, Ar_i) - \sum \frac{(Ar_i, Ap_j)^2}{(Ap_j, Ap_j)} \\ &\leq (Ar_i, Ar_i) \ .\end{aligned}$$

Q.E.D.

Lemma 5.8: For any real $x \neq 0$,

$$\frac{(x,Ax)}{(Ax,Ax)} \geq \frac{\lambda_{min}(M)}{\lambda_{min}(M)\lambda_{max}(M) + \rho(R)^2} \ . \tag{5.40}$$

Proof: Letting $y = Ax$,

$$\frac{(x,Ax)}{(Ax,Ax)} = \frac{(y,A^{-1}y)}{(y,y)} = \frac{1}{2} \frac{(y,(A^{-1}+A^{-T})y)}{(y,y)} \geq \lambda_{min}\left(\frac{A^{-1}+A^{-T}}{2}\right) \ .$$

Thus, it suffices to bound $\lambda_{min}\left(\frac{A^{-1}+A^{-T}}{2}\right)$. Consider the identity

$$X^{-1}+Y^{-1} = [Y(X+Y)^{-1}X]^{-1} \ , \tag{5.41}$$

which holds for any nonsingular matrices X and Y, provided that X+Y is also nonsingular. With $X = 2A$ and $Y = 2A^T$, (5.41) leads to

$$\begin{aligned}\frac{A^{-1}+A^{-T}}{2} &= [(2A)^T(4M)^{-1}(2A)]^{-1} = [(M - R^T) M^{-1}(M - R)]^{-1} \\ &= (M + R^TM^{-1}R)^{-1} \ .\end{aligned}$$

For any $x \neq 0$,

$$(x,(M + R^TM^{-1}R)x) = (x,Mx) + (Rx,M^{-1}Rx) > 0 \ ,$$

so that $M + R^TM^{-1}R$ is positive-definite. Therefore $\frac{A^{-1}+A^{-T}}{2}$ is positive-definite and

$$\lambda_{min}\left(\frac{A^{-1}+A^{-T}}{2}\right) = \frac{1}{\lambda_{max}(M + R^TM^{-1}R)} \ .$$

But

$$\begin{aligned}\lambda_{max}(M + R^TM^{-1}R) &= \max_{x\neq 0} \left[\frac{(x,Mx)}{(x,x)} + \frac{(x,R^TM^{-1}Rx)}{(x,x)}\right] \\ &\leq \lambda_{max}(M) + \max_{x\neq 0, Rx\neq 0} \frac{(Rx,M^{-1}Rx)}{(Rx,Rx)} \frac{(Rx,Rx)}{(x,x)} \\ &\leq \lambda_{max}(M) + \lambda_{max}(M^{-1}) \ \|R^TR\|_2 \\ &= \lambda_{max}(M) + \rho(R)^2/\lambda_{min}(M) \ .\end{aligned}$$

Hence

$$\lambda_{min}\left(\frac{A^{-1}+A^{-T}}{2}\right) \geq \frac{1}{\lambda_{max}(M) + \rho(R)^2/\lambda_{min}(M)} .$$

Q.E.D.

The following result shows that Orthomin(k) converges and provides another error bound for GCR, GCR(k), and MR.

Theorem 5.9: If $\{r_i\}$ is the sequence of residuals generated by GCR, Orthomin(k), GCR(k), or MR, then

$$\|r_i\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{i/2} \|r_0\|_2 . \tag{5.42}$$

and

$$\|r_i\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{min}(M)\lambda_{max}(M) + \rho(R)^2}\right]^{i/2} \|r_0\|_2 . \tag{5.43}$$

Proof: By (5.4) and (5.6),

$$\begin{aligned}\|r_{i+1}\|_2^2 &= (r_i,r_i) - 2a_i(r_i,Ap_i) + a_i^2(Ap_i,Ap_i) \\ &= \|r_i\|_2^2 - 2\frac{(r_i,Ap_i)^2}{(Ap_i,Ap_i)} + \frac{(r_i,Ap_i)^2}{(Ap_i,Ap_i)} \\ &= \|r_i\|_2^2 - \frac{(r_i,Ap_i)^2}{(Ap_i,Ap_i)} .\end{aligned}$$

Therefore,

$$\frac{\|r_{i+1}\|_2^2}{\|r_i\|_2^2} = 1 - \frac{(r_i,Ap_i)}{(r_i,r_i)}\frac{(r_i,Ap_i)}{(Ap_i,Ap_i)} \leq 1 - \frac{(r_i,Ar_i)}{(r_i,r_i)}\frac{(r_i,Ar_i)}{(Ar_i,Ar_i)}$$

by (5.13)/(5.33) and Lemma 5.7. But

$$\frac{(r_i,Ar_i)}{(r_i,r_i)} \geq \lambda_{min}(M) ,$$

and

$$\frac{(r_i,Ar_i)}{(Ar_i,Ar_i)} = \frac{(r_i,r_i)}{(r_i,A^TAr_i)}\frac{(r_i,Ar_i)}{(r_i,r_i)} \geq \frac{\lambda_{min}(M)}{\lambda_{max}(A^TA)} ,$$

so that

$$\|r_{i+1}\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{1/2} \|r_i\|_2 ,$$

which proves (5.42). By Lemma 5.8,

$$\frac{(r_i,Ar_i)}{(Ar_i,Ar_i)} \geq \frac{\lambda_{min}(M)}{\lambda_{min}(M)\lambda_{max}(M) + \rho(R)^2} ,$$

so that

$$\|r_{i+1}\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{min}(M)\lambda_{max}(M) + \rho(R)^2}\right]^{i/2} \|r_i\|_2 ,$$

which proves (5.43).

Q.E.D.

In general, the two error bounds given in Theorem 5.9 are not comparable. They are equal when $M = I$, and (5.43) is stronger when $R = 0$. When $R = 0$, the constant $\left[\frac{\lambda_{max}(A)-\lambda_{min}(A)}{\lambda_{max}(A)}\right]^{1/2}$ in (5.43) resembles the constant $\left[\frac{\lambda_{max}(A)-\lambda_{min}(A)}{\lambda_{max}(A)+\lambda_{min}(A)}\right]^{1/2}$ in the error bound for the steepest descent method (see [45]). Thus, we believe that the bounds in Theorem 5.9 are not strict for $k \geq 1$. See Table 11-8 for a comparison of these bounds with observed maximum values of $\frac{\|r_{i+1}\|_2}{\|r_i\|_2}$.

If $M = I$, then Orthomin(1) is equivalent to GCR, and we can improve the error bounds of Theorem 5.4 and Theorem 5.9.

Theorem 5.10: If $A = I - R$ with R skew-symmetric, then Orthomin(1) is equivalent to GCR. Moreover, the residuals $\{r_i\}$ generated by Orthomin(1) satisfy

$$\|r_j\|_2 \leq \frac{2}{C(\rho(R))^j + [-C(\rho(R))]^{-j}} \|r_0\|_2 , \qquad (5.44)$$

where

$$C(\lambda) := \frac{\sqrt{1+\lambda^2} + 1}{\lambda} .$$

Proof: To prove that Orthomin(1) is equivalent to GCR, it suffices to show that $b_j^{(i)} = 0$ in (5.9) for $j \leq i-1$. But the numerator of $b_j^{(i)}$ is

$$(Ar_{i+1}, Ap_j) = (r_{i+1}, Ap_j) - (Rr_{i+1}, Ap_j) .$$

By (5.12),

$$(r_{i+1}, Ap_j) = 0 = -(r_{i+1}, Ap_j) .$$

Hence, by the skew-symmetry of R,

$$(Ar_{i+1}, Ap_j) = -(r_{i+1}, Ap_j) + (r_{i+1}, RAp_j) = -(r_{i+1}, A^2 p_j) .$$

But by (5.6),

$$(r_{i+1}, A^2 p_j) = \frac{1}{a_j}(r_{i+1}, A(r_j - r_{j+1})) = 0$$

for $j \leq i-1$, by (5.14).

Since $A = I - R$ is a normal matrix,

$$\|r_j\|_2 \leq M_j \|r_0\|_2$$

by Theorem 5.4, where M_j is as in (5.23). But

$$M_j \leq \min_{q_j \in P_j} \max_{|\mu| \leq \rho(R)} |q_j(1+i\mu)| .$$

A bound for M_j is obtained using the particular choice

$$q_j(1+i\mu) = \frac{T_j(\mu/\rho(R))}{|T_j(i/\rho(R))|} ,$$

where $T_j(z)$ is the j^{th} Chebyshev polynomial [30]. It is well known (see [75]) that

$$\max_{|\mu| \le \rho(R)} \frac{T_j(\mu/\rho(R))}{|T_j(i/\rho(R))|} = \frac{1}{|T_j(i/\rho(R))|}$$

$$\le \frac{2}{C(\rho(R))^j + [-C(\rho(R))]^{-j}},$$

where

$$C(\lambda) = \frac{\sqrt{1+\lambda^2} + 1}{\lambda}.$$

Inequality (5.44) follows.

Q.E.D.

CHAPTER 6

Generalizations of the Conjugate Residual Method II

6.1 Introduction

In this chapter we discuss two other generalizations of the conjugate residual method for solving linear systems of the form

$$A x = f , \tag{6.1}$$

where A is a nonsingular, nonsymmetric matrix of order N. Like the methods of Chapter 5, these methods minimize the residual norm over some subspace of a Krylov space based on A. Each has a variant analogous to GCR that displays N-step convergence if a complete set of vectors is retained, and each can be restarted or truncated to produce a less costly variant at the expense of finite termination, and in one case, robustness. They differ from the methods of Chapter 5 in the form of either the solution update or the direction update.

In Section 6.2, we present a generalization of CR introduced by Axelsson [2] that is applicable to systems where the coefficient matrix has positive-definite symmetric part and show that it satisfies the same error bounds as those established for Orthomin(k) in Chapter 5. In

Section 6.3, we present an alternative computation of direction vectors introduced by Young and Jea [81] for use with Algorithm 5.1 that can also be applied to indefinite problems.

6.2 Axelsson's Generalization of the Conjugate Residual Method

There are two vector updates in the conjugate residual method that can be modified to produce a method applicable to nonsymmetric problems. We derived the generalized conjugate residual method by replacing the two-term recurrence for the direction vectors in CR with (5.8) - (5.9). Axelsson [2] has proposed an alternative modification that uses a more complicated update for $\{x_i\}$ with essentially the same update for the direction vectors. This technique is shown in Algorithm 6.1.

The computation of the steplengths $\{a_j^{(i)}\}_{j=0}^{i}$ requires the solution of a least squares problem:

$$\text{minimize } \| B^{(i)} \underline{a}^{(i)} - r_i \|_2 , \tag{6.2}$$

where $B^{(i)} := [Ap_0,\ldots,Ap_i]$. The choice of $\{a_j^{(i-1)}\}_{j=0}^{i-1}$ forces

$$\|r_i\|_2 = \min_{q_i \in P_i} \|q_i(A) r_0\|_2$$

to be satisfied, so that this method is equivalent to GCR. We refer to this method as LSGCR, for the "least squares generalized conjugate residual" method. When restarted every k+1 steps, the resulting method is equivalent to GCR(k).

Algorithm 6.1: Axelsson's generalization of the conjugate residual method.

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Set $p_0 = r_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$x_{i+1} = x_i + \sum_{j=0}^{i} a_j^{(i)} p_j \ , \tag{6.3}$$

where $\{a_j^{(i)}\}_{j=0}^{i}$ are chosen to minimize $\|r_{i+1}\|_2$

$$r_{i+1} = r_i - \sum_{j=0}^{i} a_j^{(i)} Ap_j \tag{6.4}$$

$$b_i = - \frac{(Ar_{i+1}, Ap_i)}{(Ap_i, Ap_i)} \tag{6.5}$$

$$p_{i+1} = r_{i+1} + b_i p_i \tag{6.6}$$

$$Ap_{i+1} = Ar_{i+1} + b_i Ap_i \ . \tag{6.7}$$

In the truncated version of Algorithm 6.1, (6.3) is replaced by

$$x_{i+1} = x_i + \sum_{j=i-k+1}^{i} a_j^{(i)} p_j \ , \tag{6.8}$$

where $k \geq 1$ and $\{a_j^{(i)}\}_{j=i-k+1}^{i}$ are chosen to minimize $\|r_{i+1}\|_2$ [2]. This requires the solution of a least squares problem:

$$\text{minimize } \| B^{(i)} \underline{a}^{(i)} - r_i \|_2 \ , \tag{6.9}$$

for $\underline{a}^{(i)} = (a_{i-k+1}, \ldots, a_i)^T$, where $B^{(i)} := [Ap_{i-k+1}, \ldots, Ap_i]$. We denote this method by AXEL(k). For k = 1, it is equivalent to Orthomin(1).

	LSGCR	AXEL(k)
Work/ Iteration	(3(i+1)+4)N + 1 mv + $O(i^2)$	(3k+4)N + 1 mv
Storage	(2(i+2)+2)N + $O(i^2)$	(2k+3)N

Table 6-1: Work and storage requirements of one loop of Axelsson's methods.

The costs of LSGCR and AXEL(k) are given in Table 6-1. If the least squares problems (6.2) and (6.9) are solved using the normal equations

$$B^{(i)T} B^{(i)} \underline{a}^{(i)} = B^{(i)T} r_i \ , \tag{6.10}$$

then the high-order terms for this step are i+2 (respectively k+1) inner-products to update $B^{(i)T}B^{(i)}$ and compute $B^{(i)T}r_i$ for LSGCR (respectively AXEL(k)) and, for LSGCR, $O(i^2)$ multiplications to update the factorization of $B^{(i)T}B^{(i)}$ and solve for $\underline{a}^{(i)}$. The storage cost includes space for x, r, Ar, $\{p_j\}$, and $\{Ap_j\}$. For LSGCR, we also include storage for the factors of $B^{(i)T}B^{(i)}$ and note that Ap_{i+1} can overwrite Ar_{i+1}.

Theorem 6.2: The iterates $\{x_i\}$, $\{r_i\}$, and $\{p_i\}$ generated by AXEL(k) satisfy the relations:

$$(Ap_i, Ap_{i-1}) = 0 \ ; \tag{6.11}$$

$$(r_i, Ap_j) = 0 \ , \quad j = i-k, \ldots, i-1 \ , \quad i \geq k \ ; \tag{6.12}$$

$$(r_i, Ap_i) = (r_i, Ar_i) \ ; \tag{6.13}$$

$$(r_i, Ar_j) = 0 \ ; \quad j = i-k+1, \ldots, i-1 \ , \quad i \geq k \ ; \tag{6.14}$$

$$\text{if } r_i \neq 0, \text{ then } p_i \neq 0 \ ; \tag{6.15}$$

$$\text{if } r_i \neq 0, \text{ then } \{p_j\}_{j=i-k+1}^{i} \text{ are linearly independent, so that } B^{(i)} \text{ has full rank} \ ; \tag{6.16}$$

$$\text{for } i \geq k, \ x_{i+1} \text{ minimizes } E(w) = \|f - Aw\|_2 \text{ over the affine space } x_i + \langle p_{i-k+1}, \ldots, p_i \rangle \ . \tag{6.17}$$

Proof: The proof that (6.11) - (6.15) and (6.17) hold parallels the proof of Theorem 5.2. For (6.16), assume that $\{p_j\}_{j=i-k+1}^{i}$ are linearly dependent, so that $p_i \in S := \langle p_{i-k+1}, \ldots, p_{i-1} \rangle$. Then by (6.6), $r_i \in S$. But by (6.12), $(r_i, As) = 0$ for all $s \in S$, which, in conjunction with the positive-definiteness of M, implies that $r_i = 0$.

Q.E.D.

This result shows that AXEL(k) does not break down. Assertion (6.16) shows that the vector of steplengths $a^{(i)}$ is uniquely defined. The possibility that r_{i+1} and p_i may become collinear, cited as a potential difficulty in [2], is precluded by (6.12).

The following result shows that AXEL(k) converges.

Theorem 6.3: The residuals generated by AXEL(k) satisfy

$$\|r_i\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{i/2} \|r_0\|_2 \ ,$$

and

$$\|r_i\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{min}(M)\lambda_{max}(M) + \rho(R)^2}\right]^{i/2} \|r_0\|_2 \ .$$

Proof: For any i, let $\tilde{x}_{i+1} := x_i + \tilde{a}_i p_i$, where $\tilde{a}_i := \dfrac{(r_i, Ap_i)}{(Ap_i, Ap_i)}$. That is, $\tilde{x}_{i+1}$ is generated from x_i and p_i by one step of Orthomin(1). The residual is given by $\tilde{r}_{i+1} = r_i - \tilde{a}_i Ap_i$, and $\|r_{i+1}\|_2 \leq \|\tilde{r}_{i+1}\|_2$ by the choice of $\{a_j^{(i)}\}_{j=i-k+1}^{i}$. Note that

$$(Ap_i, Ap_i) \leq (Ar_i, Ar_i) \ ,$$

as in Lemma 5.7, so that

$$\frac{\|\tilde{r}_{i+1}\|_2^2}{\|r_i\|_2^2} = 1 - \frac{(r_i, Ap_i)}{(r_i, r_i)} \frac{(r_i, Ap_i)}{(Ap_i, Ap_i)} \leq 1 - \frac{(r_i, Ar_i)}{(r_i, r_i)} \frac{(r_i, Ar_i)}{(Ar_i, Ar_i)} \ ,$$

by (6.13). Therefore, as in the proof of Theorem 5.9,

$$\|\tilde{r}_{i+1}\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{1/2} \|r_i\|_2 \ ,$$

and

$$\|\tilde{r}_{i+1}\|_2 \leq \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{min}(M)\lambda_{max}(M) + \rho(R)^2}\right]^{1/2} \|r_i\|_2 .$$

Q.E.D.

6.3 Orthodir

Orthodir, proposed by Young and Jea [80, 81], is similar to GCR in that the recursion for the direction vectors in CR is replaced by a more expensive computation while the rest of CR is held intact. Unlike GCR, Orthodir is guaranteed to converge even if the symmetric part of A is not positive-definite. However, although the iteration can be truncated as in the derivation of Orthomin, the truncated algorithm is not necessarily convergent.

In Orthodir, Algorithm 5.1 is combined with the following Lanczos-like method for computing a set of A^TA-orthogonal direction vectors:

$$p_{i+1} = Ap_i + \sum_{j=0}^{i} b_j^{(i)} p_j , \qquad (6.18)$$

where

$$b_j^{(i)} = -\frac{(A^2p_i, Ap_j)}{(Ap_j, Ap_j)} , \qquad j \leq i .$$

In Orthodir(k), the truncated variant of Orthodir, (6.18) is replaced by

$$p_{i+1} = Ap_i + \sum_{j=i-k+1}^{i} b_j^{(i)} p_j . \qquad (6.19)$$

The costs of Orthodir and Orthodir(k) are given in Table 6-2. They are identical to those of GCR and Orthomin(k), respectively.

	Orthodir	Orthodir(k)
Work/ Loop	(3(i+1)+4)N + 1 mv	(3k+4)N + 1 mv
Storage	(2(i+2)+2)N	(2k+3)N

Table 6-2: Work and storage requirements of one loop of Orthodir and Orthodir(k).

If M is positive-definite, $\{\tilde{p}_i\}$ is the set of direction vectors generated by GCR, and $p_0 = \tilde{p}_0$, then $p_i = \gamma_i\tilde{p}_i$ for some scalar γ_i (see [80, 81]). Thus, Orthodir is equivalent to GCR. Moreover, $\langle p_0,\ldots,p_i\rangle = \langle r_0, Ar_0,\ldots,A^ir_0\rangle$ even when M is indefinite, so that Orthodir converges for general nonsingular A [80].

For symmetric matrices, (6.18) reduces to the three-term recurrence given by Orthodir(2). When used with Algorithm 5.1, the resnlting variational method is equivalent to the conjugate residual method [12]. If A = I - R with R skew-symmetric, then Orthodir(2) is equivalent to Orthodir. (The proof of this fact parallels the proof of the analogous result for Orthomin(1); see Theorem 5.10.) However, we know of no theoretical results that guarantee the convergence of Orthodir(k) for more general nonsymmetric problems. Indeed, we have encountered several

linear systems for which Orthomin(k) fails to converge. For example,

consider the problem

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

which has solution $x = (1, 1, 1, 1)^T$. With initial guess $x_0 = 0$,

Orthomin(2) generates the following sequence of relative residual norms:

i	$\|r_i\|_2/\|r_0\|_2$	i	$\|r_i\|_2/\|r_0\|_2$
0	1.0000	8	0.1949
1	0.7071	9	0.1949
2	0.5774	10	0.1949
3	0.5000	11	0.1949
4	0.2132	12	0.1949
5	0.2013	13	0.1949
6	0.1951	14	0.1949
7	0.1949	15	0.1949 .

The computed value of x_{15} is $(0.8358, 0.8349, 0.8405, 0.8079)^T$.

CHAPTER 7

Generalizations of the Conjugate Gradient Method: Galerkin and Lanczos Methods

7.1 Introduction

In this chapter, we consider generalizations of the conjugate gradient method for solving linear systems of the form

$$A x = f , \tag{7.1}$$

where A is a nonsymmetric matrix of order N with positive-definite symmetric part. Because the bilinear form $\langle v,w \rangle := (Av,w)$ is not an inner product, the minimization properties of CG do not lead naturally to methods for nonsymmetric problems. However, the Galerkin condition and the relationship between CG and the Lanczos method have led to generalizations of CG applicable to nonsymmetric problems.

The second formulation of the conjugate gradient method presented in Chapter 3 is derived by imposing a Galerkin condition on the basic iteration

$$x_{i+1} = x_{i-1} + \omega_{i+1}(\alpha_i r_i + x_i - x_{i-1}) . \tag{7.2}$$

We consider two generalizations of this formulation of CG for solving nonsymmetric linear systems.

The first is the generalized conjugate gradient method (GCG) of Concus and Golub [13] and Widlund [75]. In Section 7.2, we introduce GCG as a method for solving systems where the symmetric part of A is the identity matrix, and we present error bounds. In Section 7.3, we derive an equivalence between GCG and CGNE.

The second generalization of (7.2) is Orthores, proposed by Young and Jea [80, 81]. Like GCR and Orthodir, Orthores displays N-step convergence if a complete set of vectors is retained, and it can be modified to produce a less costly method with the loss of the finite termination property (and possibly robustness). We discuss Orthores in Section 7.4.

A generalization of the Lanczos method for computing the eigenvalues of nonsymmetric matrices is Arnoldi's method, which replaces the 3-term recurrence of the Lanczos method with an i-step recurrence at step i [1]. Saad [60, 61] has shown that Arnoldi's method can be used to develop methods for solving nonsymmetric linear systems, and he has also considered heuristics for truncating the iteration to cut expenses. We consider these techniques in Section 7.5.

7.2 The Generalized Conjugate Gradient Method

The generalized conjugate gradient method was the first CG-like method developed for nonsymmetric matrices that was not based on the normal equations. We present it as a method for solving systems in which the symmetric part of the coefficient matrix is the identity matrix, i.e., $A = I - R$ with R skew symmetric. We consider its use for more general systems in Chapter 9.

Concus and Golub [13] derived GCG by choosing the real scalars $\{\alpha_i\}$ and $\{\omega_{i+1}\}$ to force the residuals of the iterates generated by (7.2) to satisfy

$$(r_i, r_j) = 0 , \quad i < j . \tag{7.3}$$

This approach resembles the classical construction of orthogonal polynomials [17].

The residuals of (7.2) satisfy

$$\begin{aligned} r_{i+1} &= r_{i-1} + \omega_{i+1}(\alpha_i A r_i - r_i - r_{i-1}) \\ &= (1 - \omega_{i+1}) r_{i-1} + \omega_{i+1}(1 - \alpha_i) r_i + \omega_{i+1}\alpha_i R r_i . \end{aligned} \tag{7.4}$$

Relation (7.3) trivially holds for $i = 0$. Assume that it holds for $i \leq t$. By (7.4),

$$\begin{aligned} (r_{t+1}, r_t) &= (1 - \omega_{t+1})(r_{t-1}, r_t) + \omega_{t+1}(1 - \alpha_t)(r_t, r_t) \\ &\quad + \omega_{t+1}\alpha_t (R r_t, r_t) \\ &= \omega_{t+1}(1 - \alpha_t)(r_t, r_t) , \end{aligned}$$

by the induction hypothesis and the skew-symmetry of R. Thus, if $\alpha_t = 1$, then $(r_{t+1}, r_t) = 0$. Similarly,

$$(r_{t+1}, r_{t-1}) = (1 - \omega_{t+1})(r_{t-1}, r_{t-1}) + \omega_{t+1}(R r_t, r_{t-1}) ,$$

which is zero when

$$\omega_{t+1} = \left[1 - \frac{(R r_t, r_{t-1})}{(r_{t-1}, r_{t-1})}\right]^{-1} .$$

To complete the induction, note that for $j \leq t-2$,

$$(r_{t+1}, r_j) = \omega_{t+1}(R r_t, r_j) = -\omega_{t+1}(r_t, R r_j) ,$$

and

$$R r_j \in \langle r_{j-1}, r_j, r_{j+1} \rangle ,$$

by (7.4), so that $(r_t, R r_j) = 0$.

Finally, the compntation of ω_{i+1} can be simplified using the observation that

$$R r_{i-1} = \frac{1}{\omega_i} r_i + v$$

for some $v \in \langle r_{i-1}, r_{i-2} \rangle$, so that

$$-(R r_i, r_{i-1}) = (r_i, R r_{i-1}) = \frac{1}{\omega_i}(r_i, r_i) ,$$

by the skew-symmetry of R and (7.3). Thus, the generalized conjugate gradient method is as follows:

Algorithm 7.1: The generalized conjugate gradient method (GCG) [13, 75].

Set $x_{-1} = 0$.

Choose x_0 .

FOR i = 0 STEP 1 UNTIL Convergence DO

$$r_i = f - Ax_i$$

$$\eta_i = (r_i, r_i)$$

$$\omega_{i+1} = \begin{cases} 1 , & \text{if } i = 0 \\ [1+(\eta_i/\eta_{i-1})/\omega_i]^{-1} , & \text{if } i \geq 1 \end{cases}$$

$$x_{i+1} = x_{i-1} + \omega_{i+1}(r_i + x_i - x_{i-1}) .$$

The work per loop of GCG is 2N multiplications plus one matrix-vector product. Storage is required for x_i, x_{i-1}, and r; x_{i+1} can overwrite x_{i-1}. If the matrix-vector product Rr can be computed easily, then the residuals could be updated by

$$r_{i+1} = (1 - \omega_{i+1})r_{i-1} + \omega_{i+1}Rr_i .$$

The work per iteration would then be 3N multiplications plus one matrix-vector product by R. Since $\alpha_i = 1$, GCG is less expensive than the three-term version of CG.

Note that $\langle r_0,\ldots,r_i\rangle \subset \langle r_0, Ar_0,\ldots,A^i r_0\rangle$. Moreover, the $\{r_i\}$ are

linearly independent by (7.3), so that this inclusion is actually an equality. Hence, (7.3) is equivalent to the Galerkin condition

$$(Ax_i, v) = (f, v) \quad \text{for all } v \in \langle r_0, Ar_0,\ldots,A^{i-1}r_0\rangle .$$

The following result establishes an error bound for GCG analogous to the bound for Orthomin(1) (see Theorem 5.10).

Theorem 7.2: The error at the i^{th} step of GCG satisfies

$$\|x-x_i\|_2 \leq \frac{2}{C(\rho(R))^i + [-C(\rho(R))]^{-i}} \|x-x_0\|_2 ,$$

where

$$C(\lambda) = \frac{\sqrt{1+\lambda^2} + 1}{\lambda} .$$

This result, due to Eisenstat [21], is somewhat stronger than the original bound presented by Widlund [75] and was first proved for the even iterates by Hageman, Luk, and Young [37]. It implies that GCG will converge rapidly if $\rho(R)$ is not too large.

Finally, we remark that GCG, like CG, is related to the Lanczos method. Given a vector v_1 such that $\|v_1\|_2 = 1$, an orthonormal basis for the Krylov space $S_i := \langle v_1, Rv_1,\ldots,R^{i-1}v_1\rangle$ can be computed by a two-term recurrence of the form

$$\beta_{j+1}v_{j+1} = Rv_j + \gamma_j v_{j-1} .$$

(The proof of this fact is essentially the same as the derivation of GCG given above.) Let $v_1 := \frac{r_0}{\|r_0\|_2}$, and let x_i be the point in $x_0 + S_i$ whose residual is orthogonal to S_i. Then x_i is the i^{th} GCG iterate. This observation is due to Widlund and is the basis of an alternative derivation of GCG [75].

7.3 A Relationship between CGN and GCG

Again assuming that the symmetric part of A is the identity matrix, we now show that there is an equivalence between GCG and CGNE (Craig's method). This result was first proved in a different manner by Hageman, Luk, and Young [37].

Let $\{x_i\}$ denote the iterates generated by GCG, and let $\{\tilde{x}_i\}$ denote the iterates generated by CGNE. We show that if $x_0 = \tilde{x}_0$, then $x_{2i} = \tilde{x}_i$ for $i \geq 0$. Following Hageman et. al., we refer to this relationship as "virtual equivalence." We establish it by showing that the even iterates of GCG satisfy the orthogonality characterization of CG given by Theorem 3.2, for the coefficient matrix $AA^T = I - R^2$.

Theorem 7.3: If $A = I - R$, then the even residuals $\{r_{2i}\}$ generated by GCG satisfy

$$r_{2i} = q_i(I-R^2)r_0 \ , \tag{7.5}$$

$$(r_{2i}, r_{2j}) = 0 \ , \qquad i \neq j \ , \tag{7.6}$$

where $q_i \in P_i$. Thus if $x_0 = \tilde{x}_0$, then $x_{2i} = \tilde{x}_i$, the i^{th} iterate generated by CGNE.

Proof: We need only prove (7.5), since GCG was derived so that (7.6) holds. We show that the residuals $\{r_i\}$ generated by GCG satisfy

$$r_i = \tilde{q}_i(R)r_0 \ , \tag{7.7}$$

where $\tilde{q}_i$ is a real polynomial of degree i, $\tilde{q}_i(1) = 1$, and

$$\tilde{q}_i \text{ is } \begin{cases} \text{even, if i is even} \\ \text{odd, if i is odd} \ . \end{cases}$$

Assertion (7.5) follows with $q_i(t) = \tilde{q}_{2i}(\sqrt{1-t})$.

We prove (7.7) by induction. It is clearly true when $i = 0$. Assume that it holds for $i \leq j$. Since $\alpha_j = 1$, (7.4) reduces to

$$\begin{aligned} r_{j+1} &= (1 - \omega_{j+1})r_{j-1} + \omega_{j+1}Rr_j \\ &= [(1 - \omega_{j+1})\tilde{q}_{j-1}(R) + \omega_{j+1}R\,\tilde{q}_j(R)]\, r_0 \ . \end{aligned} \tag{7.8}$$

The conclusion is a straightforward consequence of (7.8).

Q.E.D.

The work per step of two iterations of GCG is 4N multiplications plus two matrix-vector products. This contrasts with 5N multiplications and two matrix-vector products for CGNE (see Algorithm 4.2). GCG requires 3N storage, compared with 4N for CGNE. Thus, GCG is a more efficient method than CGNE for solving problems where $A = I - R$.

7.4 Orthores

Young and Jea [80, 81] have proposed a generalization of the three-term formulation of CG (7.2) similar in spirit to GCR and Orthodir in which r_i and $\{x_j\}_{j=0}^{i}$ are used to compute x_{i+1}.

Algorithm 7.4: Orthores [80, 81].

Choose x_0 .

Compute $r_0 = f - Ax_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_j^{(i)} = \frac{(Ar_i, r_j)}{(r_j, r_j)} , \quad j=0,\ldots,i$$

$$b_i = \left(\sum_{j=0}^{i} a_j^{(i)} \right)^{-1}$$

$$c_j^{(i)} = b_i a_j^{(i)} , \quad j=0,\ldots,i$$

$$x_{i+1} = b_i r_i + \sum_{j=0}^{i} c_j^{(i)} x_j$$

$$r_{i+1} = -b_i A r_i + \sum_{j=0}^{i} c_j^{(i)} r_j .$$

The scalars $\{a_j^{(i)}\}_{j=0}^{i}$, b_i, and $\{c_j^{(i)}\}_{j=0}^{i}$ are determined so that

$$(r_i, r_j) = 0 , \quad i \neq j , \tag{7.9}$$

and

$$\sum_{j=0}^{i} c_j^{(i)} = 1 .$$

We omit the derivation, which is straightforward. The requirement that M be positive-definite is necessary: Orthores breaks down on the first step for the problem (5.30) given in Chapter 5.

As in GCG, $\langle r_0,\ldots,r_i\rangle = \langle r_0, Ar_0,\ldots,A^i r_0\rangle$ and (7.9) is equivalent to the Galerkin condition

$$(Ax_i, v) = (f, v) \quad \text{for all } v \in \langle r_0, Ar_0,\ldots,A^{i-1}r_0\rangle . \tag{7.10}$$

In the truncated version of Orthores, denoted by Orthores(k), the scalars $a_j^{(i)}$ are defined to be zero for $0 \leq j \leq i-k$. The update for the solution then has the form

$$x_{i+1} = b_i r_i + \sum_{j=i-k+1}^{i} c_j^{(i)} x_j . \tag{7.11}$$

The three-term recurrence from which Orthores is derived corresponds to Orthores(2), and when A = I - R, Orthores(2) is equivalent to GCG.

The work and storage costs of Orthores and Orthores(k) are given in Table 7-1.

	Orthores	Orthores(k)
Work/ Loop	(3(i+1)+3)N + 1 mv	(3k+3)N + 1 mv
Storage	(2(i+2)+1)N	(2k+1)N

Table 7-1: Work and storage requirements of one loop of Orthores and Orthores(k).

As a result of the orthogonality/Galerkin conditions (7.9)/(7.10), Orthores satisfies error bounds similar to those for GCR given in Theorem 5.4. The following result gives the analogue of the first part of (5.21).

Theorem 7.5: Orthores computes the solution to (7.1) in at most N steps. The iterates generated by Orthores satisfy

$$\|x-x_i\|_2 \le \frac{\|A\|_2}{\lambda_{\min}(M)} \min_{q_i \in P_i} \|q_i(A)\|_2 \, \|x-x_0\|_2 \ . \tag{7.12}$$

Proof: The first assertion follows immediately from the orthogonality relation (7.9).

For the second assertion, note that

$$\|x-x_i\|_2^2 \le \frac{1}{\lambda_{\min}(M)} (r_i, x-x_i) \ . \tag{7.13}$$

Let $S_i := \langle r_0, Ar_0, \ldots, A^{i-1}r_0 \rangle$. For any $v \in S_i$,

$$x-x_i = x-x_0 - v - (x_i-x_0 - v) = x-x_0 - v + w \ ,$$

where $w \in S_i$. Hence, using (7.10) and the Cauchy–Schwarz inequality,

$$(r_i, x-x_i) = (r_i, x-x_0 - v) \le \|r_i\|_2 \, \|x-x_0 - v\|_2 \ ,$$

so that

$$(r_i, x-x_i) \le \|A\|_2 \, \|x-x_i\|_2 \min_{v \in S_i} \|x-x_0 - v\|_2 \ . \tag{7.14}$$

But for any $v \in S_i$,

$$v = s_{i-1}(A)r_0 = As_{i-1}(A)(x-x_0)$$

for some real polynomial s_{i-1} of degree i-1, so that

$$x-x_0 - v = q_i(A)(x-x_0) \ , \tag{7.15}$$

with $q_i \in P_i$. Combining (7.13), (7.14), and (7.15),

$$\|x-x_i\|_2 \le \frac{\|A\|_2}{\lambda_{\min}(M)} \min_{q_i \in P_i} \|q_i(A)(x-x_0)\|_2 \ ,$$

whence (7.12) follows.

Q.E.D.

We know of no result that proves that Orthores(k) is convergent.

7.5 Projection Methods

Saad [60, 61] has considered a class of oblique projection methods for solving (7.1). Let K_i and L_i be two i-dimensional subspaces of $\mathbb{R}^N$. Following Saad [61], we define an oblique projection method as one that computes an approximate solution $x_i \in x_0 + K_i$ to (7.1) whose residual r_i is orthogonal to L_i. All the techniques we have considered are oblique projection methods. For example, GCR is such a method with $K_i = \langle p_0, \ldots, p_{i-1} \rangle$ and $L_i = \langle Ap_0, \ldots, Ap_{i-1} \rangle$. In this section, we focus on one technique that is a generalization of the conjugate gradient

method.*

Consider the following generalization of the Lanczos method for nonsymmetric matrices:

$$h_{j+1,j}v_{j+1} = Av_j - \sum_{t=1}^{j} h_{tj}v_t \ , \tag{7.16}$$

where $h_{tj} = (v_t, Av_j)$ for $t \leq j$ and $h_{j+1,j}$ is chosen so that $\|v_{j+1}\|_2 = 1$. This method is due to Arnoldi [1]. By the choice of $\{h_{tj}\}_{t=0}^{j}$,

$$(v_j, v_t) = \delta_{jt} \ , \tag{7.17}$$

so that $\{v_j\}_{j=1}^{i}$ is an orthonormal basis for $\langle v_1, Av_1, \ldots, A^{i-1}v_1 \rangle$. The construction (7.16) can be written in matrix form as

$$AV_i = V_i H_i + h_{i+1,i} v_{i+1} e_i^T \ , \tag{7.18}$$

where $V_i := [v_1, \ldots, v_i]$, and H_i is the upper-Hessenberg matrix whose nonzero elements are $\{h_{tj}\}$. Relations (7.17) and (7.18) imply that

$$H_i = V_i^T A V_i \ . \tag{7.19}$$

This construction is the basis for a class of oblique projection methods for solving (7.1). Given an arbitrary initial guess x_0 with

*Saad presents several other examples. One is another algorithm that is equivalent to GCR. See [22, 61].

residual r_0, let $v_1 = \frac{r_0}{\|r_0\|_2}$ and let $K_i := L_i := \langle v_1, \ldots, v_i \rangle$. Then the oblique projection method computes

$$x_i = x_0 + V_i \underline{c}^{(i)} \quad \text{such that} \quad V_i^T r_i = 0 \ . \tag{7.20}$$

These can be combined into the single equation for $\underline{c}^{(i)}$:

$$V_i^T A V_i \underline{c}^{(i)} = V_i^T r_0 \ . \tag{7.21}$$

Relation (7.17) and the definition of v_1 imply that

$$V_i^T r_0 = \|r_0\|_2 e_1 \ ,$$

which, with (7.19), implies that (7.21) can be written as

$$H_i \underline{c}^{(i)} = \|r_0\|_2 e_1 \ . \tag{7.22}$$

That is, the computation of x_i requires the solution of an upper Hessenberg system of equations of order i. We denote the method defined by this choice of x_i as the full orthogonalization method (FOM) [60].

The truncated analogue of Arnoldi's method (7.16) is given by

$$h_{j+1,j}v_{j+1} = Av_j - \sum_{t=j-k+1}^{j} h_{tj}v_t \ . \tag{7.23}$$

H_i is now a banded upper-Hessenberg matrix with bandwidth k. The matrix equation (7.18) still holds, but V_i is no longer orthonormal. Defining K_i and L_i as above, it is possible to define an oblique projection

method as in (7.20).*

As a less expensive alternative, Saad [60] suggests a relaxation of the conditions of oblique projection with respect to K_i and L_i. The approximate solution x_i is chosen to satisfy (7.22) and the first condition of (7.20), but not the second condition of (7.20). Following Saad, we refer to this method as IOM(k) for the incomplete orthogonalization method.

If A is symmetric and positive-definite, then (7.16) reduces to the three-term Lanczos recursion, so that H_i is tridiagonal and FOM is equivalent to IOM(2). In this case, x_i is equal to the approximate solution computed after i steps of the conjugate gradient method (see [54]).

By (7.18), (7.20), and (7.22), the residual r_i in both FOM and IOM(k) satisfies

$$r_i = h_{i+1,i} e_i^T \underline{c}^{(i)} v_{i+1} , \tag{7.24}$$

so that

*Although $V_i^T A V_i$ does not have a special structure in this case, H_i can still be used in the computation of $\underline{c}^{(i)}$. The actual computation requires the least squares solution of $V_i \delta^{(i)} = h_{i+1,i} v_{i+1}$ as a correction for $\|r_0\|_2 H_i^{-1} e_1$. See [60].

$$\|r_i\|_2 = h_{i+1,i} |c_i^{(i)}| . \tag{7.25}$$

Thus, an implementation of FOM and IOM(k) that uses (7.25) in the test for convergence is given by Algorithm 7.6. If $t_j = 1$ for all j in (7.27) - (7.28), then this algorithm is FOM; if $t_j = \max(1, j-k+1)$, then it is IOM(k).

Algorithm 7.6: Projection methods: FOM and IOM(k).

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $v_1 = r_0 / \|r_0\|_2$. (7.26)

FOR j = 1 STEP 1 UNTIL Convergence DO

$$h_{tj} = (Av_j, v_t) , \quad t = t_j, \ldots, j \tag{7.27}$$

$$w = Av_j - \sum_{t=t_j}^{j} h_{tj} v_t \tag{7.28}$$

$$h_{j+1,j} = \|w\|_2 \tag{7.29}$$

$$v_{j+1} = w / h_{j+1,j} \tag{7.30}$$

$$\underline{c}^{(j)} = \|r_0\|_2 H_j^{-1} e_1 \tag{7.31}$$

$$\|r_j\|_2 = h_{j+1,j} |c_j^{(j)}| . \tag{7.32}$$

Set i = j

Compute $x_i = x_0 + \sum_{j=1}^{i} c_j^{(i)} v_j$ (7.33)

For FOM, the cost of i steps of the inner loop (7.27) - (7.32) is

$$\sum_{j=1}^{i} [(2j+2)N + O(j) + 1 \text{ mv}] = i(i+3)N + O(i^2) + i \text{ mv}$$

(where O(j) is the cost of updating the factorization for H_j and computing $\underline{c}^{(j)}$). The outer steps (7.26) and (7.33) require (i+1)N multiplications. Thus, the work for one complete step of FOM is

$$(i^2 + 4i + 1)N + i \text{ mv} + O(i^2) .$$

Assuming that $i \gg k$, the cost of IOM(k) is $i(2k+2)N + O(ki) + i$ mv for i inner loops and (i+1)N for the outer steps, for a total of

$$(i(2k+3) + 1)N + i \text{ mv} .$$

For both methods, storage is required for x, Av, $\{v_j\}_{j=1}^{i+1}$, and H_j, with w overwriting Av_j. The total storage required is

$$(i+3)N + \begin{cases} O(i^2), & \text{for FOM} , \\ O(ik), & \text{for IOM(k)} . \end{cases}$$

In IOM(k), $\{v_j\}_{j=1}^{i-k}$ can be saved in secondary storage until they are needed for (7.33).

Finally, using an LU-decomposition of H_i, FOM and IOM(k) can be implemented without an inner/outer iteration [62]. That is, a new approximate solution x_{i+1} can be computed from each new vector v_i.* Let

*To make this algorithm formally similar to those in other sections, we let the indices begin at i=0 in this discussion. Thus, e_0 denotes the unit vector with zero-component equal to 1.

H_i be the Hessenberg matrix computed after i steps of (7.16) or (7.23), and let H_i have the LU-decomposition

$$H_i = L_i U_i , \tag{7.34}$$

where L_i is lower-triangular with a single nonzero sub-diagonal and U is unit upper-triangular. Using (7.22),

$$V_i \underline{c}^{(i)} = \|r_0\|_2 V_i H_i^{-1} e_0 = \|r_0\|_2 V_i U_i^{-1} L_i^{-1} e_0 .$$

Let $P_i := V_i U_i^{-1}$ and let $\underline{a} := \|r_0\|_2 L_i^{-1} e_0$. Then

$$x_i = x_0 + \sum_{j=0}^{i-1} a_j p_j ,$$

where p_j is the j^{th} column of P_i and a_j is the j^{th} entry of $\underline{a}$. But p_i and a_i can be computed directly from V_i, L_i, and U_i, as shown in Algorithm 7.7. The costs of these methods are summarized in Table 7-2.

Algorithm 7.7: Projection methods with directions: DFOM and DIOM(k).

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $v_0 = \frac{r_0}{\|r_0\|_2}$

FOR i = 0 STEP 1 UNTIL Convergence DO

$h_{j_i i} = (Av_i, v_j), \quad j=j_i,\ldots,i$

Update the factorization $H_i = L_i U_i$

$$p_i = v_i - \sum_{j=j_i}^{i-1} u_{ji} p_j$$

$$a_i = \begin{cases} 1/L_{00} , & \text{if } i = 0 \\ -L_{i,i-1} a_{i-1}/L_{ii} , & \text{if } i \geq 1 \end{cases}$$

$$x_{i+1} = x_i + a_i p_i$$

$$w = Av_i - \sum_{j=j_i}^{i} h_{ji} v_j$$

$$h_{i+1,i} = \|w\|_2$$

$$\|r_{i+1}\|_2 = h_{i+1,i} |a_i|$$

$$v_{i+1} = w/h_{i+1,i} .$$

If $j_i = 0$ for all i, then this algorithm is DFOM; if $j_i = \max(0, i-k+1)$, then it is DIOM(k).

	DFOM	DIOM(k)
Work/ Loop	(3(i+1)+2)N + 1 mv + O(i)	(3k+2)N + 1 mv + O(k)
Storage	(2(i+1)+1)N + $O(i^2)$	(2k+2)N

Table 7-2: Work and storage requirements of one loop of projection methods with directions.

It follows from (7.16), (7.17), and (7.24) that DFOM satisfies orthogonality/Galerkin conditions analogous to (7.9)/(7.10). Hence, DFOM is equivalent to Orthores, and the error bound of Theorem 7.5 holds for both FOM and DFOM.

Theorem 7.8: The iterates generated by FOM or DFOM satisfy

$$\|x-x_i\|_2 \leq \frac{\|A\|_2}{\lambda_{\min}(M)} \min_{q_i \in P_i} \|q_i(A)\|_2 \, \|x-x_0\|_2 .$$

See [60] for an alternative error analysis. We do not know whether IOM(k) and DIOM(k) are convergent for general nonsymmetric systems.

CHAPTER 8

Nonvariational Methods and Hybrid Methods

8.1 Introduction

Consider the system of linear equations

$$A x = f , \tag{8.1}$$

where A is a nonsingular, nonsymmetric matrix of order N. In this chapter, we discuss the Chebyshev algorithm and Broyden's method, two polynomial-based iterative methods for solving (8.1) not inspired by the conjugate gradient method. In addition, we consider a class of hybrid methods that combine the Chebyshev algorithm with some of the CG-like techniques of previous chapters.

The Chebyshev polynomials are used in many areas of numerical analysis [30]. Their usefulness as the basis of an iterative method for linear problems stems from the facts that they are generated by a short recursion and they have a strong optimality property. Their main drawback is that they depend on knowledge of the eigenvalues of A in order to be effective. Manteuffel [46, 48, 49] has devised an adaptive procedure that provides estimates of the eigenvalues of A to be used with the Chebyshev polynomials for solving linear systems. We discuss

this method in Section 8.2.

The initial performance of the Chebyshev algorithm may be poor if little is known about the eigenvalues of A. As we mentioned in Section 7.5, the projection methods FOM, IOM(k), DFOM, and DIOM(k) are based on Arnoldi's method for computing eigenvalues. In Section 8.3, we show that each of these methods, as well as GCR, can be used as a preprocessor for the Chebyshev algorithm to provide it with eigenvalue estimates.

A second method not inspired by CG is Broyden's method [10], a quasi-Newton method applicable to nonlinear systems of equations. Recent interest in this technique has been provoked by the observation that it computes the solution to linear problems in at most 2N iterations [31]. In Section 8.4, we discuss an implementation of Broyden's method due to Engleman, Strang, and Bathe [26] that is suitable for sparse problems.

8.2 The Chebyshev Algorithm for Nonsymmetric Systems

The Chebyshev polynomials are defined by

$$T_n(z) = \cosh(n\cosh^{-1}(z)) , \qquad z \in C .$$

This definition is equivalent to the recurrence

$$T_0(z) = 1$$
$$T_1(z) = z \qquad (8.2)$$
$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z) \ .$$

We consider an iterative method based on these polynomials for solving linear systems (8.1) in which the eigenvalues of A lie in the open right half of the complex plane. The development of this section follows Manteuffel [48, 49].

Recall that the residuals generated by polynomial-based methods satisfy

$$r_i = q_i(A)r_0 \ ,$$

where $q_i \in P_i$. Hence

$$\|r_i\|_2 \leq \|q_i(A)\|_2 \|r_0\|_2 \ ,$$

and $\|r_i\|_2$ will approach zero rapidly if $\|q_i(A)\|_2$ decreases quickly. If A has a complete set of eigenvectors, then this is the case if and only if $|q_i(\lambda)| \longrightarrow 0$ for all $\lambda \in \sigma(A)$. If A lacks a complete set of eigenvectors, then an additional requirement is that for all eigenvalues λ with invariant subspace of dimension $m > 1$, the j^{th} derivatives $q_i^{(j)}(\lambda) \longrightarrow 0$ as $i \longrightarrow \infty$, for $j < m$. The scaled and translated Chebyshev polynomials satisfy these requirements, and they are nearly the optimal choice among polynomials satisfying the first requirement [48].

We briefly discuss the optimality properties of the Chebyshev polynomials. Given any closed and bounded infinite set E in the complex plane, there exists a unique polynomial $t_i \in P_i$ that satisfies

$$\max_{z \in E} |t_i(z)| = \min_{s_i \in P_i} \max_{z \in E} |s_i(z)| \ .$$

Since $\sigma(A)$ is contained in the open right half plane, there exists an ellipse E that contains $\sigma(A)$ in its interior, and whose closure does not contain the origiu. E is described by its center d and its foci d+c and d−c, where c and d are complex numbers. Moreover, since A is real, the eigenvalues of A ocour in complex conjugate pairs. Hence, E can be chosen to be symmetric with respect to the real axis, i.e., d is real and positive and c is either real or pure imaginary. Manteuffel's nonsymmetric Chebyshev algorithm is based on the scaled and translated Chebyshev polynomials

$$q_i(z) = \frac{T_i((d-z)/c)}{T_i(d/c)} \ .$$

If c is real (i.e., the semi-major axis of E is the real axis), then for each i, q_i is the optimal polynomial for E. If c is imaginary, then these Chebyshev polynomials are not optimal in general for E, but they approach the optimal polyuomials asymptotically as $i \longrightarrow \infty$, and the convergence to the optimal polynomial is rapid [49].

Given c and d, and using the recursion (8.2), the Chebyshev algorithm can be implemented by a set of recursive formulas:

Algorithm 8.1: The Chebyshev algorithm with fixed parameters.

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $\Delta_0 = \frac{1}{d} r_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$x_{i+1} = x_i + \Delta_i$$

$$r_{i+1} = f - Ax_{i+1}$$

$$\alpha_{i+1} = \begin{cases} 2d/(2d^2-c^2) , & \text{if } i = 0 \\ d - (c/2)^2 \alpha_i , & \text{if } i \geq 1 \end{cases}$$

$$\beta_{i+1} = d\alpha_{i+1} - 1$$

$$\Delta_{i+1} = \alpha_{i+1} r_{i+1} + \beta_{i+1} \Delta_i .$$

The work per loop of this method is 2N multiplications plus one matrix-vector product. 3N storage is required for x, r, and Δ, with Ax overwritten by r.

This algorithm requires values for c and d, and will be most effective if the ellipse containing the eigenvalues is as small as possible. Manteuffel has devised an adaptive procedure for computing good values of c and d by estimating the convex hull of the eigenvalues of A. This technique is based on the following two facts.

1. Let the scaled and translated Chebyshev polynomials $\{q_i\}$ be determined by some particular values of c and d. Then there exists a linear operator S(A) such that $q_i(A) \cong S(A)^i$, so that the residuals produced by the Chebyshev algorithm satisfy $r_i \cong S(A)^i r_0$.

2. Given a set H in the right half plane, the parameters c and d that define the smallest ellipse containing H are given by the solution to the minimax problem

$$\min_{c,d} \max_{z \in H} r(z) , \tag{8.3}$$

where

$$r(z) = \left| \frac{(d-z) + \sqrt{(d-z)^2 - c^2}}{d + \sqrt{d^2 - c^2}} \right| .$$

By the first fact, if the Chebyshev algorithm is performed with some given values of c and d, then the residuals generated represent the vectors produced by the power method for S(A), from which estimates of the extreme eigenvalues of A can be obtained. Letting H denote the convex hull of these estimates, the minimax problem (8.3) can then be solved to obtain improved values for c and d. Thus, given initial values for c and d, and taking {d+c,d-c} as the initial guess for the convex hull of the eigenvalues of A, the adaptive Chebyshev algorithm is given by Algorithm 8.2. The details of steps 2 and 3 of this procedure are given in [46; 48, 49]. Heuristics for keeping the cost of step 2 down are given in [46; 48, 64].

Algorithm 8.2: The adaptive Chebyshev algorithm.

REPEAT UNTIL Convergence

1. Perform several (say, ten to twenty) steps of Algorithm 8.1;
2. Use the computed residuals and S(A) to obtain estimates for the extreme eigenvalues of A, and update the estimate of the convex hull of $\sigma(A)$;
3. Solve problem (8.3) to compute new values for c and d.

We remark that the eigenvalue estimates obtained in step 2 lie in the field of values of A $\{\lambda \varepsilon C \mid \lambda=\frac{(z,Az)}{(z,z)}, z\varepsilon C^N\}$, but not necessarily in the convex hull of $\sigma(A)$. This does not present a serious problem if the symmetric part of A is positive-definite, but it may lead to eigenvalue estimates with negative real part if the symmetric part is indefinite. In this case, the enclosing ellipse contaius the origin and the Chebyshev polynomials are not effective. Thus, the requirement that $\sigma(A)$ lie in the open right half plane is not strong enough to guarantee the convergence of the adaptive Chebyshev algorithm, but the stronger requirement that the symmetric part of A be positive-definite is sufficient [46].

8.3 Hybrid Methods

The Chebyshev algorithm is sensitive to the parameters c and d. The associated ellipse should contain $\sigma(A)$ but be as small as possible. If it is too large or too small (i.e., does not contain $\sigma(A)$), then Algorithm 8.1 may converge slowly or diverge. The residuals generated in such a situation can be nsed by Algorithm 8.2 to compute improved parameters, but the initial set of iterations may not improve the solution to (8.1). Thus, if little is known about the spectrum of A, then the adaptive Chebyshev algorithm has a potentially high start-up cost.

In this section, we discuss a way to avoid this difficulty. Some of the conjugate gradient-like methods that we have discussed can be used to estimate eigenvalues of A, and these values can be used to compute initial values for c and d. Thus, we propose a hybrid method in which a gradient method that computes estimates for the extreme eigenvalues of A is followed by the Chebyshev algorithm. Since the gradient methods are more expensive per step than the Chebyshev algorithm, they can be viewed as preprocessing techniques, performing only enough steps to generate reasonable values for c and d. Unlike the adaptive Chebyshev algorithm alone, the hybrid method produces these parameters simultaneously with improved solution estimates. (See [52] for a hybrid CG/SOR method for symmetric, positive-definite problems.)

The CG-like methods that we consider are GCR, FOM, and IOM(k). We

first discuss the properties shared by these methods that make them useful for estimating eigenvalues. At little or no extra cost, all three methods generate upper-Hessenberg matrices that are associated with nearly invariant subspaces of A. That is, on the i^{th} iteration, there is an upper-Hessenberg matrix H_i of order i and vectors $\{v_1,\ldots,v_{i+1}\}$ such that

$$AV_i = V_iH_i + v_{i+1}e_i^T , \tag{8.4}$$

where $V_i := [v_1,\ldots,v_i]$. For small i, the eigenvalues of H_i can be computed easily [76] and used as estimates for the eigenvalues of A.

If the trailing term $v_{i+1}e_i^T$ did not appear in (8.4), then the eigenvalues of H_i would be eigenvalues of A. This is a heuristic explanation for this choice of eigenvalue estimates. A more rigorous justification applicable in the cases of GCR and FOM is as follows [63]. The vectors $\{v_i\}$ associated with both these methods are orthonormal. Let π_i denote the projection operator from C^N into $\langle v_1,\ldots,v_i\rangle$, and consider the projected operator

$$A_i := \pi_iA\pi_i ,$$

which induces a linear operator from $\langle v_1,\ldots,v_i\rangle$ into itself. Given an eigenpair (λ,w) of H_i, let $v = V_iw$. Then

$$\begin{aligned} A_iv &= \pi_iA\pi_iV_iw = \pi_iAV_iw = \pi_i(V_iH_i + v_{i+1}e_i^T)w \\ &= \lambda\pi_iV_iw + w_i\pi_iv_{i+1} = \lambda v , \end{aligned}$$

since v_{i+1} is orthogonal to $\langle v_1,\ldots,v_i\rangle$. Thus, the eigenvalues of H_i are the eigenvalues of the projected operator A_i. See [63] for an analysis of the quantitative relationship between the eigenvalnes of A and A_i.

We have described the construction of H_i for FOM and IOM(k) in Chapter 6; we now discuss the analogous construction for GCR. Using (5.6), (5.8), and (5.18), the directions and residuals generated by GCR satisfy

$$Ap_{i+1} = Ar_{i+1} - \sum_{j=0}^{i} b_j^{(i)}Ap_j = - a_iA^2p_i + w , \tag{8.5}$$

where $w \in \langle Ap_0,\ldots,Ap_i\rangle$. With $v_i := \frac{Ap_i}{\|Ap_i\|_2}$, (8.5) is eqnivalent to the matrix equation (8.4), and by (5.11), V_i is orthonormal, so that $H_i = V_i^TAV_i$, or

$$h_{jk} = \frac{(Ap_j,A(Ap_k))}{\|Ap_j\|_2\ \|Ap_k\|_2} .$$

But

$$A^2p_k = \frac{1}{a_k}(Ar_k - Ar_{k+1}) ,$$

so that

$$\begin{aligned} h_{jk} &= \frac{1}{a_k}\frac{(Ap_j,Ar_k) - (Ap_j,Ar_{k+1})}{\|Ap_j\|_2\|Ap_k\|_2} \\ &= \frac{\|Ap_k\|_2}{\|Ap_j\|_2}\left[\frac{(Ap_j,Ar_k) - (Ap_j,Ar_{k+1})}{(r_k,Ap_k)}\right] . \end{aligned} \tag{8.6}$$

All the quantities in (8.6) are computed during the execution of GCR. The quantities (r_k, Ap_k) and $\{\|Ap_j\|_2\}$ are obtained during the computation of $\{a_j\}$. For $j < k$, the inner products in the numerator are the numerators of $b_j^{(k-1)}$ and $b_j^{(k)}$. For $j = k$, (Ap_j, Ar_{k+1}) is the numerator of $b_j^{(k)}$ and $(Ap_k, Ar_k) = (Ap_k, Ap_k)$ by (5.16). For $j = k+1$, $(Ap_{k+1}, Ar_k) = 0$ by (5.15), and $(Ap_{k+1}, Ar_{k+1}) = (Ap_{k+1}, Ap_{k+1})$ by (5.16).

As the number of steps increases, GCR and FOM become considerably more expensive than the Chebyshev algorithm, so that they are cost-effective only if just a few iterations are performed before switching to the Chebyshev algorithm. An alternative is to base (8.4) on IOM(k).* The work per step and storage costs are lower, and the upper-Hessenberg matrix H_i is sparse, so that more steps can be taken with relatively modest expense. Unfortunately, the matrix V_i generated by IOM(k) is not orthogonal, and we know of no result that proves that the eigenvalues of H_i converge to those of A.

*Orthomin(k) does not lead to a relation of the form (8.4) with an upper-Hessenberg H_i.

8.4 Broyden's Method

Broyden's method [10] is an iterative method for solving nonlinear systems of equations of the form

$$F(x) = 0 , \tag{8.7}$$

where $F: R^N \to R^N$ is continuously differentiable. It is an example of a quasi-Newton method: the approximate solutions are computed by an iteration of the form

$$x_{i+1} = x_i - Q_i^{-1} F(x_i) ,$$

where Q_i^{-1} is an approximation of the inverse of the Jacobian matrix $F'(x_i)$. The philosophy behind such methods is to achieve nearly as rapid convergence as Newton's method (where $Q_i = F'(x_i)$) without the expense of computing and inverting the Jacobian (see [10]).

Typically in quasi-Newton methods, the $\{Q_i^{-1}\}$ are built up by an update of the form

$$Q_{i+1}^{-1} = Q_i^{-1} + U_i ,$$

where U_i is of low rank and the secant condition

$$Q_{i+1}^{-1}(F(x_{i+1}) - F(x_i)) = x_{i+1} - x_i$$

holds. The initial approximation Q_0^{-1} is arbitrary. In Broyden's method, Q_{i+1}^{-1} is obtained from Q_i^{-1} by a rank-one update.

Algorithm 8.3: Broyden's method for nonlinear systems.

Choose x_0 .

Choose Q_0^{-1} .

Compute $p_0 = -Q_0^{-1}F(x_0)$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$x_{i+1} = x_i + p_i$$

$$y_i = F(x_{i+1}) - F(x_i)$$

$$v_i = \frac{Q_i^{-T}p_i}{p_i^T Q_i^{-1} y_i}$$

$$Q_{i+1}^{-1} = Q_i^{-1} + (p_i - Q_i^{-1}y_i)v_i^T$$

$$p_{i+1} = -Q_{i+1}^{-1}F(x_{i+1}) .$$

If F is linear, then (8.7) is equivalent to the linear system of equations (8.1). Gay [31] has shown that if A is nonsingular, then Broyden's method computes the solution to (8.1) in at most 2N steps (see also [33]). The matrix update is not suitable for large sparse problems, however, because Q_{i+1}^{-1} is typically a dense matrix. We now present an alternative implementation for linear problems due to Engleman, Strang, and Bathe [26] in which the explicit formation of Q_{i+1}^{-1} is replaced by a sequence of inner products.

For linear problems, the notation for Broyden's method can be made to conform to that of most of the other methods we have considered, as follows:

$$r_i = -F(x_i) = f - Ax_i ,$$

$$p_{i+1} = Q_{i+1}^{-1}r_{i+1} ,$$

$$Ap_i = y_i = A(x_{i+1} - x_i) .$$

The update for Q_{i+1}^{-1} can then be written as

$$Q_{i+1}^{-1} = \left[I + \frac{(p_i - Q_i^{-1}Ap_i)p_i^T}{p_i^T Q_i^{-1} Ap_i}\right] Q_i^{-1} . \qquad (8.8)$$

For any i, let

$$v_j^{(i)} := Q_j^{-1}r_i , \quad 0 \le j \le i .$$

Starting with $v_0^{(i+1)} = Q_0^{-1}r_{i+1}$, the $\{v_j^{(i+1)}\}_{j=1}^{i+1}$ satisfy

$$v_{j+1}^{(i+1)} = \left[I + \frac{(p_j - Q_j^{-1}Ap_j)p_j^T}{p_j^T Q_j^{-1} Ap_j}\right]v_j^{(i+1)} ,$$

by (8.8). The direction vector is given by

$$p_{i+1} = Q_{i+1}^{-1}r_{i+1} = v_{i+1}^{(i+1)} .$$

Moreover,

$$Q_j^{-1}Ap_j = Q_j^{-1}(r_j - r_{j+1}) = p_j - v_j^{(j+1)} ,$$

so that $v_{j+1}^{(i+1)}$ can be computed without the use of Q_j^{-1} as

$$v_{j+1}^{(i+1)} = \left[I + \frac{v_j^{(j+1)}p_j^T}{p_j^T(p_j - v_j^{(j+1)})}\right]v_j^{(i+1)} .$$

Letting $w_i := v_i^{(i+1)}$, Broyden's method for solving linear systems is as follows:

Algorithm 8.4: Broyden's method with direction update, for linear systems.

Choose x_0 .

Choose Q_0^{-1} .

Compute $p_0 = Q_0^{-1} r_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$x_{i+1} = x_i + p_i$

$r_{i+1} = r_i - Ap_i$

$v_0^{(i+1)} = Q_0^{-1} r_{i+1}$

FOR j = 0 STEP 1 UNTIL i DO

IF (j=i) THEN

$w_i = v_i^{(i+1)}$

$b_i = 1/p_i^T(p_i - w_i)$

$v_{j+1}^{(i+1)} = v_j^{(i+1)} + b_j p_j^T v_j^{(i+1)} w_j$

$p_{i+1} = v_{i+1}^{(i+1)}$

The work per loop is $[2(i+1) + 1]N + 1$ mv and one solve $Q_0^{-1}r$ (which

is a null operation if Q_0^{-1} is the identity matrix).* Storage is required for x, r, $\{p_j\}_{j=0}^{i+1}$, $\{v_j^{(i+1)}\}_{j=0}^{i+1}$, and $\{w_j\}_{j=0}^{i}$, for a total of $[3(i+1) + 1]N$. Ap can share storage with $v_0^{(i+1)}$.

An inductive argument shows that if $Q_0^{-1} = I$, then the residuals generated by Broyden's method satisfy

$$r_i = q_i(A)r_0, \qquad q_i \in P_i .$$

Compared with the expenses of GCR (see Table 5-1), the work per loop is somewhat lower but the storage requirements are greater. Since GCR computes the optimal polynomial with respect to $\|r_i\|_2$, we suspect that Broyden's method offers little advantage over GCR.

*If $Q_0^{-1} \neq I$, then the use of Q_0^{-1} in Algorithm 8.4 actually corresponds to preconditioning by Q_0. See Chapter 9.

CHAPTER 9

Preconditioning

9.1 Introduction

Consider the system of linear equations

$$A x = f , \tag{9.1}$$

where A is a nonsingular, nonsymmetric matrix of order N. In this chapter, we consider the use of preconditioning techniques in conjunction with the iterative methods of Chapters 4 - 8.

Let $Q = Q_1 Q_2$ denote a nonsingular matrix. The solution to (9.1) can be obtained by solving any of the problems

$$\text{left: } \tilde{A}\,\tilde{x} = [Q^{-1}A]\,[x] = Q^{-1}f = \tilde{f} ; \tag{9.2}$$

$$\text{right: } \tilde{A}\,\tilde{x} = [AQ^{-1}]\,[Qx] = f = \tilde{f} ; \tag{9.3}$$

$$\text{split: } \tilde{A}\,\tilde{x} = [Q_1^{-1}AQ_2^{-1}]\,[Q_2x] = Q_1^{-1}f = \tilde{f} . \tag{9.4}$$

The use of such an auxiliary matrix is known as preconditioning. We say that (9.2) - (9.4) are three different ways of applying the preconditioning, although left and right preconditioning can also be considered as special cases of split preconditioning. (For example,

left preconditioning reduces to split preconditioning with $Q_1 = Q$ and $Q_2 = I$.)

The goal of preconditioning is to decrease the computational effort needed to solve (9.1). If Q is in some sense a good approximation of A, then the coefficient matrices of (9.2) - (9.4) are in turn closer (in some sense) to the identity matrix than A, and the iterative methods of Chapters 4 - 8 will converge more rapidly than when applied to (9.1).

For preconditioning to be effective, the faster convergence must overcome the costs of applying the preconditioning, so that the total cost of solving (9.1) is lower. The preconditioned coefficient matrix $\tilde{A}$ is usually not explicitly computed or stored. The main reason for this is that although A is sparse, $\tilde{A}$ may not be. The extra work of preconditioning, then, occurs in the part of the preconditioned matrix-vector products involving Q^{-1} (or Q_1^{-1} and Q_2^{-1}). The main storage cost for preconditioning is for Q, which must be stored so that the operation $Q^{-1}v$ (or $Q_1^{-1}v$ and $Q_2^{-1}v$) can be performed efficiently. In addition, most of the iterative methods reqnire one extra vector of length N to handle the preconditioning operation.

In this chapter, we discuss some of the issues of implementing the various preconditioned iterative methods. In Section 9.2, we discuss preconditioned versions of CGN. In Section 9.3, we discuss preconditioned GCR and Orthomin(k) as representative of the preconditioned methods of Chapters 5 - 8. In Section 9.4, we discuss

some alternative implementations applicable when Q is symmetric and positive-definite. One example of this special case is the preconditioned generalized conjugate gradient method (see Section 7.2).

9.2 Preconditioning for the Normal Equations

We showed in Chapter 4 that there are two types of normal equations to which the conjugate gradient method can be applied, and that the norm minimized by CG differs for the two problems. When preconditioning is used, the norm minimized depends on both the choice of normal equations and the technique of applying the preconditioning [55]. In Table 9-1, we list the quantities minimized for the various combinations. The first three columns correspond to the three choices of preconditioning techiques (9.2) - (9.4), the fourth column to the special case in which Q is symmetric and positive-definite. In the latter case, the algorithms can be implemented so that the norm is independent of the factorization of Q (see Algorithms 9.3 and 9.4 below).

	Left	Right	Split	SPD Split
CGNR	$\|Q^{-1}r_i\|_2$	$\|r_i\|_2$	$\|Q_1^{-1}r_i\|_2$	$\|r_i\|_{Q^{-1}}$
CGNE	$\|x-x_i\|_2$	$\|Q(x-x_i)\|_2$	$\|Q_2(x-x_i)\|_2$	$\|x-x_i\|_Q$

Table 9-1: Error norm minimized by preconditioned CGN methods.

In the interest of brevity, we present implementations of just four of the methods given in Table 9-1 (cf. [44, 55]). CGNR/Right and CGNE/Left are noteworthy because the norms they minimize depend only on the original problem (9.1), and not on the preconditioning. (The Krylov spaces do depend on the preconditioniug.) The two instances of symmetric split preconditioning are also noteworthy, since they display good asymptotic properties for elliptic problems (see Section 10.4). Moreover, the only preconditioning operation in these four methods is a solve $Q^{-1}v$, so that an explicit factorization of Q is not required.

The cost per iteration of these four algorithms is two matrix-vector products (Av and A^Tv), two preconditioning solves ($Q^{-1}v$ and $Q^{-T}v$), and 5N multiplications. Storage is required for five vectors plus Q; the specific storage requirements for each of the methods follow its description.

Algorithm 9.1: CGNR with right preconditioning.

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $\tilde{p}_0 = Q^{-T}A^T r_0$.

Compute $p_0 = Q^{-1}\tilde{p}_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(Q^{-T}A^T r_i, Q^{-T}A^T r_i)}{(Ap_i, Ap_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i Ap_i$$

$$b_i = \frac{(Q^{-T}A^T r_{i+1}, Q^{-T}A^T r_{i+1})}{(Q^{-T}A^T r_i, Q^{-T}A^T r_i)}$$

$$\tilde{p}_{i+1} = Q^{-T}A^T r_{i+1} + b_i \tilde{p}_i .$$

$$p_{i+1} = Q^{-1}\tilde{p}_{i+1}$$

Storage: x, r, $\tilde{p}$; $A^T r$ shared with p; $Q^{-T}A^T r$ shared with Ap.

Algorithm 9.2: CGNE with left preconditioning.

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $\tilde{r}_0 = Q^{-1} r_0$.

Compute $p_0 = A^T Q^{-T}\tilde{r}_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(\tilde{r}_i, \tilde{r}_i)}{(p_i, p_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$\tilde{r}_{i+1} = \tilde{r}_i - a_i Q^{-1} Ap_i$$

$$b_i = \frac{(\tilde{r}_{i+1}, \tilde{r}_{i+1})}{(\tilde{r}_i, \tilde{r}_i)}$$

$$p_{i+1} = A^T Q^{-T}\tilde{r}_{i+1} + b_i p_i .$$

Storage: x, $\tilde{r}$, p; $Q^{-T}\tilde{r}_{i+1}$ shared with Ap; $A^T Q^{-T}\tilde{r}$ shared with $Q^{-1}Ap$.

Algorithm 9.3: CGNR with symmetric, positive-definite split preconditioning.

Choose x_0 .

Compute $r_0 = f - Ax_0$.

$r_0' = Q^{-1}r_0$

Compute $p_0 = Q^{-1}A^T r_0'$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(A^T r_i', Q^{-1}A^T r_i')}{(Ap_i, Q^{-1}Ap_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1}' = r_i' - a_i Q^{-1}Ap_i$$

$$b_i = \frac{(A^T r_{i+1}', Q^{-1}A^T r_{i+1}')}{(A^T r_i', Q^{-1}A^T r_i')}$$

$$p_{i+1} = Q^{-1}A^T r_{i+1}' + b_i p_i \ .$$

Storage: x, r', p; $A^T r'$ shared with Ap; $Q^{-1}A^T r'$ shared with $Q^{-1}Ap$.

Algorithm 9.4: CGNE with symmetric, positive-definite split preconditioning.

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $p_0' = A^T Q^{-1} r_0$.

Compute $p_0 = Q^{-1}p_0'$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(r_i, Q^{-1}r_i)}{(p_i, p_i')}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i Ap_i$$

$$b_i = \frac{(r_{i+1}, Q^{-1}r_{i+1})}{(r_i, Q^{-1}r_i)}$$

$$p_{i+1}' = A^T Q^{-1} r_{i+1} + b_i p_i'$$

$$p_{i+1} = Q^{-1}p_{i+1}' \ .$$

Storage: x, r, p'; p shared with $Q^{-1}r$; Ap shared with $A^T Q^{-1} r$.

9.3 Preconditioned GCR and Orthomin(k)

The preconditioned versions of most of the methods of Chapters 5 - 8 can be handled in essentially the same way.* Note that the technique of applying the preconditioning affects the norm associated with the minimizing methods of Chapters 5 - 6; while this question does not seem important for the methods of Chapters 7 and 8 that do not minimize a norm. We focus on the preconditioned versions of GCR and Orthomin(k) as representative of the preconditioned methods.

As we mentioned above, the "one-sided" (left or right) preconditionings are actually special cases of split preconditioning. In Algorithm 9.5, we present preconditioned GCR and Orthomin(k) in terms of split preconditioning. We point out specific issues of one-sided preconditioning when they arise.

If $j_i = 0$ for all i, then this algorithm is preconditioned GCR. If $j_i = \max(0, i-k+1)$, then it is preconditioned Orthomin(k).

The work per loop for these preconditioned methods is identical to that for the unpreconditioned versions (see Table 5-1), except that the matrix-vector product is replaced by a preconditioned matrix-vector product $Q_1^{-1}AQ_2^{-1}\tilde{r}_{i+1}$. In general, this operation is performed in three steps: a system of equations with coefficient matrix Q_2 is solved for

*The one exception is GCG, which we discuss in the next section.

Algorithm 9.5: Preconditioned GCR and Orthomin(k).

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $\tilde{r}_0 = Q_1^{-1}r_0$.

Set $p_0 = Q_2^{-1}\tilde{r}_0$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(\tilde{r}_i, Q_1^{-1}Ap_i)}{(Q_1^{-1}Ap_i, Q_1^{-1}Ap_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$\tilde{r}_{i+1} = \tilde{r}_i - a_i Q_1^{-1}Ap_i$$

$$b_j^{(i)} = -\frac{(Q_1^{-1}AQ_2^{-1}\tilde{r}_{i+1}, Q_1^{-1}Ap_j)}{(Q_1^{-1}Ap_j, Q_1^{-1}Ap_j)}, \qquad j=j_i,\ldots,i$$

$$p_{i+1} = Q_2^{-1}\tilde{r}_{i+1} + \sum_{j=j_i}^{i} b_j^{(i)} p_j$$

$$Q_1^{-1}Ap_{i+1} = Q_1^{-1}AQ_2^{-1}\tilde{r}_{i+1} + \sum_{j=j_i}^{i} b_j^{(i)} Q_1^{-1}Ap_j .$$

$Q_2^{-1}r_{i+1}$; the result is multiplied by A; and that result is used as the right hand side in a system of equations to be solved for $Q_1^{-1}[A\ Q_2^{-1}\tilde{r}_{i+1}]$. If one-sided preconditioning is used, then one of the preconditioning operations reduces to multiplication by the identity matrix, and the other involves the solution of a system of eqnations

$Q\,u = v$. One can also think of the preconditioned matrix-vector product as providing the intermediate vector $Q_2^{-1}r_{i+1}$ used in the computation of p_{i+1}. (That is, a subroutine can be used to compute the two vectors $Q_2^{-1}\tilde{r}_{i+1}$ and $Q_1^{-1}AQ_2^{-1}\tilde{r}_{i+1}$.) With this viewpoint, it is relatively easy to take advantage of the efficient techniques [20] developed for preconditionings based on the incomplete factorization of A (see Chapter 10).

The storage costs depend on the technique of applying the preconditioning. We assume that the preconditioning is implemented so that $Q_1^{-1}Av$ can overwrite Av. All three techniques require storage for x, $\tilde{r}$, $\{p_j\}$, $\{Q_1^{-1}Ap_j\}$, and $AQ_2^{-1}\tilde{r}$, which by assumption can share space with $Q_1^{-1}AQ_2^{-1}\tilde{r}$. As in the unpreconditioned version of GCR, $Q_1^{-1}AQ_2^{-1}\tilde{r}_{i+1}$ can also be overwritten by $Q_1^{-1}Ap_{i+1}$. For right and split preconditioning, one additional vector of storage is required for $Q_2^{-1}\tilde{r}$; for left preconditioning, $Q_2 = I$ so that $Q_2^{-1}\tilde{r} = \tilde{r}$. Thus, the left preconditioned versions of GCR and Orthomin(k) have the same storage requirements as the unpreconditioned versions (see Table 5-1), and the right and split versions require one extra vector of storage.*

*If the actual residual $r_i = Q_1\tilde{r}_i$ is required, then left preconditioning loses this edge, since it requires an extra vector of storage for r_i. For right preconditioning, $r_i = \tilde{r}_i$, and for split preconditioning, r_i can share storage with $Q_2^{-1}\tilde{r}_i$.

As with CGN, the technique of applying the preconditioning affects the norms and Krylov spaces associated with these methods. The quantities minimized are as follows:

$$\begin{aligned} &\|Q^{-1}r_i\|_2 , && \text{for left preconditioning} \\ &\|r_i\|_2 , && \text{for right preconditioning} \\ &\|Q_1^{-1}r_i\|_2 , && \text{for split preconditioning} . \end{aligned}$$

For GCR, the Krylov spaces in which these norms are minimized are all given by

$$x_0 + \langle p_0, Q^{-1}Ap_0, \ldots, (Q^{-1}A)^{i-1}p_0 \rangle ,$$

where the dependence on the technique of preconditioning is reflected in the definition of p_0:

$$p_0 = \begin{cases} Q^{-1}r_0 , & \text{for left preconditioning} \\ r_0 , & \text{for right preconditioning} \\ Q_1^{-1}r_0 , & \text{for split preconditioning} . \end{cases}$$

We have generally favored right preconditioning with the variational methods since the norm minimized is independent of preconditioning. For nearly symmetric problems, split preconditioning may have an advantage, since the preconditioned matrix $Q_1^{-1}AQ_2^{-1}$ may also be nearly symmetric.

9.4 Symmetric Positive-Definite Preconditioning

Assume that Q is symmetric and positive-definite with factorization $Q = SS^T$, and consider split preconditioning by Q:

$$\tilde{A}\,\tilde{x} = [S^{-1}AS^{-T}]\,[S^T x] = S^{-1}f = \tilde{f}\ . \tag{9.5}$$

Like CGN, the variational methods of Chapters 5 - 6 can be implemented to solve (9.5) without reference to the factorization of Q, as shown in Algorithm 9.6.

The work per loop is the same as that for Algorithm 9.5, except that the matrix-vector product Ar' and preconditioning operation $Q^{-1}Ap$ are separated. Storage is required for x, r', $\{p_j\}$, $\{Ap_j\}$, and Ar'. $Q^{-1}Ap$ can share storage with Ar', and for GCR, this vector can be overwritten by Ap_{i+1}.

If $Q = M$, then $\tilde{A} = I - \tilde{R}$ with $\tilde{R}$ skew-symmetric, so that the error bound of Theorem 5.10 holds:

Theorem 9.7: The residuals generated by Orthomin(1) with split preconditioning by the symmetric part satisfy

$$\|r_i\|_{M^{-1}} \le \frac{2}{C(\rho(M^{-1}R))^i + [-C(\rho(M^{-1}R))]^{-i}}\,\|r_0\|_{M^{-1}}\ ,$$

where

$$C(\lambda) = \frac{\sqrt{1+\lambda^2}+1}{\lambda}\ .$$

Algorithm 9.6: GCR and Orthomin(k) with symmetric, positive-definite split preconditioning.

Choose x_0 .

Compute $r_0 = f - Ax_0$.

Compute $r_0' = Q^{-1}r_0$.

Set $p_0 = r_0'$.

FOR i = 0 STEP 1 UNTIL Convergence DO

$$a_i = \frac{(r_i', Ap_i)}{(Ap_i, Q^{-1}Ap_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1}' = r_i' - a_i Q^{-1}Ap_i$$

$$b_j^{(i)} = -\frac{(Ar_{i+1}', Q^{-1}Ap_j)}{(Ap_j, Q^{-1}Ap_j)}\ , \qquad j=j_i,\ldots,i$$

$$p_{i+1} = r_{i+1}' + \sum_{j=j_i}^{i} b_j^{(i)} p_j$$

$$Ap_{i+1} = Ar_{i+1}' + \sum_{j=j_i}^{i} b_j^{(i)} Ap_j\ .$$

The symmetric part is the only preconditioning that can be used with the generalized conjugate gradient method. This preconditioned method also can be implemented without explicit reference to the factors of M.

Algorithm 9.8: The preconditioned generalized conjugate gradient method.

Set $x_{-1} = 0$.

Choose x_0 .

FOR i = 0 STEP 1 UNTIL Convergence DO

$$r_i = f - Ax_i$$

$$v_i = M^{-1}r_i$$

$$\eta_i = (v_i, Mv_i)$$

$$\omega_{i+1} = \begin{cases} 1 , & \text{if } i = 0 \\ (1+(\eta_i/\eta_{i-1})/\omega_i)^{-1} , & \text{if } i \geq 1 \end{cases}$$

$$x_{i+1} = x_{i-1} + \omega_{i+1}(v_i + x_i - x_{i-1}) .$$

The work per loop of preconditioned GCG is 2N multiplications plus one matrix-vector product and one solve $M^{-1}r$. Storage is required for x_i, x_{i-1}, r, and v, plus M.

The analogue of Theorem 7.2 is as follows:

Theorem 9.9: The error at the i^{th} step of preconditioned GCG satisfies

$$\|x-x_i\|_M \leq \frac{2}{C(\rho(M^{-1}R))^i + [-C(\rho(M^{-1}R))]^{-i}} \|x-x_0\|_M .$$

where $C(\lambda)$ is as in Theorem 9.7.

Finally, the virtual equivalence of GCG and CGNE extends naturally to the preconditioned versions:

Theorem 9.10: Let $\{x_i\}$ denote the iterates generated by preconditioned GCG, let $\{\tilde{x}_i\}$ denote the iterates generated by CGNE with split preconditioning (Algorithm 9.4) by the symmetric part, and assume that $x_0 = \tilde{x}_0$. Then $x_{2i} = \tilde{x}_i$.

CHAPTER 10

Some Preconditioning Techniques

10.1 Introduction

Consider the system of linear equations

$$A x = f , \tag{10.1}$$

where A is a nonsingular, nonsymmetric matrix of order N. In this chapter, we consider some examples of preconditioning techniques that can be used with the iterative methods of Chapters 4 - 8 for solving (10.1).

We seek a matrix Q that is in some sense a good approximation of A, and that is inexpensive to use in the preconditioned algorithms. By a good approximation of A, we mean roughly that the condition number of the preconditioned coefficient matrix $\tilde{A}$ should be small, or the eigenvalues of $\tilde{A}$ be tightly clustered. The error bounds of Chapters 4 - 6 do not suggest a "best" definition for this concept. Moreover, it is usually difficult to analyze the effect of a particular preconditioning for nonsymmetric matrices. As a result, preconditioning is more of an art than a science, in which approximations Q are

developed from heuristic notions of closeness to A and from techniques known to be effective for symmetric problems.

In Section 10.2, we consider preconditioning matrices constructed from approximate factorizations of the coefficient matrix. A lower triangular matrix L and an upper triangular matrix U are constructed that are in some sense approximations of the factors in the LU factorization of A, but that are also sparse. The preconditioning matrix is the product Q = LU. We consider two such approximate factorizations, the incomplete LU factorization (ILU) [50, 51], and the modified incomplete LU factorization (MILU) [19, 35, 36]. In Section 10.3, we discuss the related SSOR-preconditioning, which uses an approximate factorization of A derived from the symmetric successive over-relaxation iterative method [78].

In Section 10.4, we consider fast direct methods as preconditionings for linear systems arising from the discretization of elliptic partial equations. We use the error bounds of Chapters 4 - 7 and 9 to derive bounds on the asymptotic operation counts of the preconditioned iterative methods, and we discuss the existing fast direct methods.

Finally, in Section 10.5, we consider the preconditioning that forms a reduced system to solve (10.1). If A is a two-cyclic matrix [73], then with a small amount of preprocessing some of the unknowns can be eliminated from (9.1) to produce a reduced linear system

of order $m \leq \frac{N}{2}$. This preconditioning differs from the others in that only a subproblem of the preconditioned problem is actually solved by an iterative method, and this smaller problem can be solved using any of the techniques (including other preconditionings) that we have dicussed.

10.2 Approximate Factorizations

We consider two preconditionings based on the approximate factorization of A. The heuristic used to insure that the preconditioning is inexpensive to implement is to force the factors to be sparse by allowing nonzeros only within a specified set of locations.

The first technique is the incomplete LU factorization (ILU) popularized by Meijerink and van der Vorst [50, 51]. Let Z be a set of indices contained in $\{(i,j) \mid 1 \leq i,j \leq N\}$. The ILU factorization is given by $Q = LU$, where L and U are lower triangular and unit upper triangular matrices, respectively, that satisfy

$$\text{if } (i,j) \in Z, \text{ then } L_{ij} = 0 \text{ and } U_{ij} = 0 \ ; \tag{10.2}$$

$$\text{if } (i,j) \notin Z, \text{ then } Q_{ij} = A_{ij} \ . \tag{10.3}$$

Thus, the approximate factors are forced to be zero at the indices in Z, and the product Q agrees with A at all indices (i,j) not in Z. The second requirement can be imposed by formally applying a Gaussian elimination step at all indices $(i,j) \notin Z$. For example, the following algorithm computes the nonzero entries of the factors by a modification of the Crout version of Gaussian elimination [41].

Algorithm 10.1: The incomplete LU factorization.

```
FOR i = 1 STEP 1 UNTIL N DO
   FOR j = 1 STEP 1 UNTIL N DO
      IF ( (i,j) ∉ Z ) THEN
         s_ij = A_ij - Σ_{t=1}^{min(i,j)-1} L_it U_tj
         IF (i ≥ j) THEN L_ij = s_ij
         IF (i < j) THEN U_ij = s_ij / L_ii .
```

This algorithm is well-defined provided that $L_{ii} \neq 0$ for all i. Meijerink and van der Vorst [51] show that if Z does not contain any diagonal indices $\{(i,i)\}_{i=1}^{N}$ and A is an M-matrix, then this factorization is well-defined and determines a regular splitting of A [73]. They also give empirical evidence that when applied to symmetric problems arising from the discretization of self-adjoint elliptic partial differential equations, the eigenvalues of the preconditioned system are tightly clustered, so that this is an effective technique to use in conjunction with the conjugate gradient method. (See also [44].) Typically, Z is chosen to be the set of indices for which the entries of A are zero, or some slightly smaller set (so that the factors are slightly less sparse than A).

The second approximate factorization that we consider is the modified incomplete LU factorization (MILU) proposed by Gustafsson [35, 36] and derived from the iterative method developed for

elliptic partial differential equations by Dupont, Kendall, and Rachford [19]. Let Z be the set of indices that determine the zero structure, and assume that $(i,i) \notin Z$, $1 \le i < N$. The modified incomplete LU factorization is given by $Q = LU$, where L and U are lower triangular and unit upper triangular matrices, respectively, that satisfy

$$\text{if } (i,j) \in Z, \text{ then } L_{ij} = 0 \text{ and } U_{ij} = 0 \; ; \tag{10.4}$$

$$\text{if } (i,j) \notin Z \text{ and } i \neq j, \text{ then } Q_{ij} = A_{ij} \; ; \tag{10.5}$$

$$\text{for } 1 \le i \le N, \; \sum_{j=0}^{N} (Q_{ij} - A_{ij}) = \alpha \; , \tag{10.6}$$

where α is a scalar. Let $E := Q - A$ denote the error matrix. The extra defining condition (10.6) of the MILU factorization is that the row sums of E equal α. The difference is effected by modifying the ILU computation of the diagonal entries of L so that this condition is satisfied. An implementation is given by Algorithm 10.2.

Assume that this algorithm is well-defined, i.e., that $L_{ii} \neq 0$ for all i. We demonstrate that the computed factors satisfy (10.6). The essential observation is that for $(i,j) \in Z$,

$$E_{ij} = - s_{ij} \; . \tag{10.7}$$

The residual entries satisfy

$$E_{ij} = Q_{ij} - A_{ij} = \sum_{t=1}^{\min(i,j)} L_{it}U_{tj} - A_{ij} \; . \tag{10.8}$$

Algorithm 10.2: The modified incomplete LU factorization.

```
FOR i = 1 STEP 1 UNTIL N DO
    L_ii = α
    FOR j = 1 STEP 1 UNTIL N DO
        s_ij = A_ij - Σ_{t=1}^{min(i,j)-1} L_it U_tj
        IF ( (i,j) ∉ Z ) THEN
            IF (i > j) THEN L_ij = s_ij
            IF (i = j) THEN L_ii = L_ii + s_ii
            IF (i < j) THEN Ũ_ij = s_ij
        ELSE L_ii = L_ii + s_ij
    FOR j = i+1 STEP 1 UNTIL N DO
        U_ij = Ũ_ij / L_ii .
```

But if $j < i$, then $(i,j) \in Z$ implies that $L_{ij} = 0$, so that the upper limit of the sum in (10.8) is actually j-1. Similarly, if $j > i$, then the upper limit is i-1. Hence, (10.7) follows from the definition of s_{ij} in Algorithm 10.2. Now consider $\{E_{ij}\}$ for $(i,j) \notin Z$. If $j \neq i$, then $E_{ij} = 0$ by (10.5). If $j = i$, then

$$E_{ii} = \sum_{t=1}^{i} L_{it}U_{ti} - A_{ii} = \Big[\sum_{t=1}^{i-1} L_{it}U_{ti} - A_{ii}\Big] + L_{ii}$$

$$= -s_{ii} + s_{ii} + \sum_{(i,j)\in Z} s_{ij} + \alpha \; .$$

Thus, the contributions to the i^{th} row sum of E from the off-diagonal entries are cancelled by the contribution from the diagonal, and

$$\sum_{j=1}^{N} E_{ij} = \alpha \ .$$

The MILU factorization is well-defined provided that the pivot element at each step is nonzero. The following result establishes sufficient conditions for this to hold.

Theorem 10.3: If A is strictly diagonally dominant, then the MILU factorization is well-defiued for $\alpha \geq 0$. If A is irreducibly diagonally dominant, then there exists a permutation matrix P such that any MILU factorization of PAP^T that is at least as dense as PAP^T is well-defined for $\alpha \geq 0$.

This result is due to Gustafsson [36]. Although the statement of the theorem in this reference does not mention the permutation matrix, the proof depends on it. Indeed, the MILU factorization may not be well defined for irreducibly diagonally dominant matrices whose rows are not ordered suitably. For example, let

$$A = \begin{vmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 2 \end{vmatrix} ,$$

and $Z = \{(2,3),(3,2)\}$. Execution of Algorithm 10.2 shows that $L_{2,2} = 0$,

so that the MILU factorization breaks down. If rows one and three and columns one and three are interchanged, then the factorization is well defined.

The parameter α can be used as a "tuning mechanism" to speed the convergence of MILU-preconditioned algorithms. In some applications, $\alpha = 0$ has been found to be the best value (see [12] and Chapter 11), so that the MILU factorization could be considered independent of parameters.

Several other incomplete factorization techniques have been used with success, including the strongly implicit (SIP) [68], alternating direction implicit (ADI) (see [73, 78]), and shifted incomplete LU [47] factorizations. Unlike the MILU factorization, these methods are sensitive to the choices of one or more scalar parameters. (See [11] for a recently proposed generalization of the MILU factorization known as ADDKR for use with elliptic partial differential equations.)

10.3 SSOR Preconditioning

Let

$$A = D - L - U \ ,$$

where D is the diagonal of A, L is the strict lower triangle of A, and U is the strict upper triangle of A. The symmetric successive over-relaxation (SSOR) iterative method [78] is the following two-stage algorithm:

$$(D-\omega L)x_{i+1/2} = [(1-\omega)D + \omega U]x_i + \omega f$$
$$(D-\omega U)x_{i+1} = [(1-\omega)D + \omega L]x_{i+1/2} + \omega f ,$$

where ω is a real scalar parameter between 0 and 2. With

$$Q := \frac{1}{\omega(2-\omega)}(D-\omega L)D^{-1}(D-\omega U) ,$$

this method can be formulated as a one stage algorithm

$$Qx_{i+1} = (Q-A)x_i + f .$$

Q is the SSOR preconditioning matrix.

The SSOR preconditioning has been shown to be effective when used with the conjugate gradient method for symmetric, positive-definite problems [3, 12]. It is not as sensitive to the value of ω as the SOR method, and some techniques for estimating the best value for ω have been developed [78]. Unfortunately, we know of no techniques for estimating ω for nonsymmetric problems, and the SSOR preconditioning is sensitive to ω in this case (see Chapter 11).

We consider SSOR preconditioning despite these drawbacks because it is easy to implement. Q can be expressed as

$$Q = \frac{1}{2-\omega}(\tfrac{1}{\omega}D-L)(\tfrac{1}{\omega}D)^{-1}(\tfrac{1}{\omega}D-U) .$$

Since the iterates generated by a preconditioned algorithm are independent of multiplicative constants in the preconditioning, the

factor $\frac{1}{2-\omega}$ can be ignored. Thus, the SSOR preconditioning requires at most one vector of length N for storage, for $\tilde{D} := \frac{1}{\omega}D$. Moreover, when formulated this way, the SSOR preconditioned matrix-vector product can be computed efficiently [20]. For example, consider the product $Q^{-1}Au$. Let N_L and N_U denote the number of nonzeros in L and U respectively. If the product is computed naively, then $4N + 2N_L + 2N_U$ multiplications are required. Using the techniques developed in [20], just $3N + N_L + 2N_U$ multiplications are needed. Thus, if a good value for ω can be found, SSOR may be an effective preconditioning.

10.4 Fast Direct Methods

Fast direct methods can be used as preconditionings for linear systems arising from the discretization of elliptic partial differential equations of the form (2.2), on rectangular domains. Recall that the discretization of (2.2) by the five-point operator on a uniform n x n grid results in a block-tridiagonal system of linear equations

$$A z = f \tag{10.9}$$

of order $N = n^2$. If A is separable, then (10.9) can be solved by fast direct methods, which require $O(n^2 \log_2 n)$ operations (see below). If it is nonseparable but self-adjoint, then fast direct methods can be combined with the Chebyshev algorithm [14] or the conjugate gradient method [6, 7] to solve (10.9) efficiently. In particular, the error bounds for the Chebyshev method and CG are independent of h. We show that these ideas can be extended to the nonseparable, non-self-adjoint

case, using CGN, GCG, and the variational techniques of Chapters 5 and 6. We state the results for the latter techniques in terms of Orthomin(k) only, although they also apply to GCR, GCR(k), MR, LSGCR, AXEL(k), and Orthodir.

10.4.1 Convergence Results

The symmetric part M of A is positive-definite [29]. As Widlund [75] observes, since the skew-symmetric part R is derived from a differential operator of lower order than that producing M, the eigenvalues of $M^{-1}R$ are bounded independent of h for small h.

Let $M = SS^T$, and consider split preconditioning by M:

$$A_1\tilde{z} = [S^{-1}AS^{-T}]\ [S^Tz] = S^{-1}f = f_1 \ . \tag{10.10}$$

The error bounds for GCG and Orthomin(1) applied to (10.10) depend only on $\rho(M^{-1}R)$ (see Theorems 9.7 and 9.9). Hence, these bounds are independent of h. In light of the virtual-equivalence of GCG and CGNE established in Section 9.4, it follows immediately that the convergence of CGNE with split preconditioning by M is also independent of h. Alternatively, by (4.5),

$$K(A_1) \leq 1 + \rho(S^{-1}RS^{-T}) = 1 + \rho(M^{-1}R) \ ,$$

so that the error bound (4.3) for both CGNE and CGNR is independent of h.

Each step of these preconditioned algorithms requires the solution

of a system of equations with coefficient matrix M (see Sections 9.2 and 9.4). If M is separable, then these systems can be solved with fast direct methods in $O(n^2\log_2 n)$ operations. The number of iterations needed to reduce the error by a given factor ε is independent of mesh size. If the error is to be reduced to truncation, then ε is proportional to n^{-2}, and the number of iterations required is proportional to $\log_2 n$. We summarize these observations as follows:

Theorem 10.4: The asymptotic operation counts of Orthomin(1), GCG, CGNE, and CGNR with split preconditioning by the symmetric part for solving (10.9) to truncation error are $O(n^2(\log_2 n)^2)$.

If M is not separable, then fast direct methods are not applicable to (10.10). However, there often exists a separable, symmetric, positive-definite matrix Q that satisfies

$$c_1(z,Qz) \leq (z,Mz) \leq c_2(z,Qz) \tag{10.11}$$

for all z, where c_1 and c_2 are positive constants [14]. For example, Q might be the discrete analogue of the negative Laplacian $-\left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right]$. If $Q = VV^T$, then Q can be used to precondition (10.9) to produce the linear system of equations

$$A_2\tilde{z} = [V^{-1}AV^{-T}]\ [V^Tz] = [V^{-1}f] = f_2 \ . \tag{10.12}$$

The following result shows that Q is essentially as effective as M for preconditioning (10.9).

Theorem 10.5: Let $Q = VV^T$ be a symmetric, positive-definite matrix that satisfies (10.11). Let $A_2 = V^{-1}AV^{-T}$, with symmetric part $M_2 = V^{-1}MV^{-T}$ and skew-symmetric part $R_2 = V^{-1}RV^{-T}$. Then

$$c_1 \leq \lambda(M_2) \leq c_2 \ , \tag{10.13}$$

$$\rho(R_2) \leq c_2 \ \rho(M^{-1}R) \ , \tag{10.14}$$

and

$$K(A_2^T A_2) \leq \frac{\left[c_2(1 + \rho(M^{-1}R)\right]^2}{c_1} \ . \tag{10.15}$$

Proof: For (10.13),

$$\lambda_{min}(M_2) = \min_{z \neq 0} \frac{(z, V^{-1}MV^{-T}z)}{(z,z)} = \min_{u \neq 0} \frac{(u,Mu)}{(u,Qu)} \geq c_1 \ ,$$

by (10.11). Similarly, $\lambda_{max}(M_2) \leq c_2$.

For (10.14),

$$\rho(R_2)^2 = \max_{z \neq 0} \frac{(V^{-1}RV^{-T}z, V^{-1}RV^{-T}z)}{(z,z)} = \max_{u \neq 0} \frac{(Ru, Q^{-1}Ru)}{(u,Qu)} \ .$$

But

$$\frac{1}{(u,Qu)} \leq c_2 \frac{1}{(u,Mu)} \ .$$

Moreover, we claim that

$$(Ru, Q^{-1}Ru) \leq c_2 (Ru, M^{-1}Ru) \ .$$

Hence,

$$\rho(R_2)^2 \leq \max_{u \neq 0} \frac{(Ru, M^{-1}Ru)}{(u,Mu)} = \max_{v \neq 0} \frac{(S^{-1}RS^{-T}v, S^{-1}RS^{-T}v)}{(v,v)}$$

$$= \rho(M^{-1}R)^2 \ .$$

To prove the claim, we show that

$$\max_{v \neq 0} \frac{(v, Q^{-1}v)}{(v, M^{-1}v)} \leq c_2 \ .$$

Let v be a vector at which $\frac{(v,Q^{-1}v)}{(v,M^{-1}v)}$ attains its maximum, μ. Then (μ, v) is a solution to the generalized eigenvalue problem

$$Q^{-1}v = \mu \ M^{-1}v$$

(see [76]). Let $w = M^{-1}v$, so that

$$Mw = \mu Qw \ .$$

Hence,

$$\mu = \frac{(w,Mw)}{(w,Qw)} \leq c_2 \ .$$

Finally, (10.15) follows from (10.13), (10.14), and the following inequality:

$$\lambda_{min}(M_2) \leq \lambda(A_2^T A_2) \leq \left[\lambda_{max}(M_2) + \rho(R_2)\right]^2 \ ,$$

which is a consequence of Theorem 4.3.

Q.E.D.

Corollary 10.6: If Q is a symmetric matrix that satisfies (10.11), then the asymptotic operation counts of Orthomin(k), CGNE, and CGNR with split preconditiong by Q are $O(n^2(\log_2 n)^2)$.

Proof: The residuals generated by Orthomin(k) satisfy

$$\|r_i\|_Q^{-1} \leq \left[1 - \frac{c_1^2}{c_2^2 + c_2^2\rho(M^{-1}R)^2}\right]^{i/2} \|r_0\|_Q^{-1} ,$$

by Theorem 5.9 and Theorem 10.5.

Similarly, by the error bound (4.3) for CGN and Theorem 10.5, the CGN iterates satisfy

$$E(x_i) \leq 2\left[\frac{c_2(1+\rho(M^{-1}R)) - c_1}{c_2(1+\rho(M^{-1}R)) + c_1}\right] E(x_0) ,$$

where

$$E(x_i) := \begin{cases} \|r_i\|_Q^{-1} & \text{for CGNR} , \\ \|x-x_i\|_Q & \text{for CGNE} . \end{cases}$$

Hence, the error bounds for these algorithms are independent of h.

Q.E.D.

This result shows an advantage of Orthomin(k) and CGN over GCG. GCG requires the solution of a subproblem with coefficient matrix M at each step. If M is not separable, then this may be an expensive task. In contrast, Orthomin(k), CGNE, and CGNR can be combined with an alternative preconditioning based on fast direct methods to solve

(10.9) even if M is not separable. (See [34] for an alternative GCG-like iteration for nonseparable M.)

Note that the use of split preconditioning in (10.12) requires that Q be symmetric and positive-definite. Since fast direct methods are applicable to separable nonsymmetric matrices as well [69], it may be preferable to use a separable preconditioning matrix Q that contains some approximation to the skew-symmetric part of A. Either of the one-sided formulations for the preconditioned problem (9.2) or (9.3) are suitable for such matrices. Also, as stated in Section 9.2, the norms minimized by Algorithms 9.1 and 9.2 and Algorithm 9.5 with right preconditioning depend only on the original problem and not on the preconditioning. We do not have an error bound analogous to Corollary 10.6 for snch preconditioning techniques. See Chapter 11 for numerical experiments with these techniques.

10.4.2 Examples of Fast Direct Methods

We conclude this section with a brief discussion of the known fast direct methods. These methods comprise the following techniques:

1. the cyclic reduction algorithm ;
2. Fourier analysis ;
3. the generalized marching algorithm .

Definitions and a complete list of references for the first two methods can be found in the survey papers [18, 70]. The generalized marching algorithm is described in [7, 8]. Our present concern is to outline the

advantages, limitations, and costs of these methods.

The preconditioning matrix Q is based on the discretization of a separable approximation of the differential operator in (2.2),

$$-(\tilde{B}u_x)_x - (\tilde{C}u_y)_y + \tilde{D}u_x + (\tilde{D}u)_x + \tilde{E}u_y + (\tilde{E}u)_y + \tilde{F}u = G \ , \qquad (10.16)$$

i.e.,

$$\tilde{B} = \tilde{B}(x) \ , \qquad \tilde{C} = \tilde{C}(y) \ , \qquad \tilde{D} = \tilde{D}(x) \ ,$$
$$\tilde{E} = \tilde{E}(y) \ , \qquad \tilde{F} = \tilde{F}_1(x) + \tilde{F}_2(y) \ .$$

Thus, one of the important issues of this preconditioning technique is the choice of these coefficient functions. The other issues are the particular choice of fast direct method and the technique of applying the preconditioning. To a great extent, these issues are interdependent. For example, Q is nonsymmetric if $\tilde{D}$ or $\tilde{E}$ is nonzero, but not all the methods are generally applicable to nonsymmetric problems. Also, as we have shown in Chapter 9, the symmetry of Q figures in the way it is applied as a preconditioning.

The choice of the approximating functions is highly problem dependent, and we do not address it here (see [6]). We briefly address the second issue in Table 10-1, which contains the asymptotic operation counts of the four methods and outlines restrictions on operators to which they can be applied.

The best technique to use for a given problem is not generally

Method	Operators	Operation Count
Cyclic Reduction	General separable	Self-adjoint: $5n^2\log_2 n$ Non-self-adjoint: $20n^2\log_2 n$
Fourier Analysis	Self-adjoint separable, non-self-adjoint if constant coefficient * in at least one variable	$2n^2\log_2 n$
Generalized Marching	Self-adjoint, separable	$12n^2\log_2 \frac{n}{s}$ **

Table 10-1: Properties of fast direct methods.

known (see [65]). We remark that the operation counts given in Table 10-1 contain only the highest order terms, and they reflect the assumption that the coefficients in (10.16) are not constant. (The coefficients are smaller if Q is derived from a constant coefficient operator.) The actual performance of these techniques may be affected by lower order terms, as well as idiosyncrasies of the computing environment.

Finally, lower asymptotic operation counts can be obtained from

---

*Either $\tilde{B}$, $\tilde{D}$, and $\tilde{F}_1$ are constant, or $\tilde{C}$, $\tilde{E}$, and $\tilde{F}_2$ are constant.

**The integer parameter s figures in the stability of the generalized marching algorithm. On a t-digit machine, error bounds are of order $c^s 2^{-t}$, with $c > 1$.

judicious combinations of these methods. A combination of Fourier analysis with cyclic reduction results in the FACR method [70], with an operation count of $5n^2\log_2\log_2 n$. For constant coefficient operators, a combination of the generalized marching algorithm with Fourier techniques leads to a method requiring $O(n^2)$ operations [8].

10.5 Reduced Systems

If A is a two-cyclic matrix [73], then its rows and columns can be permuted symmetrically so that (10.1) has the form

$$A\,x = \begin{vmatrix} D_1 & C_1 \\ C_2 & D_2 \end{vmatrix} \begin{vmatrix} x^{(1)} \\ x^{(2)} \end{vmatrix} = \begin{vmatrix} f^{(1)} \\ f^{(2)} \end{vmatrix} = f\ ,$$

where D_1 and D_2 are diagonal matrices of order m_1 and m_2. Without loss of generality, $m_1 \geq m_2$. Premultiplying by

$$\begin{vmatrix} D_1^{-1} & 0 \\ -C_2D_1^{-1} & I_{m_2} \end{vmatrix}$$

results in the preconditioned problem

$$\begin{vmatrix} I_{m_1} & D_1^{-1}C_1 \\ 0 & D_2-C_2D_1^{-1}C_1 \end{vmatrix} \begin{vmatrix} x^{(1)} \\ x^{(2)} \end{vmatrix} = \begin{vmatrix} D_1^{-1}f^{(1)} \\ f^{(2)}-C_2D_1^{-1}f^{(1)} \end{vmatrix}$$

in which the block unknowns $x^{(1)}$ and $x^{(2)}$ are decoupled. Block $x^{(2)}$ can then be computed by solving the reduced system

$$[D_2 - C_2D_1^{-1}C_1]\,x^{(2)} = f^{(2)} - C_2D_1^{-1}f^{(1)} \qquad (10.17)$$

of order m_2, with $x^{(1)}$ recovered by

$$x^{(1)} = D_1^{-1}(f^{(1)} - C_1x^{(2)})\ .$$

Typically $m_2 \tilde{=} \frac{N}{2}$.

In contrast to the other preconditioning techniques, the coefficient matrix and right-hand side of (10.17) are computed explicitly. This enables (10.17) to be solved by most of the other preconditioned iterative methods discussed in this dissertation. (This computation is not necessary if no additional preconditioning is used to solve (10.17). See [12] for a discussion of this alternative in the symmetric case; see [9] for a discussion of preconditioning for a coefficient matrix that is not explicitly formed.) If C_1 and C_2 are sparse, then the preprocessing step to decouple the block unknowns is inexpensive and the coefficient matrix in the reduced system is sparse.

CHAPTER 11

Numerical Experiments

11.1 Introduction

In this chapter, we describe the performance of some of the iterative methods and preconditionings discussed earlier. The test problems are linear systems arising from the discretization of several non-self-adjoint elliptic partial differential equations. In Section 11.2, we describe the differential equations and resulting discrete problems. In Section 11.3, we discuss some implementation issues. In Section 11.4, we present the results of numerical experiments with the conjugate gradient method applied to the normal equations (see Chapter 4), the minimizing methods of Chapter 5, and the Chebyshev and hybrid methods of Chapter 8, combined with the preconditioning techniques of Chapter 10. Other numerical experiments are described in [2, 5, 23, 24, 34, 46, 48, 60, 61, 75, 80].

11.2 The Test Problem

Consider the elliptic partial differential equation

$$-(Bu_x)_x - (Cu_y)_y + Eu_y + (Eu)_y + Fu = G \,, \tag{11.1}$$

where

$$B(x,y) = e^{-xy} \,, \qquad C(x,y) = e^{xy} \,,$$

$$E(x,y) = \gamma(x+y) \,, \qquad F(x,y) = \frac{1}{1+x+y} \,,$$

γ is a real scalar parameter, and the right hand side G is chosen so that

$$u(x,y) = x\, e^{xy} \sin(\pi x) \sin(\pi y)$$

is the solution to (11.1). For the test problem, we pose (11.1) on the unit square $0 \leq x \leq 1$, $0 \leq y \leq 1$, with homogeneous Dirichlet boundary conditions.

We discretize (11.1) using the five-point centered finite difference scheme [29, 73] on a uniform n x n grid, with $h = \frac{1}{n+1}$, producing a linear system

$$A z = f \tag{11.2}$$

of order $N = n^2$. The symmetric part of A is determined by the second- and zero-order terms of (11.1), and is positive-definite [29]. The skew-symmetric part is determined by the first-order terms. The cost of the matrix-vector product Av is approximately 5N

multiplications.

In the numerical experiments, we use values of h = 1/16, 1/32, 1/48, and 1/64, which result in linear systems of order 225, 961, 2209, and 3969 respectively. We use the values γ = 5, 50, and 250.

11.3 Implementation Issues

The iterative methods CGNR, Orthomin(k), GCR(k), MR, the Chebyshev algorithm, and the hybrid method consisting of GCR followed by the Chebyshev algorithm are used in conjunction with the preconditioning techniques of Chapter 10. We outline some of the issues and costs concerning the implementation of these methods below.

Chebyshev method: We use the Chebyshev code TCHEB written by Manteuffel [48], modified to handle preconditioned matrix-vector products. The initial choice of the ellipse parameters c and d (see Section 8.2) is as follows: with no preconditioning, d is taken to be the average of the diagonal elements of the coefficient matrix; for preconditioned problems, d is initialized to 1. In both cases, c is initially chosen to be 0. New parameters are computed at most every 20 iterations (the adaptive procedure is invoked earlier if the residual norm is increasing rapidly). The overhead for the adaptive procedure (14N multiplications [46]) is not included in the operation counts.

MILU, ILU, and SSOR preconditioning: Except for the reduced systems, we use the efficient implementation of these factorizations due to Eisenstat [20]. The cost of the preconditioned matrix-vector product is 9N multiplications. Unless otherwise indicated, α = 0 is used as the MILU parameter.

Fast direct preconditioning: We use the cyclic reduction algorithm implemented in the routine BLKTRI in the FISHPACK subroutine package [71]. The number of multiplications for the preconditioning solve $Q^{-1}v$ is approximately

$$20n^2[\log_2(n+1)] - 55n^2 + 40n[\log_2(n+1)] ,$$

where [x] denotes the largest integer less than or equal to x (see [69]). We approximate the coefficient functions of (11.1) by

$$\tilde{B}(x) = B(x,\tfrac{1}{2}) , \qquad \tilde{C}(y) = C(\tfrac{1}{2},y) , \qquad (11.3)$$
$$\tilde{E}(y) = \delta(y)E(\tfrac{1}{2},y) , \qquad \tilde{F}(x,y) = \tfrac{1}{2}F(x,\tfrac{1}{2}) + \tfrac{1}{2}F(\tfrac{1}{2},y) .$$

The scaling factor $\delta(y)$ in $\tilde{E}(y)$ is introduced to prevent the off-diagonal entries of Q from being too large and violating an error condition in FISHPACK. It satisfies $0 < \delta(y) \leq 1$, and is identically 1 for γ = 5, all h, and γ = 50, $h \leq 1/48$.

Reduced system preconditioning: The cost of the matrix-vector product for the reduced system is approximately $\frac{9}{2}$N multiplications. The MILU preconditioning operation costs an additional $\frac{9}{2}$N multiplications. Hence, the preconditioned matrix-vector product for the reduced system costs approximately 9N multiplications.

Note that the terms "cyclic reduction" and "reduced system" here refer to two distinct preconditionings. Cyclic reduction is one example of a fast direct method, whereas reduced system refers to a way of eliminating some of the unknowns from (11.2). See Chapter 10.

11.4 Numerical Results

In all tests, we use right preconditioning as in (9.3) so that the norm of the residual of the unpreconditioned system is minimized by CGNR and Orthomin(k), et. al. The initial guess is the zero vector. The stopping criterion is

$$\frac{\|r_i\|_2}{\|r_0\|_2} \leq 10^{-6}.$$

The tests were run in optimized FORTRAN-20 on a DECSYSTEM 2060. With the exception of the experiments using the cyclic reduction preconditioning, all tests were run in double precision (63 bit mantissa). The experiments using cyclic reduction were run in single precision (27 bit mantissa). The Chebyshev algorithm was not used with cyclic reduction preconditioning.

The main set of results is shown on pages 142-164. This data is arranged in three groups, corresponding to $\gamma = 5$, 50, and 250. We comment briefly on some of the patterns exhibited in these tests.

Among the iterative methods, the norm-minimizing methods Orthomin(k), GCR(k), and MR exhibit fairly similar behavior, but require fewer operations with small k (0 or 1) than with $k = 5$. The Chebyshev

algorithm usually converges more rapidly (in terms of multiplications) than these methods, but it may diverge initially before good parameters are found. The conjugate gradient method applied to the normal equations is always the least effective iterative method. (See Figures 11-1, 11-2, 11-3, 11-8, 11-9, 11-10, 11-15, and 11-16.)

If enough preprocessing steps are taken, then the hybrid methods prevent the initial divergence of the Chebyshev algorithm, but they do not always cut the total cost of satisfying the stopping criterion. (See Figures 11-6, 11-7, 11-13, 11-14, 11-19, and 11-20.) This is due in part to the higher cost of the GCR iterations. Also, we suspect that the eigenvalue estimates provided by GCR may delay the acquisition of information about the extreme eigenvalues by the Chebyshev algorithm.

All of the preconditioning techniques are improvements over no preconditioning. (See Figures 11-4, 11-11, and 11-17.) Among the preconditionings, the combination of the reduced system with the MILU factorization seems to be the most effective. (See Figures 11-5, 11-12, and 11-18.) The MILU preconditioning seems to be more effective than the ILU preconditioning. The cyclic reduction preconditioning is competitive with the incomplete factorizations only for small γ. (Because of this, we did not include a figure for cyclic reduction preconditioning for $\gamma = 250$, $h = 1/48$; a representative picture of its performance for this problem can be seen in Figure 11-17.) Its difficulty for larger γ is probably due in part to the scaling $\delta(y)$ in

(11.3). Note, though, that the number of iterations required to reach the stopping criterion with this preconditioning does not grow with decreasing h (see Tables 11-2, 11-4, and 11-6), suggesting that it may be useful for very large problems.

Recall that most of the convergence results for the methods of Chapter 5 require that the symmetric part of the coefficient matrix be positive-definite. Although this requirement is satisfied by A, we do not know in general whether it holds for the preconditioned coefficient matrix. Indeed, we have encountered numerous preconditioned problems arising from elliptic partial differential equations in which the symmetric part of the preconditioned matrix is indefinite, including some of those used in the preceding tests. We itemize these below:

- MILU preconditioning: $h = 1/48$, $\gamma = 5$,
  $h = 1/64$, $\gamma = 5, 50, 250$;
- Cyclic reduction preconditioning: $h = 1/16$; $\gamma = 50$,
  $h = 1/16; 1/32, 1/48, 1/64$, $\gamma = 250$.

We have also encountered indefiniteness with the ILU preconditioning in other problems, although it seems less prone to this difficulty than MILU. We feel that this issue merits further study.

In Table 11-7, we examine in more detail the effect of the number of directions k on Orthomin(k). The data suggests that there is little advantage to taking k much larger than 2, and that MR (k = 0) may at

times converge more slowly than Orthomin(k) for $k \geq 1$. GCR(k) behaves in a similar manner.

In Figure 11-21, we show the effect of α on the MILU preconditioning with Orthomin(1), for $h = 1/48$. The value $\alpha = 0$ is approximately optimal ($\alpha = 0.1$ required one fewer iteration for $\gamma = 250$). We do not know how α affects the definiteness of the preconditioned problem.

In Figure 11-22, we briefly examine the SSOR preconditioning with Orthomin(1), for $h = 1/48$. These results suggest that SSOR preconditioning is very sensitive to the value of ω. Indeed, the curves are cut off abruptly at their right endpoints because the preconditioned problems are indefinite for larger values of ω and Orthomin(1) failed to converge.

Finally, in Table 11-8, we compare the bounds for $\frac{\|r_{i+1}\|_2}{\|r_i\|_2}$ derived in the proof of Theorem 5.9 with the maximum values attained during the execution of Orthomin(1). We consider four problems: $h = 1/16$; $\gamma = 5$ and 50, no preconditioning and MILU preconditioning. In the table,

$$B_1 := \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{max}(A^TA)}\right]^{i/2}$$

$$B_2 := \left[1 - \frac{\lambda_{min}(M)^2}{\lambda_{min}(M)\lambda_{max}(M) + \rho(R)^2}\right]^{i/2} .$$

The data suggests that the bounds may not be tight.

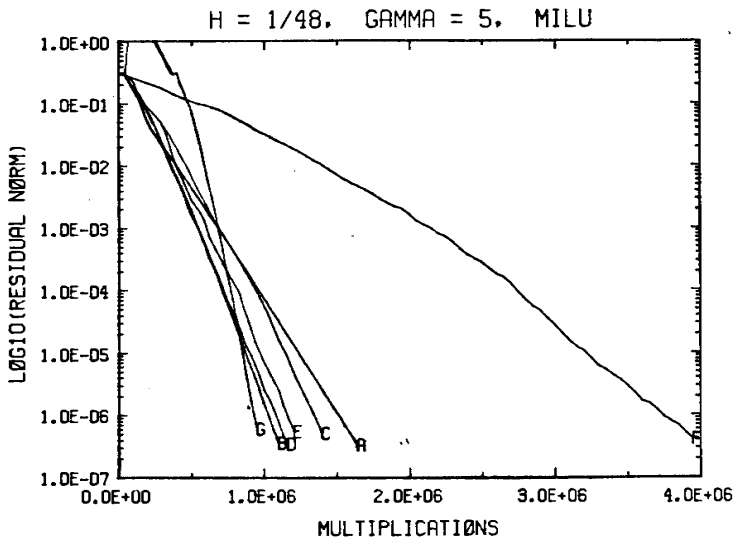


Figure 11-1: Residual norm vs. multiplications for several iterative methods with MILU preconditioning, for γ=5, h=1/48.

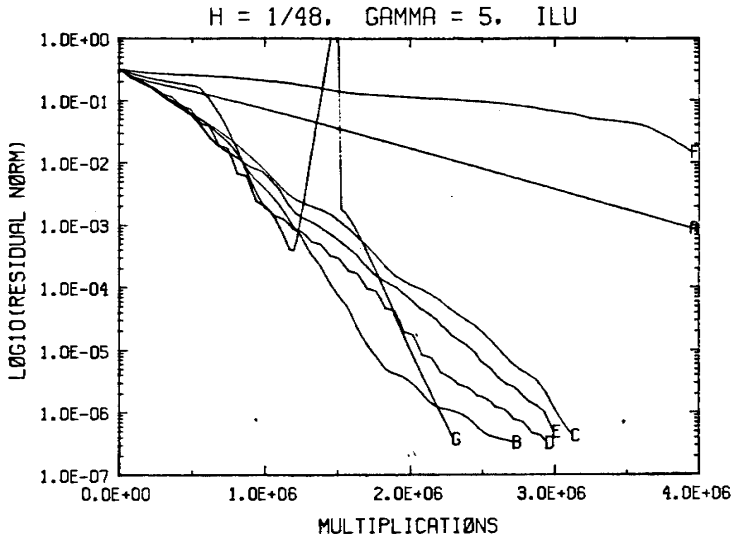


Figure 11-2: Residual norm vs. multiplications for several iterative methods with ILU preconditioning, for γ=5, h=1/48.

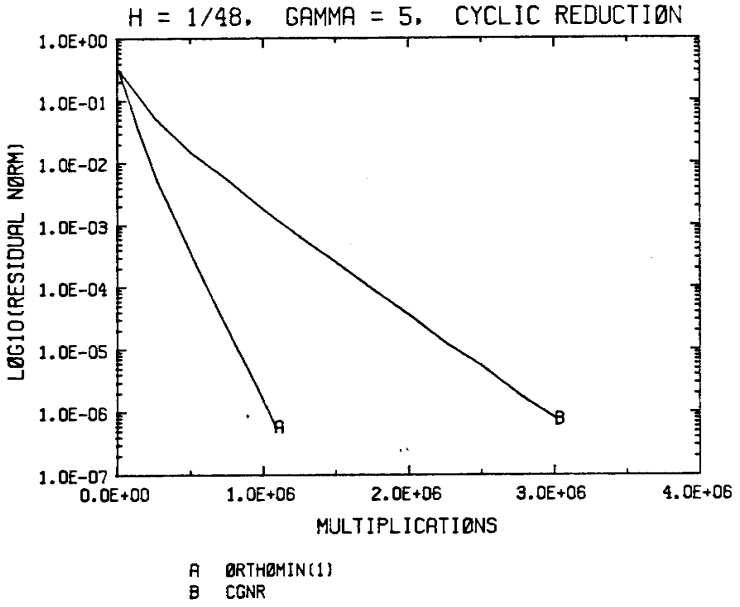


Figure 11-3: Residual norm vs. multiplications for several iterative methods with cyclic reduction preconditioning, for γ=5, h=1/48.

---

*The data for MR, Orthomin(5), GCR(1), and GCR(5) was nearly identical to that for Orthomin(1) in this problem.

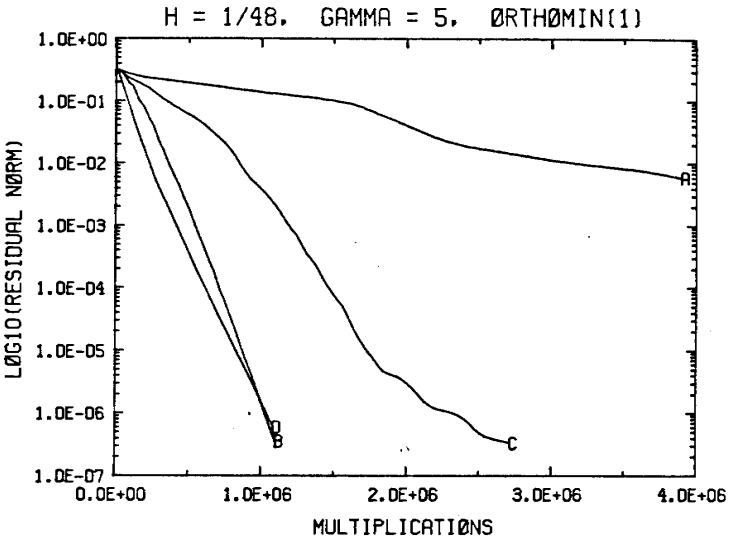


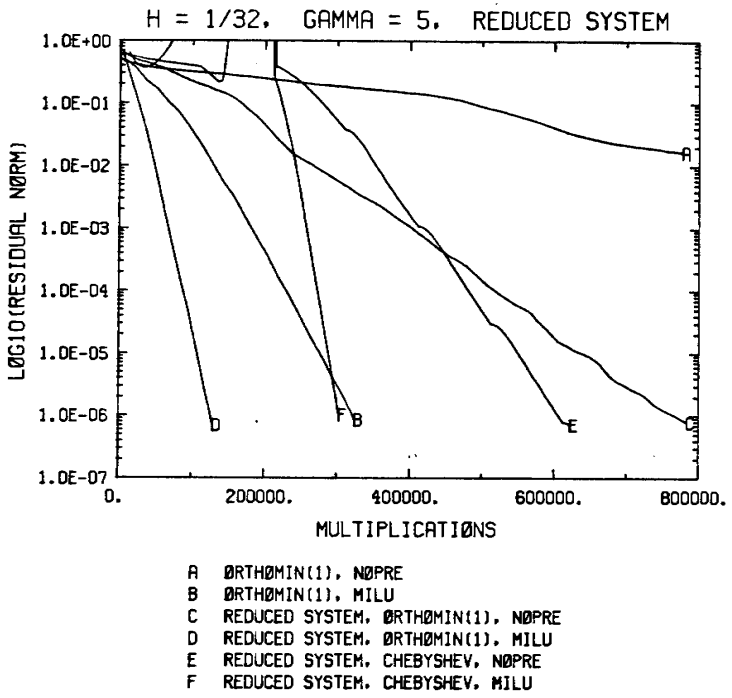
Figure 11-4: Residual norm vs. multiplications for Orthomin(1) with several preconditionings and with no preconditioning, for γ=5, h=1/48.

	MILU	ILU	Cyc.Red.
MR	58	323	11
Orthomin(1)	32	78	9
Orthomin(5)	25	53	9
GCR(1)	37	93	9
GCR(5)	28	67	9
CGNR	80	166	13
Chebyshev	38	94	-

Table 11-1: Number of iterations to satisfy stopping criterion, for γ=5, h=1/48.

	Iterations			Multiplications		
1/h	MILU	ILU	Cyc.Red.	MILU	ILU	Cyc.Red.
16	14	19	8	50397	67957	85725
32	22	50	9	339741	765117	549492
48	32	78	9	1134925	2747869	1220372
64	40	123	9	2548429	7788053	2892032

Table 11-2: Iterations and multiplications used by preconditioned Orthomin(1) to satisfy stopping criterion, for γ=5 and several mesh sizes.



Fignre 11-5: Reduced system preconditioning, and reduced system followed by MILU preconditioning, compared with methods applied to the full linear system, for γ=5, h=1/32.

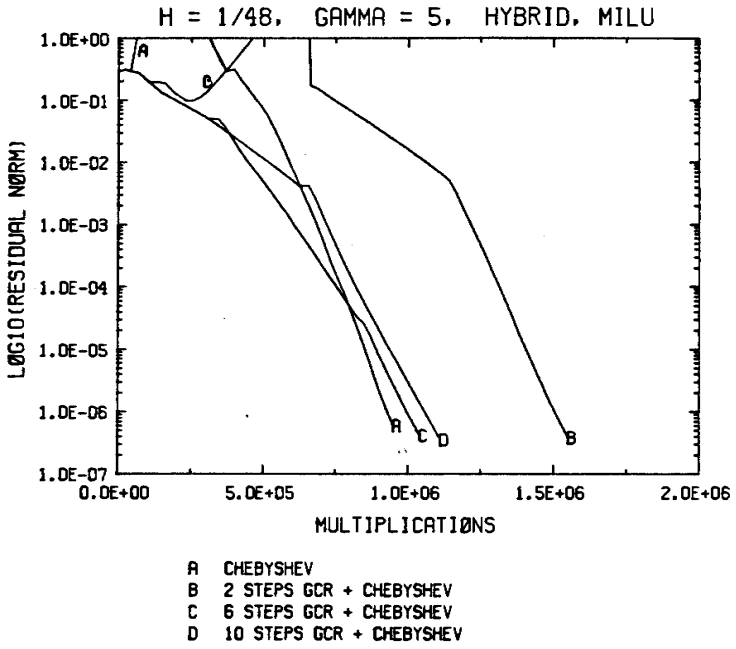


Figure 11-6: Residual norm vs. multiplications for the GCR/Chebyshev hybrid method with MILU preconditioning, for γ=5, h=1/48.

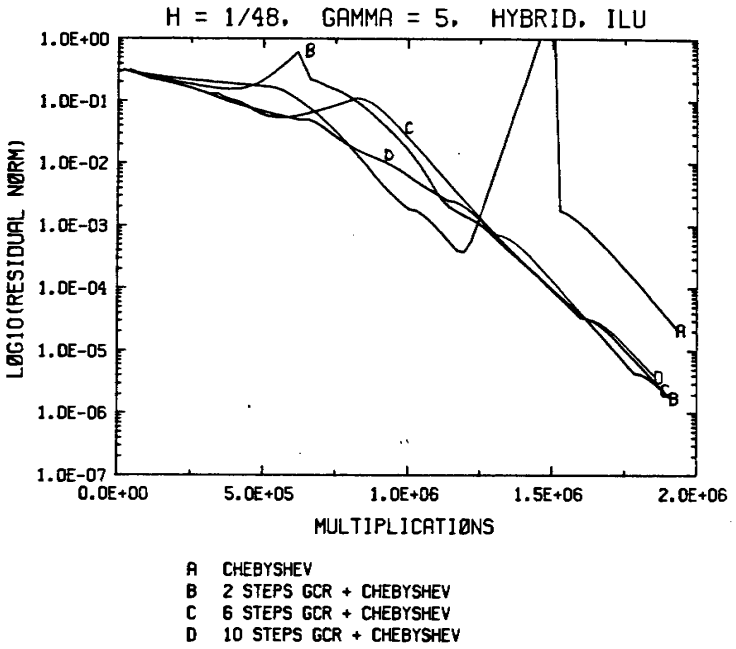


Figure 11-7: Residual norm vs. multiplications for the GCR/Chebyshev hybrid method with ILU preconditioning, for γ=5, h=1/48.

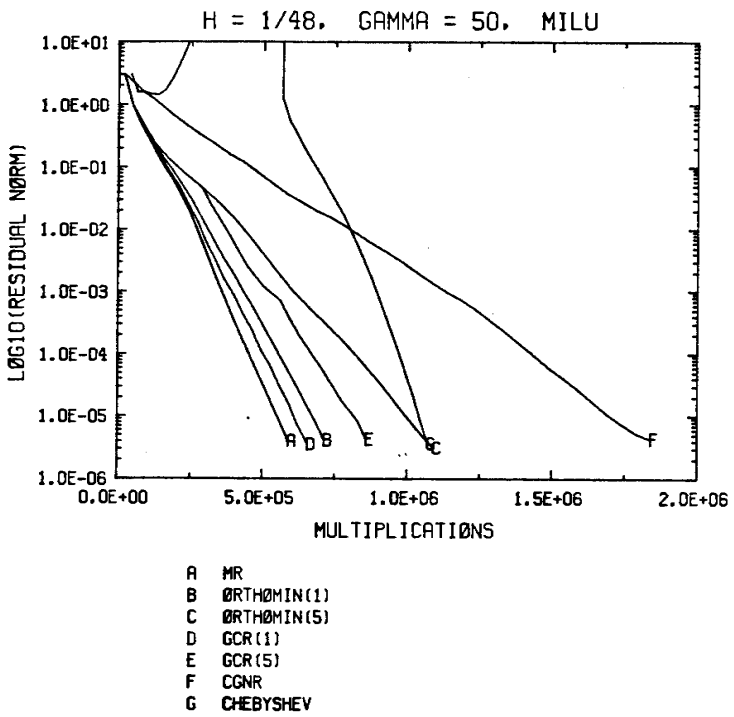


Figure 11-8: Residual norm vs. multiplications for several iterative methods with MILU preconditioning, for γ=50, h=1/48.

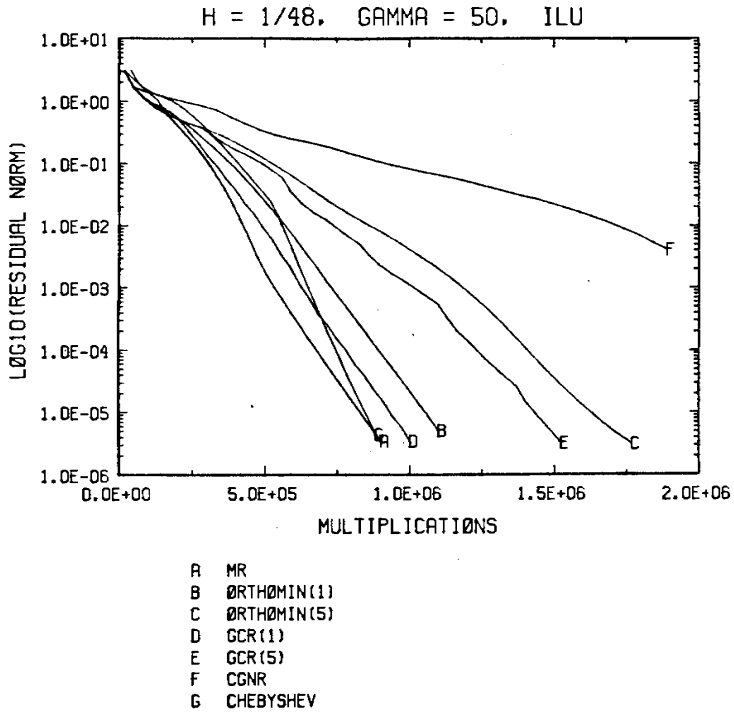


Figure 11-9: Residual norm vs. multiplications for several iterative methods with ILU preconditioning, for γ=50, h=1/48.

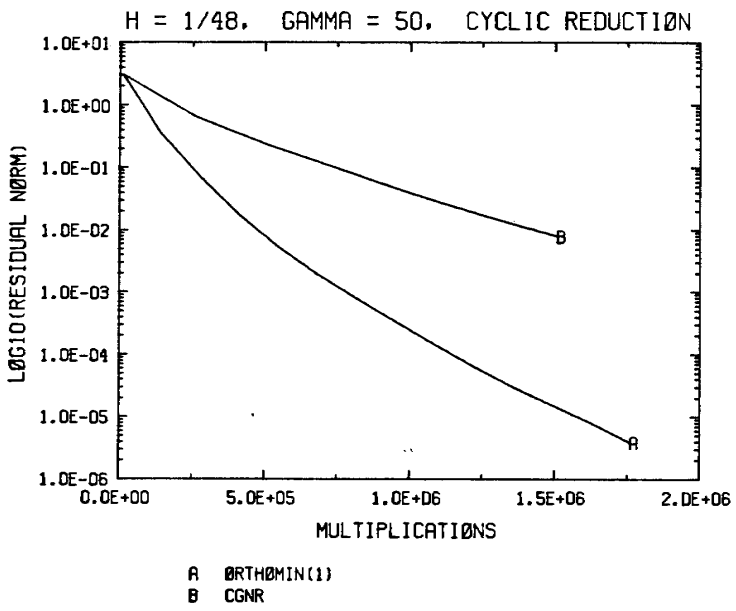


Figure 11-10: Residual norm vs. multiplications for several iterative methods with cyclic reduction preconditioning, for γ=50, h=1/48.

*The data for MR, Orthomin(5), GCR(1), and GCR(5) was nearly identical to that for Orthomin(1) in this problem.

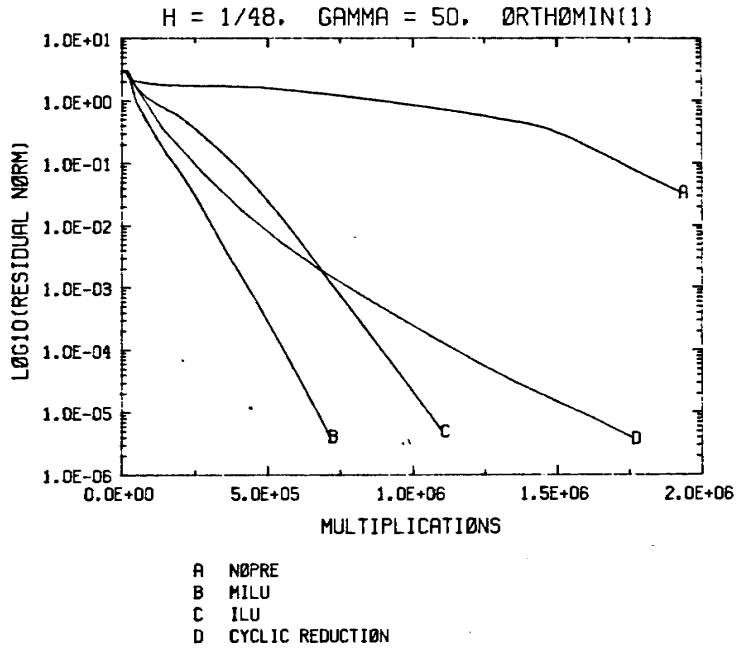


Figure 11-11: Residual norm vs. multiplications for Orthomin(1) with several preconditionings and with no preconditioning, for γ=50, h=1/48.

	MILU	ILU	Cyc.Red.
MR	21	32	14
Orthomin(1)	21	32	14
Orthomin(5)	20	31	12
GCR(1)	21	32	14
GCR(5)	20	35	12
CGNR	37	58	17
Chebyshev	43	36	-

Table 11-3: Number of iterations to satisfy stopping criterion, for γ=50, h=1/48.

	Iterations			Multiplications		
1/h	MILU	ILU	Cyc.Red.	MILU	ILU	Cyc.Red.
16	9	10	23	32837	36349	245730
32	15	19	17	233397	294165	1036332
48	21	32	14	749221	1134925	1896007
64	27	45	14	1727765	2864069	4494447

Table 11-4: Iterations and multiplications used by preconditioned Orthomin(1) to satisfy stopping criterion, for γ=50 and several mesh sizes.

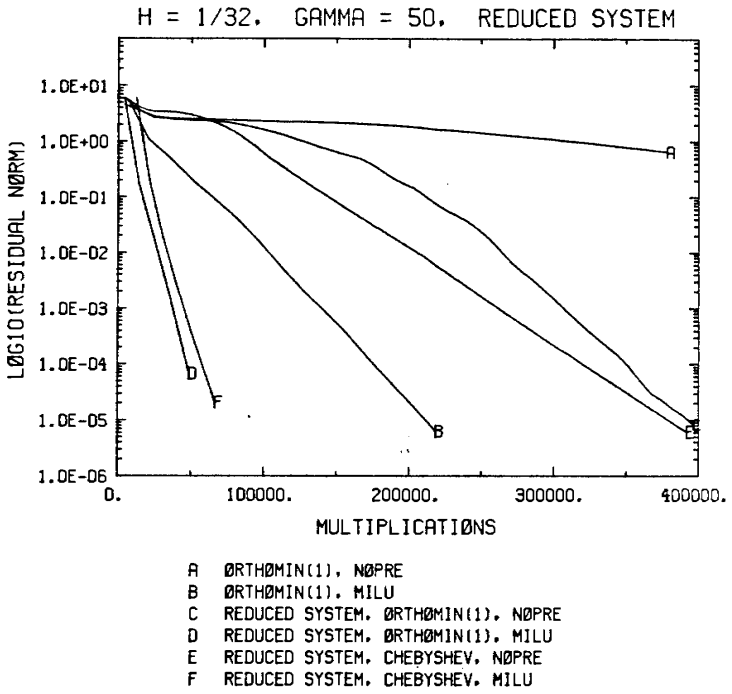


Figure 11-12: Reduced system preconditioning, and reduced system followed by MILU preconditioning, compared with methods applied to the full linear system, for γ=50, h=1/32.

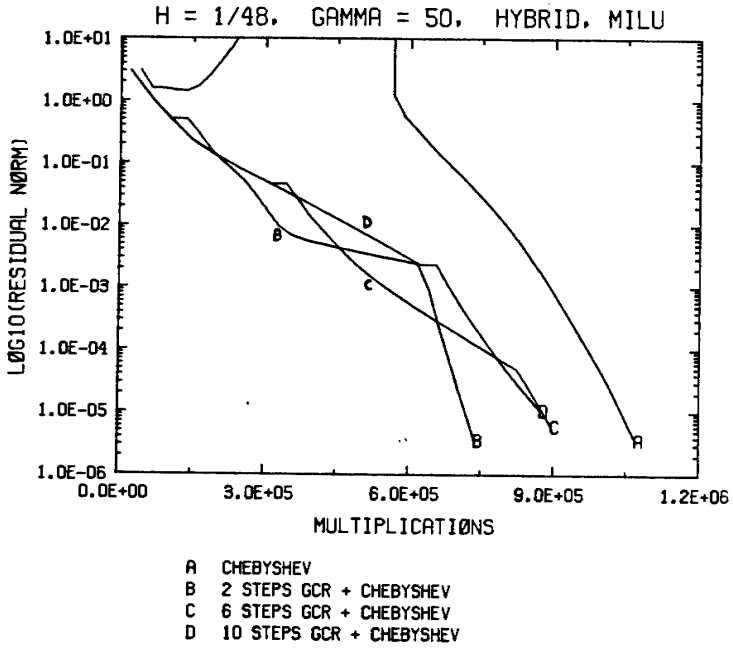


Figure 11-13: Residual norm vs. multiplications for the GCR/Chebyshev hybrid method with MILU preconditioning, for γ=50, h=1/48.

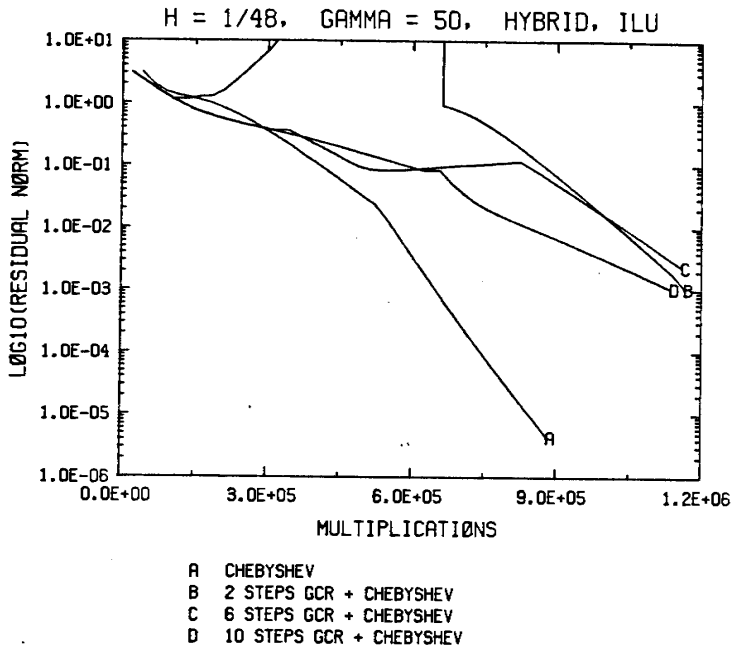


Figure 11-14: Residual norm vs. multiplications for the GCR/Chebyshev hybrid method with ILU preconditioning, for γ=50, h=1/48.

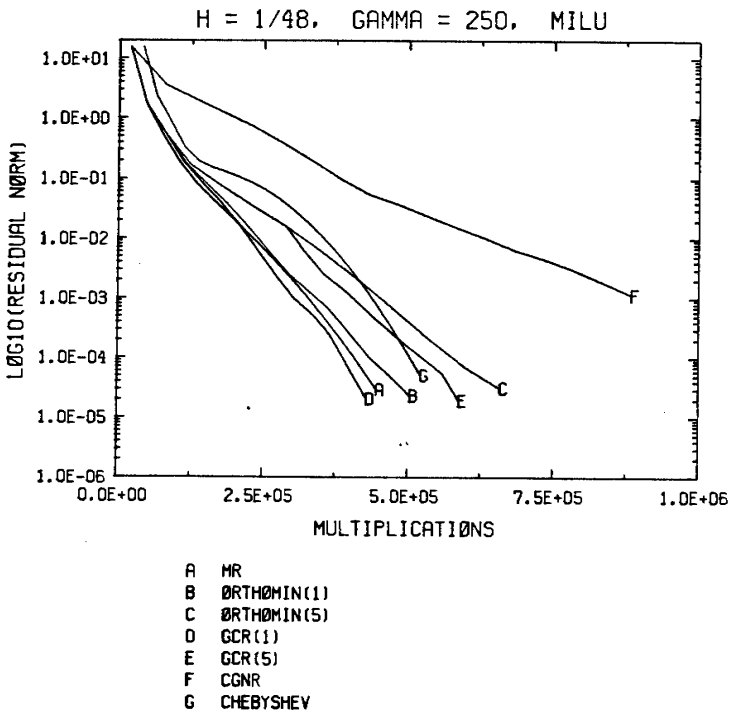


Figure 11-15: Residual norm vs. multiplications for several iterative methods with MILU preconditioning, for γ=250, h=1/48.

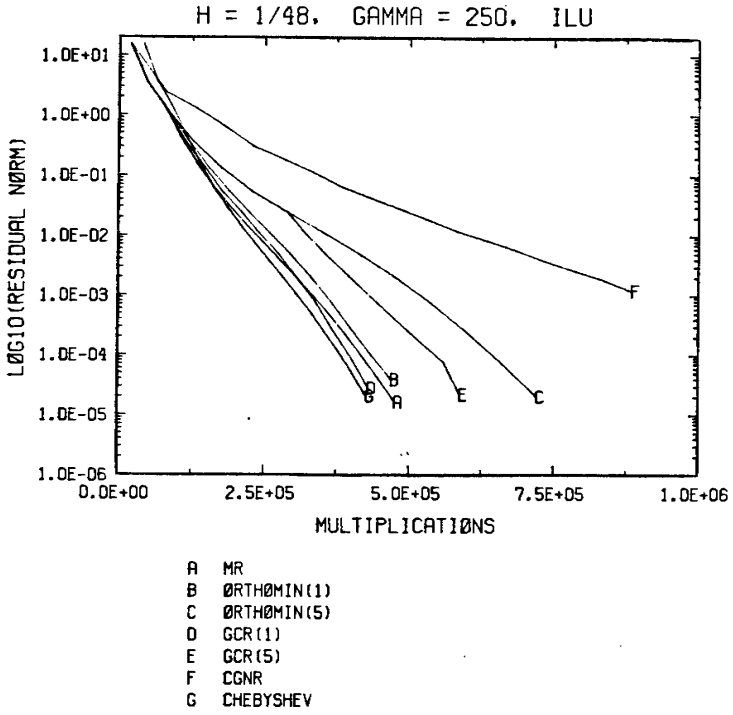


Figure 11-16: Residual norm vs. multiplications for several iterative methods with ILU preconditioning, for γ=250, h=1/48.

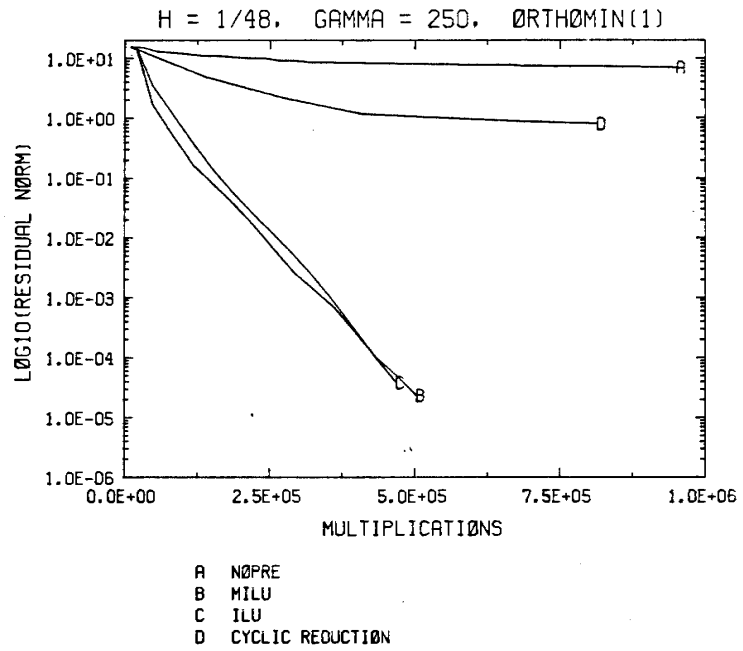


Figure 11-17: Residual norm vs. multiplications for Orthomin(1) with several preconditionings and with no preconditioning, for γ=250, h=1/48.

	MILU	ILU	Cyc.Red.
MR	16	17	39
Orthomin(1)	15	14	39
Orthomin(5)	13	14	38
GCR(1)	14	14	39
GCR(5)	14	14	38
CGNR	26	26	172
Chebyshev	21	17	-

Table 11-5: Number of iterations to satisfy stopping criterion, for γ=250, h=1/48.

	Iterations			Multiplications		
1/h	MILU	ILU	Cyc.Red.	MILU	ILU	Cyc.Red.
16	7	8	94	25813	29325	1003087
32	10	11	53	157437	172629	3227112
48	15	14	39	538837	503773	5274182
64	20	19	30	1285869	1222741	9622175

Table 11-6: Iterations and multiplications used by preconditioned Orthomin(1) to satisfy stopping criterion, for γ=250 and several mesh sizes.

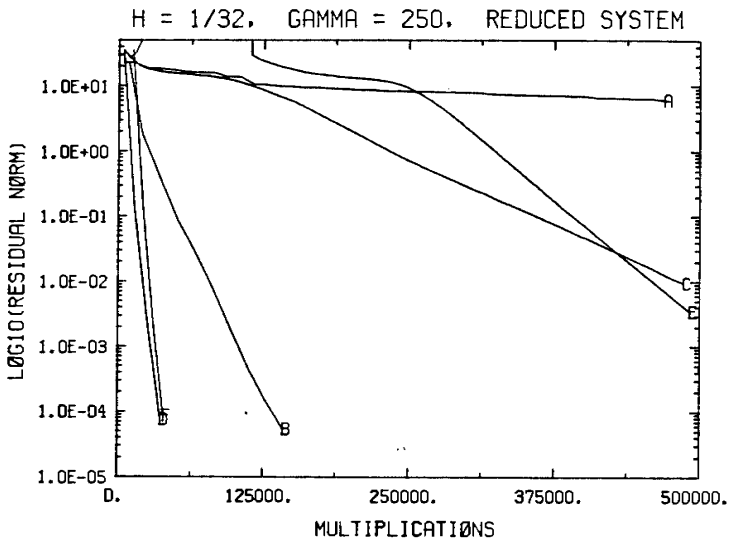


Figure 11-18: Reduced system preconditioning, and reduced system followed by MILU preconditioning, compared with methods applied to the full linear system, for γ=250, h=1/32.

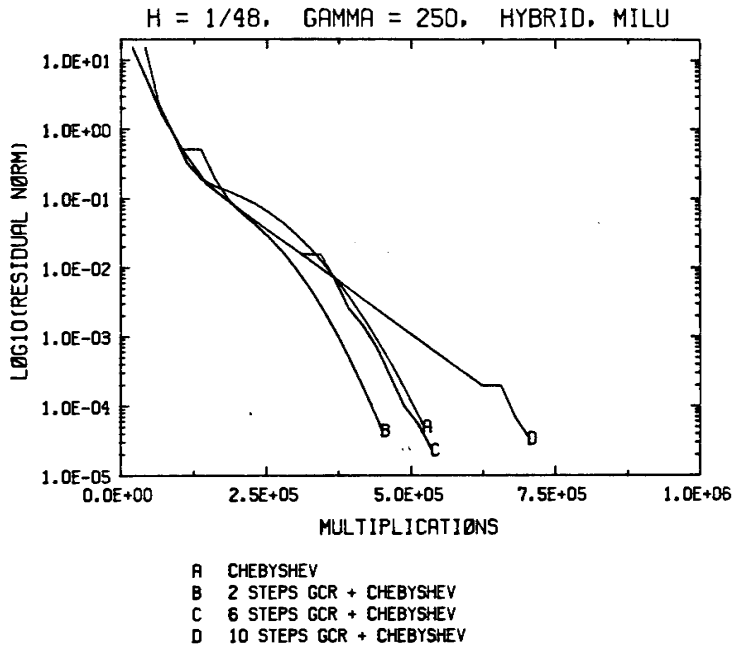


Figure 11-19: Residual norm vs. multiplications for the GCR/Chebyshev hybrid method with MILU preconditioning, for γ=250, h=1/48.

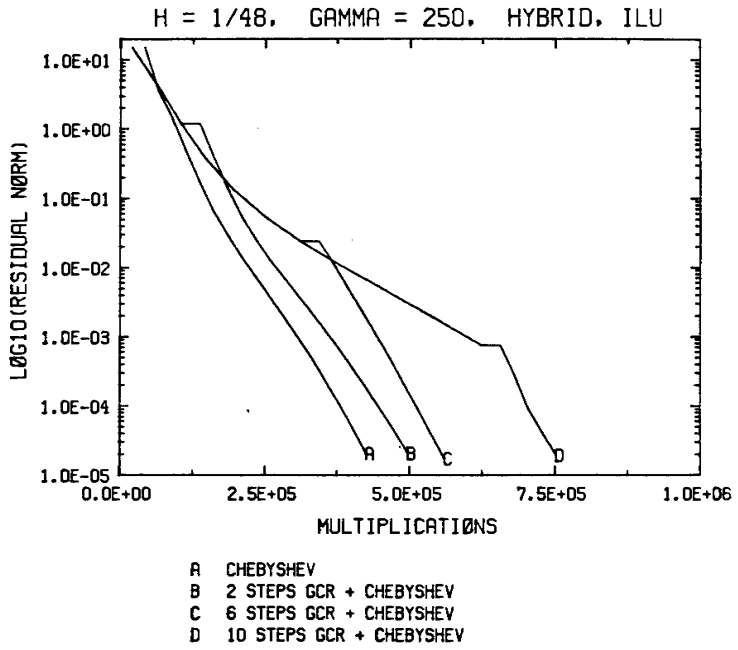


Figure 11-20: Residual norm vs. multiplications for the GCR/Chebyshev hybrid method with ILU preconditioning, for γ=250, h=1/48.

		No preconditioning		MILU preconditioning	
	k	Iterations	Mults	Iterations	Mults
γ = 5	0	>500	>4250000	39	488413
	1	306	3493257	22	339741
	2	156	2225893	21	379345
	3	174	2976349	21	431257
	4	167	3326195	20	456441
	5	143	3242699	20	499701
	6	138	3508657	20	540077
	7	132	3714177	20	577569
	8	125	3850607	20	612177
	9	129	4323079	20	643901
	10	135	4890471	20	672741
	k	Iterations	Mults	Iterations	Mults
γ = 50	0	135	1155691	15	193021
	1	142	1622017	15	233397
	2	108	1539781	14	252813
	3	117	1997203	14	284537
	4	121	2403343	14	313377
	5	120	2714941	14	339333
	6	120	3043717	13	332793
	7	121	3398323	13	350097
	8	127	3913803	13	364517
	9	125	4185151	13	376053
	10	132	4778373	13	384705
	k	Iterations	Mults	Iterations	Mults
γ = 250	0	>500	4250000	11	143789
	1	205	2340847	10	157437
	2	210	2997769	10	180509
	3	213	3646291	10	200697
	4	220	4389481	9	194157
	5	186	4229377	9	205693
	6	194	4955137	9	214345
	7	193	5465731	9	220113
	8	185	5746487	9	222997
	9	189	6391999	9	222997
	10	197	7207163	9	222997

Table 11-7: Iterations and multiplications used by Orthomin(k) to satisfy stopping criterion.

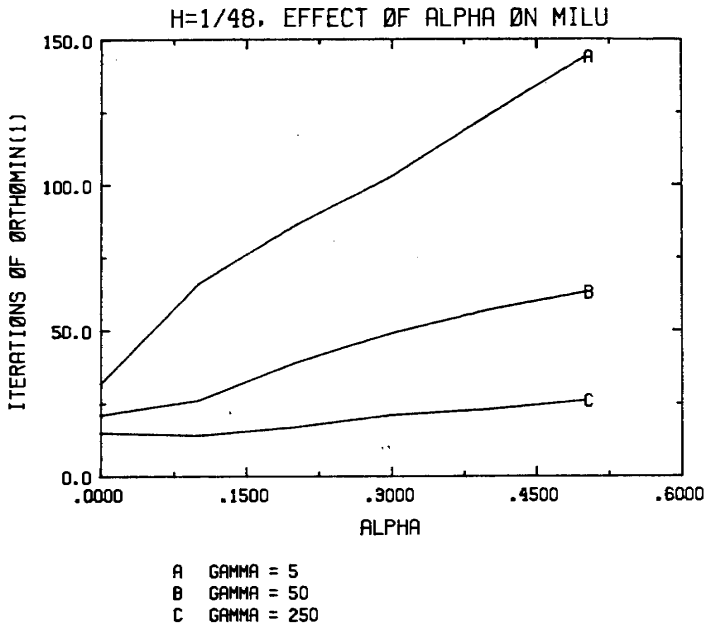


Figure 11-21: The effect of α on MILU preconditioning, for h=1/48.

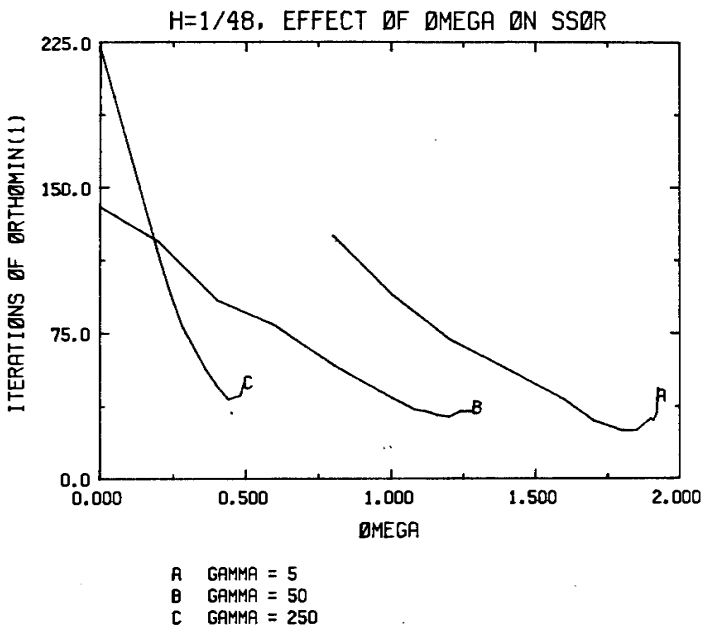


Figure 11-22: The effect of ω on SSOR preconditioning, for h=1/48.

	B_1	B_2	Ratio
$\gamma = 5$, No precon.	.99996	.99807	.93578
$\gamma = 50$, No precon.	.99998	.99997	.98928
$\gamma = 5$, MILU	.97572	.91728	.58727
$\gamma = 50$, MILU	.94071	.88482	.29181

Table 11-8: Upper bounds for $\|r_{i+1}\|_2/\|r_i\|_2$ compared with maximum values obtained during execution of Orthomin(1) for h=1/16.

CHAPTER 12

Conclusions

In this chapter, we review the main ideas of the dissertation and suggest some areas for further research.

We have attempted to describe and add to the development of iterative methods for large, sparse, nonsymmetric linear systems. Much of the effort in this direction has been in generalizations of the conjugate gradient method. Algorithms based on this idea have the advantage of requiring no a priori information about parameters (e.g., eigenvalues) associated with the coefficient matrix. Although the optimality properties of CG appear to be achievable for nonsymmetric problems only at large expense, several algorithms have been proposed that are less than optimal but inexpensive and convergent.

An alternative approach is to acquire information about the coefficient matrix during the iteration, and to use this information to modify the iteration. This idea is the basis for both the adaptive Chebyshev method and Broyden's method. Of these two techniques, only the Chebyshev method has low cost per step.

	Method	Convergence Domain	Expense
	CGNR, CGNE	General	Fixed, A^TA
Minimizing	GCR	M p.d.	Increase
	Orthomin(k) $k \geq 0$	M p.d.	Fixed
	LSGCR	M p.d.	Increase
	Axel(k)	M p.d.	Fixed
	Orthodir	General	Increase
	Orthodir(k) $k \geq 2$	M = I	Fixed
Galerkin-Lanczos	GCG	M = I	Fixed
	Orthores	M p.d.	Increase
	Orthores(k) $k \geq 2$	M=I, unknown for more general A	Fixed
	FOM, DFOM	M p.d.	Increase
	IOM(k), DIOM(k) $k \geq 2$	M=I, unknown for more general A	Fixed
Others	Chebyshev	M p.d.	Fixed
	Broyden	General	Increase
	Hybrid	M p.d.	Fixed

Table 12-1: Summary of iterative methods and their properties.

In Table 12-1, we list the methods that we have considered, their domains of convergence, and some aspects of their work and storage costs. The CG-like methods are grouped according to the properties of

CG that they generalize. The convergence domains consist of the largest classes of nonsymmetric matrices A for which the methods are known to converge. They are identified as general nonsingular matrices (General), matrices with positive-definite symmetric part (M p.d.), and matrices whose symmetric part is the identity matrix (M = I). For the work/storage properties, we specify whether these expenses per iteration are fixed or increasing with the number of iterations. In addition, we identify those methods that generate a Krylov sequence based on A^TA rather than A. Note that methods with increasing costs, such as GCR, can have their costs fixed through restarting.

We elaborate on the convergence domains. For the norm-minimizing methods, we have constructed counter-examples in the next larger domain for which the methods do not converge. For example, Orthomin(k) is not guaranteed to converge for general nonsymmetric matrices. We are less certain about Orthores(k), IOM(k), and DIOM(k). We know of no convergence results for these methods that are applicable to problems with positive-definite symmetric part, but we do not have examples of such problems for which the methods do not converge. We indicate this uncertainty in the table. (GCG was not designed for more general problems.) Finally, recall that while the Chebyshev polynomials are of use if the eigenvalues of A lie in the right half plane, the adaptive Chebyshev algorithm requires that the symmetric part be positive-definite.

We have performed numerical experiments with a subset of these methods, using several preconditioning techniques. In general, we have found the Chebyshev method to be more rapidly convergent than the CG-like methods (Orthomin(k) and GCR(k)) tested, but the former method is sensitive to iteration parameters. All of these methods appear to be more effective than the conjugate gradient method applied to the normal equations. Preconditionings based on incomplete factorizations, fast direct methods, and reduced systems all speed the convergence of the iterative methods.

Finally, several important issues concerning these methods are unresolved. We feel the most important ones are the following:

1. Error bounds: The iterative methods seem to converge more rapidly than the current error bounds suggest; we do not know if these bounds are tight or if there are stronger bounds.

2. Stopping criteria: We have used the norm of the residual for the stopping criterion because it is easy to compute. However, the error $x-x_i$ might be large even if the residual is small. For symmetric, positive-definite problems, an upper bound for the norm of the error can be obtained fairly easily [38]. We know of no simple way to do this for nonsymmetric problems.

3. Indefinite systems: Problems in which the symmetric part is indefinite occur in applications, and (as we showed in Chapter 11) they may result from preconditioning. Most of the methods considered are not rigorously applicable to indefinite problems.

4. Vector machines: Since the iterative methods are based on inner products and scalar–vector products, they are well-suited to vector machines. However, the forward- and back-solves required by the incomplete factorization preconditionings cannot be implemented as efficiently on vector machines. Other techniques need to be examined. (See [72] for a list of references to work in this area.)

REFERENCES

[1] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. Quart. Appl. Math. 9:17-29, 1951.

[2] Owe Axelsson. Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations. Linear Algebra and Its Applications 29:1-16, 1980.

[3] Owe Axelsson. A generalized SSOR method. BIT 13:443-467, 1972.

[4] Owe Axelsson. Solution of linear systems of equations: iterative methods. In V. A. Barker, Editor, Sparse Matrix Techniques, Springer-Verlag, New York, 1976, pp. 1-51.

[5] Owe Axelsson and I. Gustafsson. A modified upwind scheme for convective transport equations and the use of a conjugate gradient method for the solution of non-symmetric systems of equations. Journal of the Institute of Mathematics and its Applications 23:321-337, 1979.

[6] Randolph E. Bank. An automatic scaling procedure for a D'Yakanov-Gunn iteration scheme. Technical Report CNA-142, Center for Numerical Analysis, University of Texas at Austin, August 1978.

[7] Randolph E. Bank. Marching algorithms for elliptic boundary value problems. II: The variable coefficient case. SIAM Journal on Numerical Analysis 14:950-970, 1977.

[8] Randolph E. Bank and Donald J. Rose. Marching algorithms for elliptic boundary value problems. I: The constant coefficient case. SIAM Journal on Numerical Analysis 14:792-829, 1977.

[9] Ake Bjorck and Tommy Elfving. Accelerated Projection Methods for Computing Pseudoinverse Solutions of Systems of Linear Equations. BIT 19:145-163, 1979.

[10] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. Mathematics of Computation 19:577-593, 1965.

[11] Tony F. Chan, Kenneth R. Jackson, and Benren Zhu. Alternating Direction Incomplete Factorizations. Technical Report 208, Yale University Department of Computer Science, August 1981.

[12] Rati Chandra. Conjugate Gradient Methods for Partial Differential Equations. Ph.D. Thesis, Department of Computer Science, Yale University, 1978. Also available as Research Report #129.

[13] Paul Concus and Gene H. Golub. A generalized conjugate gradient method for nonsymmetric systems of linear equations. In R. Glowinski and J. L. Lions, Editors, Lecture Notes in Economics and Mathematical Systems, Volume 134, Springer-Verlag, Berlin, 1976, pp. 56-65.

[14] Paul Concus and Gene H. Golub. Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations. SIAM Journal on Numerical Analysis 10:1103-1120, 1973.

[15] Paul Concus, Gene H. Golub, and Dianne P. O'Leary. A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. In James R. Bunch and Donald J. Rose, Editors, Sparse Matrix Computations, Academic Press, New York, 1976, pp. 309-332.

[16] James W. Daniel. The Approximate Minimization of Functionals. Prentice-Hall, New York, 1971.

[17] Philip J. Davis. Interpolation and Approximation. Dover Publications, Inc., New York, 1975. Originally published by Blaisdell Publishing Co., 1963.

[18] Fred W. Dorr. The direct solution of the discrete Poisson equation on a rectangle. SIAM Review 12:248-263, 1970.

[19] Todd Dupont, Richard P. Kendall and H. H. Rachford Jr. An approximate factorization procedure for solving self-adjoint elliptic difference equations. SIAM Journal on Numerical Analysis 5:559-573, 1968.

[20] Stanley C. Eisenstat. Efficient implementation of a class of conjugate gradient methods. SIAM Journal on Scientific and Statistical Computing 2:1-4, 1981.

[21] Stanley C. Eisenstat. A note on the generalized conjugate gradient method. Technical Report 228, Yale University Department of Computer Science, April 1982.

[22] Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. Variational Iterative Methods for Nonsymmetric Systems of Linear Equations. Technical Report 209, Yale University Department of Computer Science, August 1981.

[23] S. C. Eisenstat, H. Elman, M. H. Schultz, and A. H. Sherman. Solving approximations to the convection diffusion equation. In Society of Petroleum Engineers of AIME, Proceedings of the Fifth Symposium on Reservoir Simulation, 1979, pp. 127-132.

[24] Howard C. Elman. Preconditioned conjugate gradient methods for nonsymmetric systems of linear equations. In R. Vichnevetsky and R. S. Stepleman, Editors, Advances in Computer Methods for Partial Differential Equations - IV, IMACS, 1981, pp. 409-417.

[25] M. Engeli, T. Ginsberg, H. Rutishauser, and E. Stiefel. Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems. Birkhauser Verlag, Basel, Stuttgart, 1959.

[26] M. S. Engleman, G. Strang, and K.-J. Bathe. The application of quasi-Newton methods in fluid mechanics. International Journal for Numerical Methods in Engineering 17:707-718, 1981.

[27] D. K. Fadeev and V. N. Fadeeva. Computational Methods of Linear Algebra. Freeman and Company, London, 1963.

[28] R. Fletcher. Conjugate Gradient Methods for Indefinite Systems. In G. A. Watson, Editor, Numerical Analysis Dundee 1975, Springer-Verlag, New York, 1976, pp. 73-89.

[29] George E. Forsythe and Wolfgang R. Wasow. Finite-Difference Methods for Partial Differential Equations. John Wiley and Sons, New York, 1960.

[30] L. Fox and I. B. Parker. Chebyshev Polynomials in Numerical Analysis. Oxford University Press, London, 1968.

[31] David M. Gay. Some convergence properties of Broyden's method. SIAM Journal on Numerical Analysis 16:623-630, 1979.

[32] Alan George. Nested dissection of a regular finite element mesh. SIAM Journal on Numerical Analysis 10:345-363, 1973.

[33] Richard R. Gerber and Franklin T. Luk. A generalized Broyden's method for solving simultaneous linear equations. SIAM Journal on Numerical Analysis 18:882-890, 1981.

[34] Gene H. Golub and Michael L. Overton. Convergence of a two-stage Richardson iterative procedure for solving systems of linear equations. Technical Report 38, Computer Science Department, Stanford University, September 1981.

[35] Ivar Gustafsson. A class of first order factorizations. BIT 18:142-156, 1978.

[36] Ivar Gustafsson. Stability and Rate of Convergence of Modified Incomplete Cholesky Factorization Methods. Ph.D. Thesis, Department of Computer Sciences, Chalmers University of Technology and the University of Goteborg, 1978. Also available as Research Report #77.04R.

[37] L. A. Hageman, Franklin T. Luk, and David M. Young. On the equivalence of certain iterative acceleration methods. SIAM Journal on Numerical Analysis 17:852-873, 1980.

[38] Louis A. Hageman and David M. Young. Applied Iterative Methods. Academic Press, New York, 1981.

[39] Magnus R. Hestenes. The conjugate-gradient method for solving linear systems. In American Mathematical Society, New York, Numerical Analysis, 1956, pp. 83-102. Proceedings of the Symposium in Applied Mathematics, Vol. 6.

[40] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards 49:409-435, 1952.

[41] Alston S. Householder. The Theory of Matrices in Numerical Analysis. Dover Publications, Inc., New York, 1975. Originally published by Blaisdell Publishing Co., 1964.

[42] D. A. H. Jacobs. Generalizations of the conjugate gradient method for solving nonsymmetric and complex systems of algebraic equations. Technical Report RD/L/N 70/80, Central Electricity Research Laboratories, Surrey, England, 1980.

[43] Kang Chang Jea. Generalized Conjugate Gradient Acceleration of Iterative Methods. Ph.D. Thesis, University of Texas at Austin, 1982. Also available as CNA Research Report #CNA-176.

[44] David S. Kershaw. The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations. Journal of Computational Physics 26:43-65, 1978.

[45] David G. Luenberger. Optimization by Vector Space Methods. John Wiley and Sons, New York, 1969.

[46] Thomas A. Manteuffel. Adaptive procedure for estimation of prameters for the nonsymmetric Tchebychev iteration. Numerische Mathematik 31:187-208, 1978.

[47] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. Mathematics of Computation 34:473-497, 1980.

[48] Thomas Albert Manteuffel. An iterative method for solving nonsymmetric linear systems with dynamic estimation of parameters. Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1975.

[49] Thomas A. Manteuffel. The Tchebychev iteration for nonsymmetric linear systems. Numerische Mathematik 28:307-327, 1977.

[50] J. A. Meijerink and H. A. van der Vorst. Guide lines for the usage of incomplete decompositions in solving sets of linear equations as occur in practical problems. Technical Report 550, Koninklijke/Shell Exploratie en Produktie Laboratorium, December 1980. To appear in Journal of Computational Physics.

[51] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Mathematics of Computation 31:148-162, 1977.

[52] Dianne Prost O'Leary. Hybrid Conjugate Gradient Algorithms. Ph.D. Thesis, Department of Computer Science, Stanford University, 1976.

[53] Dianne P. O'Leary. The block conjugate gradient algorithm and related methods. Linear Algebra and Its Applications 29:293-322, 1980.

[54] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. SIAM Journal on Numerical Analysis 12:617-629, 1975.

[55] M. Petravic and G. Kuo-Petravic. An ILUCG algorithm which minimizes in the Euclidean norm. Journal of Computational Physics 32:263-269, 1979.

[56] J. K. Reid. On the method of conjugate gradients for the solution of large sparse systems of linear equations. In J. K. Reid, Editor, Large Sparse Sets of Linear Equations, Academic Press, New York, 1971, pp. 231-254.

[57] J. K. Reid. The use of conjugate gradients for systems of linear equations possessing "property A". SIAM Journal on Numerical Analysis 9:325-332, 1972.

[58] Donald J. Rose. A graph theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. Read, Editor, Graph Theory and Computing, Academic Press, New York, 1973, pp. 183-217.

[59] D. J. Rose and G. F. Whitten. Automatic Nested Dissection. In Association for Computing Machinery, New York, Proceedings of the ACM Annual Converence, 1974, pp. 82-88.

[60] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. Mathematics of Computation 37:105-126, 1981.

[61] Y. Saad. The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems. Technical Report UIUCDCS-R-1036; University of Illinois at Urbana Champaign, 1980. To appear in Siam Journal on Numerical Analysis.

[62] Y Saad. Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems. Technical Report 214, Yale University Department of Computer Science, January 1982.

[63] Y. Saad. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. Linear Algebra and its Applications 34:269-295, 1980.

[64] Paul E. Saylor. Use of the singular value decomposition with the Manteuffel algorithm for nonsymmetric linear systems. Siam Journal on Scientific and Statistical Computing 1:210-222, 1980.

[65] U. Schumann, Editor. Fast Elliptic Solvers. Advance Publications, United Kingdom, 1978. Proceedings of the GAMM-Workshop on Fast Solution Methods for the Discretized Poisson Equation, Karlsruhe, 1977.

[66] Andrew Harry Sherman. On the Efficient Solution of Sparse Systems of Linear and Nonlinear Equations. Ph.D. Thesis, Department of Computer Science, Yale University, 1975.

[67] Eduard L. Stiefel. Relaxationsmethoden bester strategie zur losung linearer gleichungssystems. Comment. Math. Helv. 29:157-179, 1955.

[68] Herbert L. Stone. Iterative solution of implicit approximations of multidimensional partial differential equatins. SIAM Journal on Numerical Analysis 5:530-558, 1968.

[69] Paul N. Swarztrauber. A direct method for the discrete solution of separable elliptic equations. SIAM Journal on Numerical Analysis 11:1136-1150, 1974.

[70] Paul N. Swarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. SIAM Review 19:490-501, 1977.

[71] P. Swarztrauber and R. Sweet. Efficient FORTRAN subprograms for the solution of elliptic partial differential equations. Technical Report TN/IA-109, National Center for Atmospheric Research, 1975.

[72] Henk A. van der Vorst. A vectorizable variant of some ICCG methods. 1981. Unpublished manuscript, Academisch Computer Centrum Utrecht, the Netherlands.

[73] Richard S. Varga. Matrix Iterative Analysis. Prentice-Hall, New York, 1962.

[74] P. K. W. Vinsome. Orthomin, an iterative method for solving sparse sets of simultaneous linear equations. In Society of Petroleum Engineers of AIME, Proceedings of the Fourth Symposium on Reservoir Simulation, 1976, pp. 149-159.

[75] Olof Widlund. A Lanczos method for a class of non-symmetric systems of linear equations. SIAM Journal on Numerical Analysis 15:801-812, 1978.

[76] J. H. Wilkinson. The Algebraic Eigenvalue Problem. Oxford University Press, London, 1965.

[77] Yau Shu Wong. Conjugate gradient type methods for unsymmetric matrix problems. Technical Report 79-36, Department of Computer Science, University of British Columbia, 1979.

[78] David M. Young. Iterative Solution of Large Linear Systems. Academic Press, New York, 1971.

[79] David M. Young, Linda J. Hayes, and Kang C. Jea. Generalized Conjugate Gradient Acceleration of Iterative Methods. Part I: The Symmetrizable Case. Technical Report CNA-162, Center for Numerical Analysis, The University of Texas at Austin, 1981.

[80] David M. Young and Kang C. Jea. Generalized Conjugate Gradient Acceleration of Iterative Methods. Part II: The Nonsymmetrizable Case. Technical Report CNA-163, Center for Numerical Analysis, The University of Texas at Austin, 1981.

[81] David M. Young and Kang C. Jea. Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods. Linear Algebra and Its Applications 34:159-194, 1980.