ROW PROJECTION METHODS FOR LINEAR SYSTEMS

Randall Barry Bramley
B.S., University of Texas at Arlington, 1981
M.S., University of Illinois at Urbana-Champaign, 1984

This thesis would not have been possible without the generous help of Ahmed Sameh, my graduate advisor. The many hours of conversation and consultation he provided are beyond value, and his careful editing has spared the reader some difficult reading. I also thank the rest of the thesis committee, W. Gear, R. Skeel, T. Kerkhoven, and S. Gallopoulos for the time and help they have given.

The Center for Supercomputing Research and Development has provided me with financial support and more importantly, an access to both hardware and people that is unmatched at any university. In particular, Mike Berry, Sy-Shin Lo, Ulrike Meier, and Youcef Saad have generously given me valuable help and information in many conversations, and Jeannie Covert, Deb Hudson, and Mark Washburn have conscientiously kept the necessary equipment running. Many past and present students have also helped me. Ed Anderson optimized for the Alliant FX/8 the PCGPAK library used for some of the test results, Hsin-Chu Chen taught me the ropes around CSRD and much more, Ding-Kai Chen has been a valuable source of answers to machine architecture questions, Dave Semararo has provided useful conversations that sometimes involved CFD, and Tom Conte developed and answered questions about the Latex macros used for writing this thesis. I especially thank Mei-Qin Chen and Xiaoge Wang for helpful mathematical discussions and for being willing to listen to long pointless stories.

Over the years the faculty in the Mathematics Department at the University of Illinois has also given both professional and personal support. In particular I thank S.-P. Han, H. Lotz, A. Peressini, B. Reznick, and J. Uhl. W. Zaring deserves special mention for the good advice and willing ear he has provided not only to myself but also to several hundred graduate students.

I also thank the people who gave me a shove in the right direction at the right time. This includes Sid Strachan and the other members of the International Association of Machinists and Aerospace Workers' Lodge 1591 who encouraged me to apply for the IAM scholarship without which I would never have completed an undergraduate degree, Jacob Beard who steered me into mathematics, Karen Cunningham and Larry F. Heath who persuaded me to pursue graduate studies, and again Ahmed Sameh who induced me to transfer to the Computer Science Department and has since provided the many nudges needed to keep me on course.

The research this thesis is based on has been supported in part by the following grants: NSF MIP-8410110, DOE DE-FG02-85ER25001, AT&T-AFFL-67-SAMEH, NSF CCR8717492, and AFOSR-85-0211. Additionally, the Digital Equipment Corporation has provided funds from the CSRD Affiliates Program, and the National Science Foundation from a graduate fellowship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Solving linear systems of equations is the most fundamental problem in scientific, engineering, and statistical computing. Virtually all such computational problems involve creating one or a sequence of linear systems which are then to be solved, possibly in a least squares sense, on a computer. For this reason over the last forty years much research effort has been devoted to the development of effective algorithms for this problem. The increasing size of linear systems has in particular generated a need for iterative methods, which produce a sequence of increasingly accurate estimates to a solution and which require relatively small amounts of computer memory.

Highly reliable and efficient iterative algorithms have been developed for some of the more tractable linear problems, such as those involving symmetric positive definite coefficient matrices. For nonsymmetric or indefinite problems, however, existing algorithms often require computationally expensive parameter estimation which sometimes requires user intervention. Furthermore they generally require special properties to hold which may not be intrinsic to the problems and are often difficult to verify in practice. These algorithms can frequently fail, even on otherwise reasonable problems. In the language of scientific computing they lack robustness.

This thesis examines a class of algorithms called row projection (RP) methods. RP methods require no parameter estimation, allowing them to be used as 'black-box' solvers by nonspecialists. For a wide class of important problems they can be implemented efficiently on computers, especially modern vector and multiprocessor machines. Most importantly, they are robust and can successfully solve problems on which other current iterative solvers fail.

1.1 Statement of the Problem

Let $A \in \Re^{M \times N}$, the set of M by N real matrices, and $b \in \Re^{M}$, the set of M by 1 column vectors. The fundamental problem considered in this thesis is to find an $x \in \Re^{N}$ such that

(i) if $b \in \text{range}(A)$, then $Ax^* = b$

(ii) if b is not in range(A), then $x^* = Gb$ where G satisfies $AGA = A$ and some or all of the remaining Moore-Penrose conditions [3]:

$$\begin{aligned} GAG &= G \\ (AG)^T &= AG \\ (GA)^T &= GA \end{aligned} \tag{1.1}$$

The matrix G is called a generalized inverse of A, and if G also satisfies all of the conditions in (1.1) it is called the (unique) pseudo-inverse of A and denoted as $G = A^{\dagger}$. When $b \in \text{range}(A)$, the system $Ax = b$ is said to be consistent, and inconsistent otherwise. In the consistent case if more than one x satisfies $Ax = b$ there is a unique $\bar{x}$ that has minimal two-norm $\| \bar{x} \|_2$ over all such solutions. This $\bar{x}$ is called the minimum norm solution to $Ax = b$. Finally, when A is nonsingular the inverse is denoted A^{-1} and $x^* = A^{-1}b$ is sought. The particular algorithms implemented and tested are for the case when A is nonsingular, but Chapter 2 shows that RP methods can be applied when the system is inconsistent or even if A is rectangular.

1.2 Sources of Linear Problems

Linear programming, optimization, control theory, eigenvalue computations, and tomography are among the many important applications that require the solution of linear systems. In fact, linear systems are so common in scientific and engineering computations that listing all of their sources is impossible. The explanation for this ubiquity is simple; unlike nonlinear problems, linear systems can be solved immediately and accurately on a computer. For this reason, although many physical processes are inherently continuous and nonlinear, numerical computations dealing with those processes usually involve some form of discretization in order to create a related finite dimensional problem, and linearization in order to reduce the problem to one of solving a linear system of equations.

An important class in this category is the numerical solution of partial differential equations arising in fields as diverse as fluid dynamics, structural mechanics, neutron transport, elasticity, and weather prediction. For a description of how the linear systems are created from partial differential equations, see [54, 5]. Such linear systems have the important features of being large scale ($N > 10^5$), especially when three dimensional problems are being solved, often are nonsymmetric, i.e., $A \neq A^T$, and are sparse because the number of zero entries is much greater than the number of nonzeros. The test problems in this thesis arise from the class of non-selfadjoint linear elliptic partial differential equations because this is an increasingly important problem in numerical analysis, the systems have until now been the most challenging for iterative methods, and they allow an instructive analysis of implementation details. As the results in Chapter 2 show, however, RP methods can be applied to a much wider range of linear systems.

1.3 Direct and Iterative Solvers

Solution methods for $Ax = b$ can be classified as direct or iterative. Direct methods solve the system exactly in infinite precision arithmetic, using a number of operations that is some fixed function of the problem size. For nonsingular systems some version of LU factorization is commonly used [26], although other schemes such as the WZ factorization [21] have been developed for parallel computers. These factorization approaches are effective for medium sized and dense or narrowly banded systems. The obstacle to using them for large sparse systems is *fill-in*, the presence of many more nonzero entries in the factors than in A. Matrix reordering algorithms [23] can greatly reduce fill-in but for many problems the additional computer memory required still exceeds allowable limits. Another problem with factorization methods is that for the most part they are inherently sequential and not suitable for vector or multiprocessor machines.

Iterative methods generate a sequence of approximate solutions and can circumvent the difficulties associated with direct solvers. The system can be solved to a predetermined accuracy. If this accuracy is much larger than the machine precision the number of operations required can be far less than that required by direct solvers. Iterative methods generally can be implemented with a small number of additional vectors for storage, avoiding the excessive memory requirements caused by fill-in. Finally, iterative methods usually are based on operations such as matrix-vector products, which can be vectorized well.

The most onerous price that must be paid for using iterative methods is a loss of robustness. While the LU factorization with pivoting exists for any nonsingular matrix, only one commonly used iterative method shares this theoretical robustness, the conjugate gradient (CG) algorithm applied to the normal equations of the system. Other iterative solvers require special properties of the coefficient matrix. Nevertheless for applications such as self-adjoint elliptic problems for which these properties can be easily verified in advance iterative solvers are more effective than direct solvers.

1.4 Existing Nonsymmetric Iterative Solvers

For symmetric positive definite systems the method of conjugate gradients (CG) was recognized as an effective iterative solver beginning in 1971 [48], although it had been proposed as a direct solver as early as 1952 by M. Hestenes and E. Stiefel [34]. CG has the remarkable properties of requiring no parameter estimation, giving monotonically decreasing errors in an elliptic norm, and in exact arithmetic ending with the solution x^* after at most N steps. The process works by implicitly maintaining orthogonality conditions between the residuals and search directions generated, and simultaneously minimizing a norm of the error vector on a Krylov subspace.

The success of CG on symmetric positive definite systems led several researchers beginning in the mid 1970's to try to duplicate the algorithm for nonsymmetric systems. Exploiting the connection between CG and the symmetric Lanczos algorithm, in 1975 R. Fletcher [22] proposed the biconjugate gradient algorithm which is analogously based on the Lanczos biorthogonalization algorithm. Unfortunately it also shares the numerical instability of Lanczos biorthogonalization and has not achieved practical success. In 1976 P. Concus and G. Golub [13] published a generalization of the CG method to nonsymmetric problems, but it was directly applicable only to problems where the symmetric part $(A + A^T)/2$ of A is positive definite.

Beginning in 1976 [60] several researchers proposed a number of CG-like algorithms, in particular generalized conjugate residuals (GCR), Orthomin, generalized minimum residuals (GMRES), Axelsson's method, Orthodir (see [50] for a presentation of these methods in a unified framework and references for their origins.) These methods, like CG, generate a Krylov subspace using only matrix-vector products with A and then enforce some minimization or orthogonality property, called the *defining property* here, on that space. They differ primarily in how the basis of the Krylov subspace is generated and which inner product is used to maintain the defining property. These CG-like algorithms also guarantee convergence in exact arithmetic within N iterations, but with storage and computational costs that grow on each iteration. This immediately led to the ideas of restarting the algorithms after every k steps, or truncating the algorithms by requiring their defining properties to hold only for the last k vectors generated.

In 1982 H. Elman [20] proved that the restarted versions of many of these methods converge provided that the symmetric part $(A + A^T)/2$ of A is positive definite. As the testing in this thesis will show, for practical problems this condition cannot be removed. In spite of this restriction the restarted GMRES(k) algorithm in particular is currently one of the more popular methods for nonsymmetric systems.

An independent class of algorithms, the residual polynomial methods, depends on finding a convex body in the complex plane that contains the eigenvalues of A. Iterates x_k are then formed so that the corresponding residuals r_k satisfy $r_k = P_k(\zeta)r_0$, where P_k is a polynomial with $P_k(0) = 1$ and $P_k(\zeta) \to 0$ for complex numbers ζ in that convex body. Chebyshev methods [43] use ellipses for the convex body, and Chebyshev polynomials to define the P_k's. T. Manteuffel [44] has developed adaptive procedures for estimating the necessary parameters that define the ellipse. Least squares polynomial methods use the convex hull $\mathrm{cvx}(\sigma(A))$ of the extremal eigenvalues of A for the enclosing body, and then define the polynomials P_k to minimize a weighted sum of squares on the boundary of $\mathrm{cvx}(\sigma(A))$. P. Saylor [51] has proposed these methods and also shown how to combine the eigenvalue estimation phase with the iterative phase to save on matrix-vector multiplications.

Residual polynomial methods fail when the convex hull of the eigenvalues of A contains the origin since the Minimum Principle implies that the conditions $P_k(0) = 1$ and $P_k(\zeta) \to 0$ for $\zeta \in \mathrm{cvx}(\sigma(A))$ are contradictory. Sufficient conditions for convergence of these methods are that the eigenvalues all lie in a halfspace of the complex plane that does not contain the origin.

One method which in theory converges for any nonsymmetric A is CG applied to the normal equations $A^T Ax = A^T b$, which dates back to numerical analysis antiquity [34]. One weakness of this approach is a loss of information and a large preprocessing cost if $A^T A$ is explicitly formed. Even if this is not done, a more serious problem is that if A has small singular values, then $A^T A$ has even smaller eigenvalues and hence is a far worse conditioned matrix than A. As the numerical testing of Chapter 6 shows, this can lead to outright failure of the method.

In summary, most iterative methods either require storage and computation that grows excessively with the iteration number, special spectral properties of the matrix A, or a symmetrization process that has a potentially disastrous effect on the spectrum of the coefficient matrix used.

One class of methods that avoids these pitfalls is the group of CG accelerated block SSOR methods examined by C. Kamath [38] and by C. Kamath and A. Sameh [39] in 1986. For structured problems such as block tridiagonal systems these methods can be implemented with parallelism, and their robustness has been demonstrated on matrice arising in the solution of two dimensional elliptic partial differential equations. Motivated by this success, this thesis places the algorithm in [38] and [39] in the general framework of row projection methods, proposes a new algorithm with computational advantages, suggests and tests an implementation for three dimensional problems, and provides a more detailed theoretical foundation for the resulting algorithms.

1.5 Goals and Overview of Thesis

The practical purpose of this thesis is the development of algorithms for solving large sparse nonsymmetric systems of linear equations. The algorithms should achieve the goals of

1. Requiring no parameter estimation from the user.

2. Requiring only a few additional vectors of storage; the number of necessary vectors must not grow with the problem size.

3. Having efficient implementations on a variety of modern computers. In particular they must be suitable both for vector and multiprocessor machines, and exhibit good data locality.

4. Possessing robustness; they must not rely on special spectral properties of the systems being solved.

Of these four goals, the last is of overriding importance.

For an overview of the thesis, Chapter 2 defines and presents the theory of the two classes of row projection methods. Past work is summarized, convergence proofs are given, and a new RP iteration with theoretical and computational advantages is presented. A complete eigenvalue analysis is given for the special case where A is partitioned into two block rows, and an initial comparison is made of the two classes of RP methods.

In Chapter 3 the application of RP methods to seven point difference operator matrices is shown is detail. Test problems are introduced and analyzed, and two partitioning schemes are proposed that allow efficient implementation on vector and multiprocessor machines. Chapter 4 provides an error analysis for the methods, and the particular implementation chosen is shown to yield well-conditioned subproblems and stable algorithms.

Chapter 5 discusses acceleration techniques for RP methods. For conjugate gradient acceleration the special properties of RP methods are shown to allow a significant reduction in the computational requirements. A detailed analysis of the block Stiefel technique is provided that is applicable to any iterative method and not just RP iterations.

Chapter 6 demonstrates the robustness and efficiency of RP methods compared to other nonsymmetric solvers on the test problems. An explanation of the behaviour of RP algorithms is given which also identifies categories of problems for which they are especially suitable and methods of enhancing their performance. Finally, Chapter 7 summarizes the results and presents conclusions.

1.6 Notation

The notation used generally follows that introduced by Householder. Capital Roman letters denote matrices, lower case Roman letters vectors, and Greek letters scalars. The only exceptions to this are the letters N, M, m, n, i, j, and k which refer to integers, and Ω_i which refers to certain weighting matrices. Other notation used includes the following:

- A^T = the transpose of the matrix A
- $\| A \|$= the two norm of A
- $\| A \|_F$= the Frobenius norm of A
- $A^\dagger$ = the Moore-Penrose pseudo-inverse of A
- $\Re^{M \times N}$ = the set of M by N matrices with real entries
- $\Re^N$ = the set of N by 1 column vectors
- range(A) = the range space of A
- null(A) = the null space of A
- P_L = the orthogonal projector matrix onto the subspace L
- $\sigma(A)$ = the spectrum or set of eigenvalues of matrix A
- $\sigma_{min}(A)$ = the smallest singular value of A
- $\sigma_{max}(A)$ = the largest singular value of A
- $\lambda_{min}(A)$ = the smallest eigenvalue of A
- $\lambda_{max}(A)$ = the largest eigenvalue of A
- $\rho(A)$ = the spectral radius of A, i.e., the maximum modulus over all of the elements in $\sigma(A)$
- (α, β) = the open interval from α to β
- $[\alpha, \beta]$ = the closed interval from α to β

- $[\alpha, \beta)$ = the half-open interval from α to β, including α but not β
- I = the identity matrix (of order conformable to the context)
- $\kappa(A)$ = the two norm condition number of A, that is, $\| A \| \ \| A^{-1} \|$
- $D = diag(\alpha_1, \alpha_2, \ldots, \alpha_N)$ is the diagonal matrix with main diagonal given by the α's
- e_i is the i^{th} unit coordinate vector

When describing an iteration of the form $x_{k+1} = Qx_k + b$, where $Q \in \Re^{N \times N}$, k is the iteration counter. Q is called the *iteration matix*, and $I - Q$ is called the *system matrix* because the iteration can be viewed as an attempt to solve the system $(I - Q)x = b$. Additional notation is defined in the text as needed.

Chapter 2

Theory of Row Projection Methods

2.1 Introduction

Consider the linear system

$$Ax = b \tag{2.1}$$

where $A \in \Re^{M \times N}$ and $b \in \Re^{M}$. Partition the rows of A into m blocks, denoted by

$$A^T = [A_1, A_2, \ldots, A_m], \tag{2.2}$$

and partition the vector b conformally. A *row projection* (RP) method for solving (2.1) is any algorithm which requires computing orthogonal projections $A_i A_i^{\dagger} x$ of a vector x onto the range spaces of the A_i's. Two special cases of row partitioning occur frequently enough in the literature to warrant special mention. When $m = 2$, A is divided into two blocks. As will be discussed in Section (2.4), in this case a complete theoretical analysis of RP methods is possible in terms of canonical angles between the two induced subspaces range(A_1) and range(A_2). When $m = M$, each A_i^T is a single row a_i^T of A, and the computation of the projections becomes especially simple. Much of the early work on RP methods dealt exclusively with this case.

One assumption is now made that will hold throughout this thesis: each A_i is of full rank. In practice this can always be achieved by taking $m = M$ and imposing the reasonable requirement that each row of A is nonzero.

Although this assumption is not necessary for convergence of some of the algorithms presented here, it does greatly simplify the notation and proofs.

The next chapter will show in detail how to select m and the row partitioning for a particular class of practical problems. For now, however, the partitioning is allowed to be arbitrary.

The RP methods examined here fall into two general categories, the Kaczmarz or product forms and the Cimmino or sum forms. The following sections will present a history of these two approaches, their relationship with the Jacobi and SOR methods applied to related systems, convergence results, and a preliminary comparison. However, simple intuitive arguments can show the strong geometric flavor of RP methods and help explain their remarkable robustness properties. Suppose that A is square and nonsingular, and let

$$H_i = \{x : A_i^T x = b_i\} \tag{2.3}$$

be the affine set of solutions to the i^{th} block row of equations. The solution x^* to (2.1) is then the unique intersection point of those affine sets, i.e., the unique point that satisfies all of the equations simultaneously. Referring to Figure (2.1) for the two dimensional case, an initial estimate x_0 can be orthogonally projected from one affine set to the next, eventually converging to x^*. This solution process is simply the method of successive projections applied to finding the intersection of the sets H_i.

Let $P_i = A_i(A_i^T A_i)^{-1}A_i^T$ for $i = 1, 2, \ldots, m$. The orthogonal projection of x onto H_i is the point

$$(I - P_i)x + \hat{b}_i \tag{2.4}$$

where $\hat{b}_i = (A_i^T)^\dagger b_i$, and so the successive projection process above can be written as

$$x_{k+1} = (I - P_1)(I - P_2)\cdots(I - P_m)x_k + b_\pi, \tag{2.5}$$

where

$$\begin{aligned} b_\pi \;=\; & \hat{b}_1 + (I - P_1)\hat{b}_2 + (I - P_1)(I - P_2)\hat{b}_3 \\ & +(I - P_1)\cdots(I - P_{m-1})\hat{b}_m \end{aligned} \tag{2.6}$$

An efficient method for computing the modified right hand side vector b_π is shown in Chapter 6. Iteration (2.5) forms the next iterate using the product of orthogonal projectors, and was first proposed by S. Kaczmarz in 1939 for the partitioning where each block row A_i^T is comprised of a single row of A.

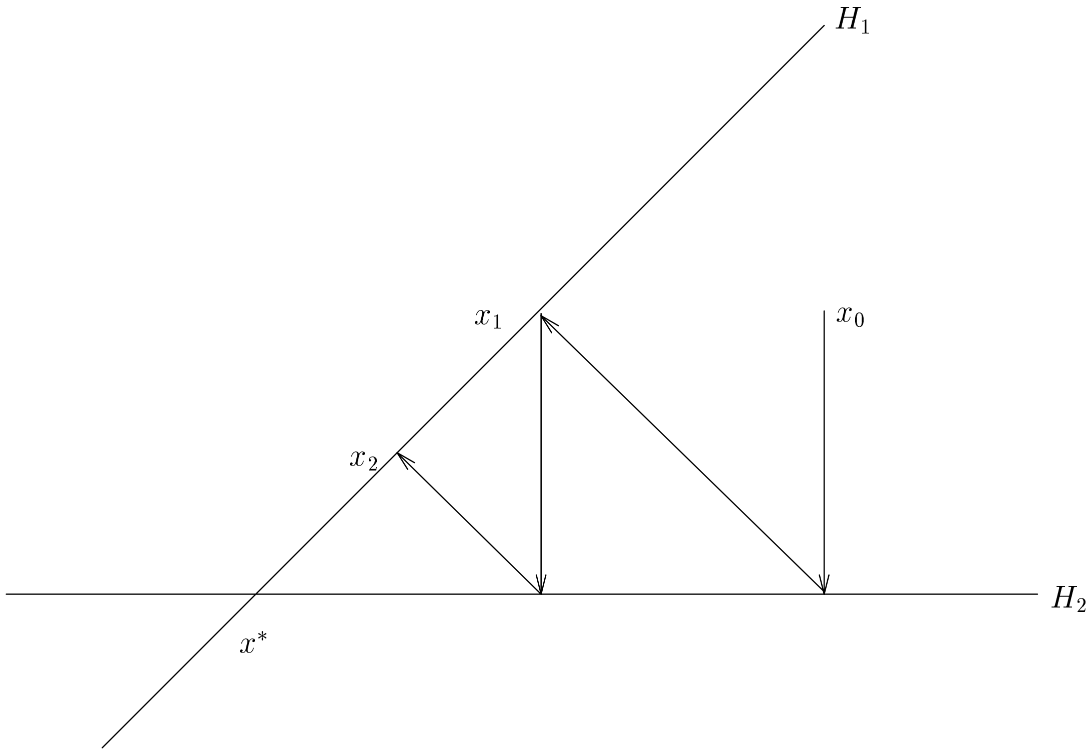


Figure 2.1: The Kaczmarz Iteration

For these reasons iterations of the form (2.5) are called *product* or *Kaczmarz* forms of RP methods.

The second category of RP methods can be motivated by considering the homogeneous system $Ax = 0$. Instead of projecting x_0 onto H_i, let y_i be the reflection of x_0 through H_i. Then geometrically x_0 and each y_i lie on a circle centered at the solution $x^* = 0$ as in Figure (2.2).

Any nontrivial convex combination of those points must lie strictly inside the circle and hence be closer to the solution, forming the basis of an iterative method. The reflection y_i is given by

$$y_i = (I - 2P_i)x_0 + 2\hat{b}_i \tag{2.7}$$

where P_i and $\hat{b}_i$ are as defined in the Kaczmarz form. If the weights defining the convex combination are denoted by ω_i, the iteration becomes

$$x_{k+1} = \left[I - 2\sum_{i=1}^{m} \omega_i P_i\right] x_k + b_\sigma \tag{2.8}$$

where

$$b_\sigma = 2\sum_{i=1}^{m} \omega_i \hat{b}_i \tag{2.9}$$

Figure (2.2) shows the next iterate for $\omega_1 = \omega_2 = 1/m$. Iteration (2.8) was first proposed by G. Cimmino in 1937 for the $m = M$ partitioning, and since the sum of row projections is required such iterations are called *sum* or *Cimmino* forms of RP methods.

Note that the simple geometric derivations above place no restrictions on the spectrum of A, and do not require symmetry or other properties. Although nonsingularity is assumed in Figure (2.1), the result of having a singular A is readily pictured; the affine sets are parallel and so the subsequences $\{x_{km+i}\}_{k=0}^{\infty}$ for $i = 1, 2$ are convergent. This geometric aspect and apparent robustness are two of the more appealing features of RP methods. In Chapter 3 they are also shown to be implementable in efficient ways for a wide class of important problems.

The RP methods considered here are cyclic in that they sweep through the block rows of A in a fixed order. Other RP schemes are also possible; for example, at each step k the affine set H_i to project onto next can be selected as one for which $\| b_i - A_i^T x_k \|$ is maximal. Unfortunately this requires computing a matrix-vector product after every projection, which is

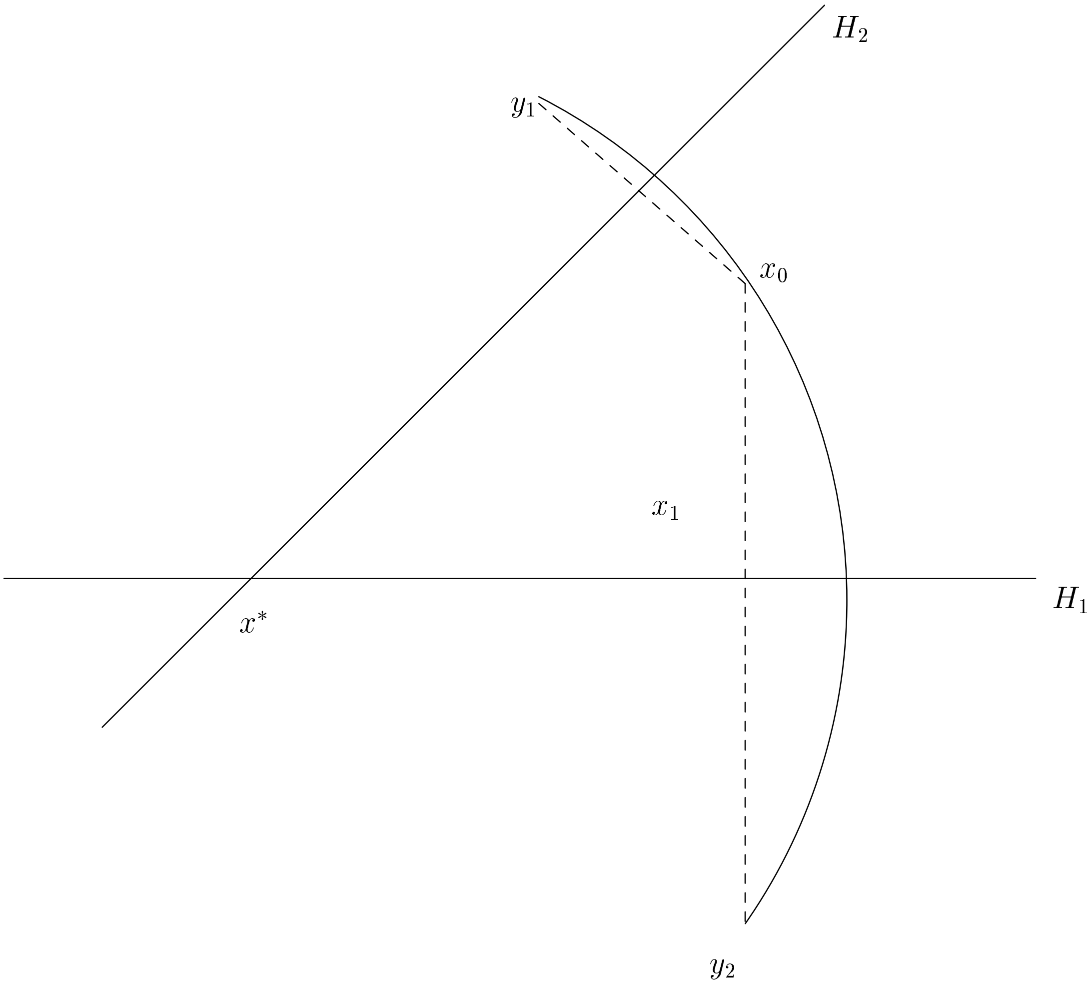


Figure 2.2: The Cimmino Iteration

expensive. More complicated 'quasi-optimal' schemes have also been developed which adaptively find parameters that guide the choice of which H_i should be used next. R. Wainwright and R. Keller's work in [61] contains a discussion of and references for such approaches. These methods are not considered further here because they require extensive parameter estimation, are not directly extensible to blocks of size larger than 3, and yield nonstationary iterative processes that are not amenable to the powerful acceleration methods discussed in Chapter 5.

2.2 Properties of Kaczmarz Methods

2.2.1 History of Kaczmarz Methods

Because of the simplicity of RP methods, they have been periodically rediscovered. S. Kaczmarz [37] proposed the iteration (2.5) and proved convergence for square nonsingular matrices when $m = M$ in 1939, although F. Deutsch [14] cites a set of mimeographed lecture notes by J. von Neumann from 1933 as establishing convergence for the $m = 2$ case. I. Halperin [29] extended this proof to $m > 2$ in 1962. Both von Neumann and Halperin were considering the problem where P_i is the projector onto a closed subspace in a Hilbert space and so their iteration is actually the classical successive or alternating projection method for homogeneous systems. Using the notation of the previous section, their results are that if P is the projection onto the space

$$\bigcap_{i=1}^{m} range(I - P_i) = \bigcap_{i=1}^{m} null(A_i^T)$$

and $Q_\pi = (I - P_1)(I - P_2)\cdots(I - P_m)$, then for every x

$$\lim_{k\to\infty} \| (Q_\pi^k - P)x \| = 0$$

Deutsch also established the rate of convergence for this limit in the $m = 2$ case.

In 1971 K. Tanabe [57] proved that the iteration (2.5) with $m = M$ converges even when A is rectangular or singular, without requiring that the system be consistent. His result was that

$$x_k \to Gb + P_{null(A)}x_0, \tag{2.10}$$

where $P_{null(A)}$ is the orthogonal projector onto the null space of A and G is a generalized inverse of A satisfying $AGA = A$, $GAG = G$, $GA =$ the orthogonal projector onto range(A^T), and $AG =$ the oblique projector onto range(A) along null(T), where

$$T = \left[(A_1^T)^\dagger, (I - P_1)(A_2^T)^\dagger, \ldots, (I - P_1)(I - P_2)\cdots(I - P_{m-1})(A_m^T)^\dagger\right] \tag{2.11}$$

Note that T is the matrix mapping the right hand side vector b into the modified right hand side vector b_π, and G is not in general the pseudoinverse $A^\dagger$ because AG is not the orthogonal projector onto range(A). However, when (2.1) is a consistent system and $x_0 \in$ range(A^T), $G = A^\dagger$ and $x_k \to A^\dagger b$. Tanabe's approach has, with few exceptions, been the basis for proofs involving arbitrary block sizes and the introduction of iteration parameters.

Kaczmarz's method was independently rediscovered by R. Gordon, R. Bender, and G. Herman [27] in 1970 and further developed by several researchers for solving linear systems arising in the field of medical image reconstruction. Called algebraic reconstruction techniques (ART), versions exist which take into account constraints, provide regularized least squares solutions, solve quadratic minimization problems, and solve systems of linear inequalities. References for these applications can be found in [10] and [33].

When the system (2.1) is consistent, the Kaczmarz forms are equivalent to the block Gauss-Seidel method applied to the system

$$\left\{ \begin{array}{rcl} AA^Ty & = & b \\ x & = & A^Ty \end{array} \right. \tag{2.12}$$

This naturally leads to considering an SOR analogue to (2.5) which if the relaxation parameter is allowed to depend on the projector P_i and the iteration number k becomes

$$x_{k+1} = (I - \omega_1^{(k)}P_1)(I - \omega_2^{(k)}P_2)\cdots(I - \omega_m^{(k)}P_m)x_k + T_kb, \tag{2.13}$$

with the i^{th} block of T_k given by

$$\omega_i^{(k)}\left(\prod_{j=1}^{i-1}(I - \omega_j^{(k)}P_j)\right)(A_i^T)^\dagger \tag{2.14}$$

G. Herman, A. Lent and D. Lutz [32] in 1978 showed that if the system (2.1) is consistent and $m = M$, (2.13) converges provided that

$$0 < \liminf_{k\to\infty} \omega_i^{(k)} \leq \limsup_{k\to\infty} \omega_i^{(k)} < 2 \tag{2.15}$$

In 1976 W. Peters [47] showed convergence for the block version ($m < M$) of (2.5), provided the system is consistent and each A_i is full rank (as is assumed for convenience in this thesis.) T. Elfving [17] in 1977 generalized this result by dropping the requirement that each A_i be of full rank, and showed that if $\omega_i^{(k)} = \omega$ is fixed, (2.13) converges iff $0 < \omega < 2$. As with the $m = M$ case, these block versions converge to $A^\dagger$ when the system is consistent and $x_0 \in \text{range}(A^T)$.

The most general conditions for the weights in Kaczmarz methods were established in 1981. If the relaxation parameters are replaced by matrices $\Omega_i^{(k)}$, (2.13) becomes

$$x_{k+1} = \prod_{i=1}^{m} \left(I - A_i(A_i^T A_i)^{-1}\Omega_i^{(k)} A_i^T\right) x_k + \hat{T}_k b \tag{2.16}$$

where the j^{th} block column of $\hat{T}_k$ is

$$\left(\prod_{i=1}^{j-1} \left(I - A_i(A_i^T A_i)^{-1}\Omega_i^{(k)} A_i^T\right)\right) A_j(A_j^T A_j)^{-1}\Omega_j^{(k)} \tag{2.17}$$

For this iteration P. Eggermont, G. Herman, and A. Lent [16] showed that if

$$\limsup_{k\to\infty} \ \| \ (A_i^T)^\dagger (I - \Omega_i^{(k)}) A_i^T \ \| < 1,$$

then x_k converges in the consistent case to a solution of (2.1). If (2.1) is inconsistent, then each subsequence $\left\{x^{nm+k}\right\}_{n=1}^{\infty}$, $k = 1, 2, \ldots, m$, converges although they did not characterize the limit points. Note that this iteration is not a RP method according to the definition used here, but is clearly closely related. Apparently no one has used these results except for the $\Omega_i^{(k)} = \omega_i^{(k)} I$ case.

When the parameters are fixed to be a constant ω, the convergence of Kaczmarz type methods to the generalized inverse solution (2.10) is not mathematically satisfying, although in the field of image reconstruction such solutions seem sufficient to generate realistic images. A. Trummer [58, 59] in 1981 has shown that for the $m = M$ case with $x_0 \in \text{range}(A^T)$, iteration (2.13) converges to the weighted minimum norm least squares solution $(DA)^{\dagger} Db$ provided that the weights are fixed during a given cycle through the m projections, that is, depend only on k and not i, and satisfy

$$\omega^{(k)} \geq 0, \quad \sum_{k=1}^{\infty} (\omega^{(k)})^2 < \infty, \quad \sum_{k=1}^{\infty} \omega^{(k)} = \infty \tag{2.18}$$

Here $D^{-1} = diag(\| a_1 \|, \| a_2 \|, \ldots, \| a_M \|)$, $a_i^T = i^{th}$ row of A. This means that if the weights go to zero but not too quickly, a weighted least squares solution can be found. In 1983 Y. Censor, P. Eggermont, and D. Gordon [11] proved a similar result for the block case of the related iteration

$$\begin{aligned} x_{k+1} &= (I - \omega A_1 A_1^T)(I - \omega A_2 A_2^T) \cdots (I - \omega A_m A_m^T) x_k \\ &\quad + \omega \textstyle\sum_{i=1}^{m} \prod_{j=1}^{i-1} (I - \omega A_j A_j^T) A_i b_i \end{aligned} \tag{2.19}$$

They showed that for (2.19), if ω is sufficiently small then x_k converges to some x_ω, and

$$\lim_{\omega \to 0} x_\omega = [I - A^{\dagger} A] x_0 + A^{\dagger} b. \tag{2.20}$$

This is not a RP method unless the block rows of A have been scaled so that the columns of each A_i are orthonormal. Also, as $\omega \to 0$ the iteration (2.19) becomes more difficult to solve. The effectiveness of block RP methods for solving *unweighted* minimum norm least squares problems has not yet been demonstrated.

Bounds for the rate of convergence when all of the weights $\omega_i^{(k)} = 1$ are given in [29] and improved bounds for the case $m = 2$ in [14]. These bounds are subsumed in the more general results given by S. Nelson and M. Neumann [46] in 1987 which are now summarized for the application to Kaczmarz type RP methods. Let $\omega_i^{(k)} = \omega_i$ not depend on the iteration number k, suppose $0 < \omega_i < 2$ and set

$$Q_\pi = (I - \omega_1 P_1)(I - \omega_2 P_2) \cdots (I - \omega_m P_m)$$

Then $\text{null}(Q_\pi) = \bigcap_{i=1}^{m} \text{null}(A_i^T)$ as in Deutsch's work, and $Q_\pi^k x$ converges for any x at a linear rate, with a convergence constant bounded above by

$$1 - \prod_{i=1}^{m} (1 - (1 - \omega_i)^2) \prod_{i=1}^{m} \sin^2 \theta_i \tag{2.21}$$

Here θ_i is the smallest angle between the subspaces $\text{null}(A_i^T)$ and $\bigcap_{j=i+1}^{m} \text{null}(A_j^T)$ (see [7] for a definition of the angles θ_i.)

The essentials of the above results may be summarized.

- When $\omega_i^{(k)} = 1$ and $m = M$ the iteration

$$x_{k+1} = (I - P_1)(I - P_2) \cdots (I - P_m)x_k + Tb$$

converges to $P_{null(A)}x_0 + Gb$, where G is a generalized inverse of A and T is given by (2.11). Hence when $x_0 \in \text{range}(A^T)$ and $b \in \text{range}(A)$, $x_k \to A^{\dagger}b$.

- When $0 < \omega_i < 2$ and m is arbitrary, the iteration

$$x_{k+1} = (I - \omega_1 P_1)(I - \omega_2 P_2) \cdots (I - \omega_m P_m)x_k + Tb,$$

with the i^{th} block column of T as given in (2.14) with $\omega_i^{(k)} = \omega_i$, converges but no characterization of the limit point has yet been established. However, if additionally $x_0 \in \text{range}(A^T)$ and $b \in \text{range}(A)$, $x_k \to A^{\dagger}b$.

- When $\omega^{(k)}$ go to zero in a way satisfying (2.18), $m = M$, and $x_0 \in \text{range}(A^T)$, the iteration

$$x_{k+1} = (I - \omega^{(k)}P_1)(I - \omega^{(k)}P_2) \cdots (I - \omega^{(k)}P_m)x_k + T_k b,$$

converges to $(DA)^{\dagger}Db$ even if the system is inconsistent. Here $D^{-1} = diag(\| a_1 \|, \| a_2 \|, \ldots, \| a_M \|)$, $a_i^T = i^{th}$ row of A.

In Section 2.2.4 a theorem proving convergence and characterizing the limit for a symmetrized Kaczmarz type RP method is given for the case when x_0, m, and b are arbitrary.

2.2.2 Symmetrization of Product Forms

The bound given by (2.21) shows that the rate of convergence is determined by the smallest angles between subspaces $\text{null}(A_i^T)$ and $\bigcap_{j=i+1}^{m}\text{null}(A_j^T)$; a single small angle can make the convergence disastrously slow even if the other angles between the subspaces are nearly orthogonal. For this reason it is desirable to accelerate the iterative process.

As Elfving noted in [17] the Kaczmarz type RP iteration with $\omega_i^{(k)} = \omega$ fixed is equivalent to a block SOR method applied to (2.12). Except in the $m = 2$ case, as will be shown later, the eigenvalues of the iteration matrix are generally not real even though $0 < \omega < 2$ guarantees that they all have moduli bounded by 1. For these reasons it is natural to consider the block SSOR method applied to (2.12) as is done in [39] and [17]. The resulting iteration matrix is symmetric and since the block SSOR method is equivalent to block SOR applied to $\tilde{A}x = \tilde{b}$ with

$$\tilde{A}^T = [A_1, A_2, \ldots, A_m, \ldots, A_2, A_1]$$

and $\tilde{b}$ defined analogously, the eigenvalues of the resulting iteration matrix must lie in the interval $(-1, 1)$. This allows powerful acceleration techniques such as Chebyshev and conjugate gradients to be applied. A full discussion of acceleration methods is provided in Chapter 5. Again the iteration parameters can be allowed to vary between blocks A_i^T, but now they must remain fixed between iterations. The resulting Kaczmarz form of RP for $Ax = b$ becomes

$$x_{k+1} = Q_\pi x_k + b_\pi \tag{2.22}$$

with

$$Q_\pi = (I - \omega_1 P_1)(I - \omega_2 P_2) \cdots (I - \omega_m P_m) \cdots (I - \omega_2 P_2)(I - \omega_1 P_1), \tag{2.23}$$

$b_\pi = T_\pi b$, and the i^{th} block column of T_π given by

$$\omega_i \prod_{j=1}^{i-1} (I - \omega_j P_j) \left[I + \prod_{j=i}^{m} (I - \omega_j P_j) \prod_{j=m-1}^{i+1} (I - \omega_j P_j) \right] (A_i^T)^\dagger \tag{2.24}$$

The first product above should be interpreted as I when $i = 1$ and the third product should be interpreted as I when $i = m - 1$ and 0 when $i = m$, so that the first summand in forming b_π is

$$\omega_1 [I + (I - \omega_1 P_1) \cdots (I - \omega_m P_m) \cdots (I - \omega_2 P_2)] (A_1^T)^\dagger,$$

the $(m-1)^{st}$ summand in forming b_π is

$$\omega_{m-1} \prod_{j=1}^{m-2} (I - \omega_j P_j) \left[I + (I - \omega_{m-1} P_{m-1})(I - \omega_m P_m) \right] (A_{m-1}^T)^\dagger b_{m-1}$$

and the m^{th} summand in forming b_π is

$$\omega_m \prod_{j=1}^{m-1} (I - \omega_j P_j)(A_m^T)^\dagger b_m.$$

2.2.3 A New Kaczmarz Method

A closely related iteration is now introduced which has computational advantages over (2.22). Let $A_i = Q_i R_i$ be the QR decomposition of A_i for $i = 1, 2, \ldots, m$. Note that $P_i = Q_i Q_i^T$ in this case. Let $AA^T = L + D + L^T$ be the decomposition of AA^T into the strictly lower block triangular, block diagonal, and strictly upper block triangular parts induced by the block row partitioning. As shown in [31, p.31], the block SSOR iteration for $AA^T = y$ can be written as

$$y_{k+1} = [I - \omega(2-\omega)(D+\omega L)^{-T} D (D+\omega L)^{-1} AA^T] y_k + \omega(2-\omega)(D+\omega L)^{-T} D (D+\omega L)^{-1} b \tag{2.25}$$

This can be viewed as an iteration to solve the system

$$[(D + \omega L)^{-T} D (D + \omega L)^{-1} AA^T] y = (D + \omega L)^{-T} D (D + \omega L)^{-1} b \tag{2.26}$$

The product RP form replaces $A^T y$ with x and premultiplies by A^T, or equivalently applies a similarity transformation with $W = A^T$ to the system matrix in (2.26) to obtain

$$[A^T (D + \omega L)^{-T} D (D + \omega L)^{-1} A] x = A^T (D + \omega L)^{-T} D (D + \omega L)^{-1} b \tag{2.27}$$

which can be shown to be identical to (2.22) with $\omega_i = \omega$. In [6] A. Björck and T. Elfving proposed applying instead a similarity transformation using $W = D^{-1/2}(D + \omega L)^T$; they showed that when CG acceleration is used this has the desirable property of being an error reducing process in the two-norm, instead of reducing the error in an elliptic norm. Unfortunately, computing $D^{-1/2}$ is only practical in the $m = M$ case.

However, the factors R_i can be practically computed in the block case, either from a QR decomposition of A_i or a Cholesky factorization of $A_i^T A_i$ since A_i is of full rank. Let

$$R = diag(R_1, R_2, \ldots, R_m) \in \Re^{N \times N} \tag{2.28}$$

and now consider applying a similarity transformation using $W = R^{-T}(D + \omega L)^T$ to (2.26). The resulting system becomes

$$R(D+\omega L)^{-1}AA^T(D+\omega L)^{-T}R^Tz = R(D+\omega L)^{-1}b \tag{2.29}$$

with

$$z = R^{-T}(D+\omega L)^T y \tag{2.30}$$

The corresponding iteration becomes

$$z_{k+1} = V_\pi z_k + R(D+\omega L)^{-1}b \tag{2.31}$$

with

$$V_\pi = I - R(D+\omega L)^{-1}AA^T(D+\omega L)^{-T}R^T \tag{2.32}$$

and the solution x retrieved from z at the end by

$$x = A^T y = A^T(D+\omega L)^{-T}R^T z \tag{2.33}$$

In Chapters 3 and 5 the theoretical and computational advantages of this new form of the block SSOR iteration are examined. For now, note that both (2.22) and (2.31) have coefficient matrices with the same eigenvalues since they are both similarity transformations of the block SSOR matrix.

2.2.4 Convergence of Symmetrized Kaczmarz RP Methods

Using techniques similar to those in [30] and [57] the properties of iteration (2.22) are now stated.

Theorem 2.1 *Let $A \in \Re^{M\times N}$ be partitioned as in (2.2) with each A_i of full rank. Let $x_0 \in \Re^N$, $b \in \Re^M$ be arbitrary. For any choice of $\omega_i \in (0,2)$, $i = 1,2,\ldots,m$,*

(i) *The spectrum $\sigma(I - Q_\pi)$ lies in the interval $[0,2]$.*

(ii) *If A is nonsingular, $\sigma(I - Q_\pi) \subseteq (0,2)$.*

(iii) *Iteration (2.22) converges to $P_{null(A)}x_0 + Gb$, where G is a generalized inverse of A satisfying $GAG = G$, $AGA = A$, $GA =$ the orthogonal projector onto range(A^T), and $AG =$ the oblique projector onto range(A) along null$(A(T_\pi A)^\dagger T_\pi)$.*

Proof:

(i) For a given i in $\{1, 2, \ldots, m\}$ the eigenvalues of $I - \omega_i P_i$ are either 1 corresponding to eigenvectors in null(P_i) or $1 - \omega_i$ corresponding to eigenvectors in range(P_i). Hence

$$\rho(Q_\pi) \leq \prod_{i=1}^{m} \rho(I - \omega_i P_i) \leq \prod_{i=1}^{m} \max\{1, |1 - \omega_i|\} \leq 1 \tag{2.34}$$

and so $\sigma(I - Q_\pi) \in [0, 2]$.

(ii) Suppose some eigenvector x of $I - Q_\pi$ corresponds to an eigenvalue of 0, so that $Q_\pi x = x$. From (i), this implies that x is an eigenvector corresponding to 1 for $I - \omega_i P_i$, $i = 1, 2, \ldots, m$ since $\omega_i \in (0, 2)$ implies that $1 - \omega_i \in (-1, 1)$. Hence

$$x \in \bigcap_{i=1}^{m} null(P_i) = \bigcap_{i=1}^{m} null(A_i^T) \tag{2.35}$$

and so $Ax = 0$. Since A is nonsingular, this implies $x = 0$, a contradiction since x is an eigenvector. A similar argument shows that no eigenvalue of $I - Q_\pi$ can be equal to 2, so $\sigma(I - Q_\pi) \subseteq (0, 2)$.

(iii) From the proof of (ii), null$(I - Q_\pi) \subseteq$ null(A) for any A. If $x \in$ null(A), then $A_i^T x = 0$ for all i and so

$$(I - \omega_i P_i)x = (I - \omega_i A_i (A_i^T A_i)^{-1} A_i^T)x = x \tag{2.36}$$

Then $(I - Q_\pi)x = 0$ and null$(I - Q_\pi) =$ null(A). Note that this also shows that range$(I - Q_\pi) =$ range(A^T). From [4, p.198], the iteration converges to

$$x_\omega = [I - (I - Q_\pi)(I - Q_\pi)^D]x_0 + (I - Q_\pi)^D T_\pi b \tag{2.37}$$

where $(I - Q_\pi)^D$ denotes the Drazin inverse of $(I - Q_\pi)$. Since Q_π is symmetric, the Drazin inverse is actually the pseudo-inverse ([3, pp.162-172]). Hence $(I - Q_\pi)(I - Q_\pi)^D$ is the orthogonal projector onto null$(I - Q_\pi) =$ null(A).

It can be shown from the definition of T_π that $T_\pi A = I - Q_\pi$ (c.f. the result in [57]). Hence

$$\begin{aligned} x_\omega &= P_{null(A)} x_0 + (I - Q_\pi)^\dagger T_\pi b \\ &= P_{null(A)} x_0 + (T_\pi A)^\dagger T_\pi b \end{aligned} \tag{2.38}$$

Let $G = (T_\pi A)^\dagger T_\pi$. Then from the defining properties of pseudo-inverses and their relationship with orthogonal projectors,

$$\begin{aligned} GAG &= (T_\pi A)^\dagger T_\pi A (T_\pi A)^\dagger T_\pi \\ &= (T_\pi A)^\dagger T_\pi = G \end{aligned} \tag{2.39}$$

$$\begin{aligned} AGA &= A(T_\pi A)^\dagger T_\pi A \\ &= AP_{range(I-Q_\pi)} \\ &= AP_{range(A^T)} = AA^\dagger A = A \end{aligned} \tag{2.40}$$

This also shows that GA is the orthogonal projector onto range(A^T). Finally, $AG = A(T_\pi A)^\dagger T_\pi$ is an oblique projector since $AGA = A$. Range(AG) is clearly contained in range(A). If z is in the orthogonal complement of range(AG), $z^T A(T_\pi A)^\dagger T_\pi v = 0$ for all v; replacing v with the columns of A shows that

$$0 = z^T A(T_\pi A)^\dagger T_\pi A = z^T AP_{range(A^T)} = z^T A \tag{2.41}$$

and so z is in the orthogonal complement of range(A). Hence range$(AG) =$ range(A) and AG is the oblique projector onto range(A) along null$(AG) =$ null$(A(T_\pi A)^\dagger T_\pi)$. $\square$

From this theorem the following corollary is immediate:

Corollary 2.1 *If $x_0 \in range(A^T)$ and $b \in range(A)$, then iterates x_k generated by (2.22) converge to $A^\dagger b$. In particular, if A is nonsingular, $x_k \to A^{-1}b$.*

The same method of proof of Theorem (2.1) can also be applied to the new formulation of the block SSOR method to show:

Theorem 2.2 *Let $A \in \Re^{M\times N}$ be partitioned as in (2.2) with each A_i of full rank. Let $z_0 \in \Re^N$, $b \in \Re^M$ be arbitrary.*

(i) *For $\omega \in (0, 2)$, iterates z_k generated by (2.31) converge to*

$$z_\omega = P_{null(A^T)} z_0 + (I - V_\pi)^\dagger R(D + \omega L)^{-1} b. \tag{2.42}$$

(ii) *When $z_0 \in range(A)$, the corresponding limit for x_k given by (2.33) is $x_\omega = G_V b$, where*

$$G_V = A^T(D+\omega L)^{-T}R^T[R(D+\omega L)^{-1}AA^T(D+\omega L)^{-1}R^T]^\dagger R(D+\omega L)^{-1} \tag{2.43}$$

and $G_V A G_V = G_V$, $G_V A$ is the orthogonal projector onto $range(A^T)$, and AG_V is the oblique projector onto $range(A)$ along $null(AG_V)$.

(iii) *If $z_0 \in range(A)$ and $b \in range(A)$, then x_ω is the minimum norm solution $A^\dagger b$ to $Ax = b$. In particular, if A is nonsingular then $x_\omega = A^{-1}b$.*

2.3 Properties of Cimmino RP Methods

2.3.1 History of Cimmino Methods

The development of Cimmino forms of RP methods has closely paralleled that of the Kaczmarz forms. In the same way that Kaczmarz methods are analogous to a block SOR method applied to $AA^Ty = b$, $x = A^Ty$, Cimmino methods are analogous to a block Jacobi method applied to the same system. The similarities extend to the rates of convergence, and explicitly for the $m = 2$ case and numerically for test problems the Cimmino methods can be shown to have a less favorable spectrum than the Kaczmarz methods. Note that this says more than classical results about Jacobi and SOR methods, which only show a relationship between the spectral radii of the iteration matrices, and only for special test problems such as Poisson's equation. Like Jacobi methods, however, Cimmino methods have potential parallelism that can be exploited by multiprocessor machines.

G. Cimmino [12] in 1939 first proposed the iteration

$$x_{k+1} = Q_\sigma x_k + b_\sigma \tag{2.44}$$

with

$$\begin{array}{rcl} Q_\sigma & = & I - 2\sum_{i=1}^m \omega_i P_i \\ b_\sigma & = & 2\sum_{i=1}^m \omega_i \hat{b}_i = \sum_{i=1}^m \omega_i (A_i^T)^\dagger b_i \\ \sum_{i=1}^m \omega_i & = & 1, \quad \omega_i > 0 \end{array} \tag{2.45}$$

He proved convergence for square systems ($M = N$) with the single row $m = M$ partitioning provided that the system is consistent and $\text{rank}(A) > 1$.

Apparently Cimmino forms were not examined directly again until P. Gilbert [24] in 1972 independently proposed a Cimmino form with $m = M$ and $\omega_i = \omega$ as an alternative to the Kaczmarz type ART methods for image reconstruction problems. Calling the approach the simultaneous iterative reconstruction technique (SIRT), Gilbert showed that it produced superior reconstructions to ART on some test problems, but did not establish convergence. It should be noted that Gilbert's algorithm was specialized for his application and took into account nonnegativity constraints on the solution.

A. Kydes and R. Tewarson [41] analyzed in 1975 a Cimmino method with $m = 2$; they considered only the case $\omega_1 = \omega_2 = 1/2$ because (as is shown in Section 2.4) these are the optimal values. They also numerically tested the method on small ($N = 6$) matrices and demonstrated robustness for the method.

In 1979 A. Lakshminarayanan and A. Lent [42] examined an unconstrained Cimmino type RP method and showed that for the $m = M$ case with $\omega_i = \omega$ it is equivalent to Richardson's method applied to the normal equations $A^T Ax = A^T b$. They noted that T. Whitney and R. Meany [62] showed convergence to $A^\dagger b$ if $x_0 \in \text{range}(A^T)$ and they also specialized the results to image reconstruction problems, using the above results to show convergence to a weighted least squares solution.

Block forms of Cimmino type methods first appear in T. Elfving's 1977 technical report [17], where he considered the block Jacobi method for equations (2.12). By introducing a parameter ω, he derived an algorithm equivalent to a Cimmino method with $\omega_i = \omega$. For $x_0 \in \text{range}(A^T)$ and $b \in \text{range}(A)$, Elfving proved x_k given by (2.44) converges to $A^\dagger b$ iff $0 < \omega < 1/\rho(P_1 + P_2 + \cdots + P_m)$. Elfving also noted in [18] that his proof shows convergence when the weights depend on i and $\omega_i > 0$, $\sum_{i=1}^m \omega_i = 1$. He incorrectly stated that the convergence result holds when the weights depend on the iteration number k as well as i.

Returning to the single row $m = M$ case, R. Ansorge [2] in 1984 modified the Cimmino algorithm so that the convex combination of the reflections through the affine sets H_i (see Section 2.1) also included the point x_k, allowing the removal of the restriction that $\text{rank}(A) > 1$. He also proposed letting the weights vary from one iteration to the next by setting the weight attached to x_k to be ω_0 and the remaining weights as

$$\omega_i^{(k)} = (1-\omega_0)|r_i^{(k)}|^\gamma \left[\sum_{i=1}^{m} |r_i^{(k)}|^\gamma\right]^{-1} \tag{2.46}$$

where $\gamma > 0$ and $r_i^{(k)} = b_i - a_i^T x_k$. Ansorge showed that in the consistent case this scheme converges to the same solution as Tanabe showed for Kaczmarz's method. Although the title of Ansorge's paper promises ' Connections between the Cimmino-methods and the Kaczmarz-methods for the solution of singular and regular systems of equations', in fact he only showed the theoretical robustness that these methods share. The underlying relationship of these two approaches is shown later in this thesis.

2.3.2 Symmetrization of Cimmino Forms

Because each P_i in (2.44) is an orthogonal projector and hence symmetric, so is the iteration matrix Q_σ. Note that this holds without requiring a backward sweep through A, as the Kaczmarz forms do.

2.3.3 Convergence of Cimmino RP Methods

The following theorem extends the results from Section 2.3.1.

Theorem 2.3 *Let $A \in \Re^{M\times N}$ be partitioned as in (2.2). Let $x_0 \in \Re^N$, $b \in \Re^M$ be arbitrary, and let x_k be generated by iteration (2.44). For given values of ω_i $(i = 1, 2, \ldots, m)$, let*

$$C_\omega = 2[\omega_1(A_1^T)^\dagger, \omega_2(A_2^T)^\dagger, \ldots, \omega_m(A_m^T)^\dagger] \tag{2.47}$$

so that $Q_\sigma = I - C_\omega A$.

(i) *If $\omega_i \geq 0$ and $\sum_{i=1}^m \omega_i \leq 1$, then $\sigma(C_\omega A) \subseteq [0, 2]$ and x_k converges to*

$$x_\omega = P_{null(Q_\sigma)} x_0 + Q_\sigma^\dagger C_\omega b \tag{2.48}$$

(ii) *If $\omega_i \neq 0$ and A is nonsingular, then C_ω is nonsingular.*

(iii) *If $\omega_i > 0$, $\sum_{i=1}^m \omega_i < 1$, and A is nonsingular, then x_k converges to $A^{-1}b$.*

Proof:

(i) $C_\omega A$ is symmetric as noted in the last section. Since $\sigma(P_i) \subseteq \{0,1\}$,

$$\begin{array}{rcl} \lambda_{\min}(C_\omega A) & \geq & 2\sum_{i=1}^m \omega_i \lambda_{\min}(P_i) \geq 0 \\ \lambda_{\max}(C_\omega A) & \leq & 2\sum_{i=1}^m \omega_i \lambda_{\max}(P_i) \leq 2\sum_{i=1}^m \omega_i \leq 2 \end{array} \tag{2.49}$$

Also, since $C_\omega A$ is symmetric, range$(I-C_\omega A)$ and nuull$(I-C_\omega A)$ are complementary subspaces and so $I - C_\omega A$ is semiconvergent. The characterization of semiconvergent iterative processes in [4, p.198] applies again to give

$$\begin{array}{rcl} x_k & \rightarrow & [I-(I-C_\omega A)(I-C_\omega A)^\dagger]x_0 + (I-C_\omega A)^\dagger C_\omega b \\ & = & [I - Q_\sigma Q_\sigma^\dagger]x_0 + Q_\sigma^\dagger C_\omega b \\ & = & P_{null(Q_\sigma)}x_0 + Q_\sigma^\dagger C_\omega b \end{array} \tag{2.50}$$

(ii) Suppose that for some $x \neq 0$,

$$0 = C_\omega x = 2\sum_{i=1}^m \omega_i A_i A_i^\dagger x. \tag{2.51}$$

Then for $y^T = (y_1^T, y_2^T, \ldots, y_m^T)$ where $y_i = \omega_i A_i^\dagger x$, $\sum_{i=1}^m A_i y_i = A^T y = 0$. Note that this y is nonzero because each ω_i is nonzero and if $x \in$ null$(A_i^\dagger)$ for each i, then $x \in$ null(A_i^T) for each i, implying $Ax = 0$ and contradicting the nonsingularity of A. But A^T is also nonsingular, so $A^T y = 0$ also leads to a contradiction. Hence C_ω must also be nonsingular.

(iii) Since (ii) shows that both A and C_ω are nonsingular, 0 is not in $\sigma(C_\omega A)$. Furthermore, if $\sum_{i=1}^m \omega_i < 1$, then the last inequality in the proof of (i) is strict and so 2 is not in $\sigma(C_\omega A)$. From (i) then $\rho(I - C_\omega A) = \rho(Q_\sigma) < 1$ and x_k must converge to

$$(C_\omega A)^{-1} C_\omega b = A^{-1} b, \tag{2.52}$$

completing the proof. □

It can also be shown that null$(I-C_\omega A) \subseteq$ range(A^T) and so the projection in (i) is onto a space contained in range(A^T). Also, (ii) shows that C_ω is a preconditioner for the original system, which is nonsingular when A is and the weights are nonzero.

2.4 The Two Projector Case

When the matrix A is partitioned into two block rows a complete analysis of RP methods is possible using the concept of the angles θ_k between the two subspaces $L_i = \mathrm{range}(A_i)$, $i = 1, 2$. The first modern description of the angles θ_k appears in [1]; the definition presented here follows that of A. Björck and G. Golub in [7]. The smallest angle θ_1 between L_1 and L_2 is defined as the angle in $[0, \pi/2]$ satisfying

$$\begin{aligned} &\cos\theta_1 = \max_{u \in L_1} \max_{v \in L_2} u^T v \\ &\text{subject to } \; \| u \| = \| v \| = 1 \end{aligned} \tag{2.53}$$

Let u_1 and v_1 be the attainment vectors; then for $k = 2, 3, \ldots, N/2$ the remaining angles between the two subspaces are defined as

$$\begin{aligned} &\cos\theta_k = \max_{u \in L_1} \max_{v \in L_2} u^T v \\ &\text{subject to} \left\{ \begin{array}{rcl} \| u \| & = & \| v \| = 1 \\ u_i^T u & = & v_i^T v = 0, \quad i = 1, 2, \ldots, k-1 \end{array} \right. \end{aligned} \tag{2.54}$$

Furthermore, when u_i and v_j are defined as above, $u_i^T v_j = 0$, $i \neq j$ can also be shown.

From (2.53) and (2.54) the CS (Cosine-Sine) decomposition [56, 26] can be obtained, which is stated below in terms of the projectors P_i defined previously.

Theorem 2.4 (CS Decomposition) *Let $P_i \in \Re^{N\times N}$ be the orthogonal projectors onto the subspaces L_i, for $i = 1, 2$. Then there are orthogonal matrices U and V such that*

$$P_1 = U \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} U^T , \quad P_2 = V \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} V^T, \tag{2.55}$$

and

$$U^T V = \begin{bmatrix} C & -S \\ S & C \end{bmatrix}, \tag{2.56}$$

$$\begin{array}{rcl} C & = & diag(c_1, c_2, \ldots, c_{N/2}) \\ S & = & diag(s_1, s_2, \ldots, s_{N/2}) \\ I & = & C^2 + S^2 \\ 1 & \geq & c_1 \geq c_2 \geq \cdots \geq c_{N/2} \geq 0. \end{array} \tag{2.57}$$

In the theorem $c_k = \cos\theta_k$ and $s_k = \sin\theta_k$, where the angles θ_k are defined above. Now consider the unsymmetric Kaczmarz iteration

$$x_{k+1} = Q_u x_k + b_\pi = (I - \omega_1 P_1)(I - \omega_2 P_2)x_k + b_\pi .$$

Letting $\alpha_i = 1 - \omega_i$, and substituting in the expressions for P_1 and P_2 from the CS decomposition gives

$$\begin{array}{rcl} Q_u & = & U \begin{bmatrix} \alpha_1 I & 0 \\ 0 & I \end{bmatrix} U^T V \begin{bmatrix} \alpha_2 I & 0 \\ 0 & I \end{bmatrix} V^T \\ & = & U \begin{bmatrix} \alpha_1 I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} C & -S \\ S & C \end{bmatrix} \begin{bmatrix} \alpha_2 I & 0 \\ 0 & I \end{bmatrix} V^T \end{array} \tag{2.58}$$

$$= U \begin{bmatrix} \alpha_1\alpha_2 C & -\alpha_1 S \\ \alpha_2 S & C \end{bmatrix} V^T \tag{2.59}$$

Applying a similiarity transformation using the matrix V^T gives

$$\begin{array}{rcl} V^T Q_u V & = & V^T U \begin{bmatrix} \alpha_1\alpha_2 C & -\alpha_1 S \\ \alpha_2 S & C \end{bmatrix} V^T V \\ & = & \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} \alpha_1\alpha_2 C & -\alpha_1 S \\ \alpha_2 S & C \end{bmatrix} \\ & = & \begin{bmatrix} \alpha_1\alpha_2 C^2 + \alpha_2 S^2 & (1-\alpha_1) CS \\ \alpha_2(1-\alpha_1) CS & C^2 + \alpha_1 S^2 \end{bmatrix} \end{array} \tag{2.60}$$

Since each block in the above 2×2 matrix is diagonal, $V^T Q_u V$ has the same eigenvalues as its scalar 2×2 principal submatrices. These eigenvalues are given by

$$(1+\alpha_1\alpha_2)c_i^2+(\alpha_1+\alpha_2)s_i^2 \pm \left\{[(\alpha_1\alpha_2-1)c_i^2+(\alpha_1-\alpha_2)s_i^2]^2+4\alpha_2(1-\alpha_1^2)c_i^2 s_i^2\right\}^{1/2} \tag{2.61}$$

for $i = 1, 2, \ldots, N/2$. When $\alpha_1 = \alpha_2 = \alpha$, these eigenvalues become

$$(1-\alpha)^2 c_i^2 + 2\alpha \pm |1-\alpha| c_i \sqrt{(1-\alpha)^2 c_i^2 + 4\alpha} \tag{2.62}$$

For a given α the modulus of this expression is a maximum when c_i is largest, i.e., when $c_i = c_1$. To examine the spectral radius of Q_u then it suffices to consider $c_i = c_1$ and take the positive sign in (2.62). Doing so and minimizing with respect to α gives

$$\omega = 1 - \alpha = 2/(1 + s_1) \tag{2.63}$$

and a spectral radius of $(1-s_1)/(1+s_1)$. The same result was derived in [18] using classical SOR theory. The advantage of using the CS decomposition is that (2.61) gives the full spectrum of Q_u and not simply the eigenvalue of largest modulus.

Now note that when $\omega = 1$ the eigenvalues of the nonsymmetric Kaczmarz iteration matrix Q_u become

$$\left\{c_1^2, c_2^2, \ldots, c_{N/2}^2, 0\right\} \tag{2.64}$$

with 0 repeated $N/2$ times. Applying the CS decomposition to the symmetrized Kaczmarz iteration matrix Q_π from (2.22) with $\omega_i = \omega$ gives

$$\begin{aligned} Q_\pi &= (I-\omega P_1)(I-\omega P_2)(I-\omega P_1) \\ &= U \begin{bmatrix} \alpha^2(\alpha C^2 + S^2) & \alpha(\alpha-1)CS \\ \alpha(\alpha-1)CS & \alpha S^2 + C^2 \end{bmatrix} U^T \end{aligned} \tag{2.65}$$

where again $\alpha = 1 - \omega$. When $\omega = 1$, the eigenvalues of the symmetrized matrix are identical to those of the unsymmetrized matrix Q_u. Although this result does not hold when $m > 2$ because in general Q_u has complex eigenvalues, it does indicate that the symmetrization process does not have as deleterious an effect on the spectrum as, e.g., forming the normal equations

$A^T A$ has on the spectrum of A. One objection to the symmetrization process is that Q_π requires three projections while Q_u only needs two. However, when $\omega = 1$ and $x_0 = 0$ are chosen, the Kaczmarz iteration (2.22) can be shown to keep each $x_k \in \text{range}(I - P_1)$ and so can be implemented with only two projections per iteration. As is shown in Chapter 5, Chebyshev or CG acceleration can be implemented in this case also using only two projections per iteration.

The value $\omega = 1$ for Q_π has two other special properties when $m = 2$. First, it minimizes the spectral radius of Q_π as was shown theoretically in [39] and verified numerically in [38]. Second, half of the eigenvalues are exactly zero, corresponding to eigenvectors in range(P_1). Since CG acceleration requires in exact arithmetic a number of iterations equal to the number of distinct eigenvalues this implies only half the number of iterations are needed compared to when $\omega \neq 1$. This property carries over to general m; when $\omega = 1$ at least $(m-1)/m$ of the eigenvalues of Q_π are exactly zero.

The CS decomposition can also be applied to the Cimmino iteration $x_{k+1} = Q_\sigma x_k + b_\sigma$, where

$$Q_\sigma = I - 2(\omega_1 P_1 + \omega_2 P_2) \tag{2.66}$$

Proceeding as before gives

$$V \begin{bmatrix} (1-2\omega_2)I - \omega_1 C^2 & 2\omega_1 CS \\ 2\omega_1 CS & I - 2\omega_1 S^2 \end{bmatrix} V^T \tag{2.67}$$

which in turn has eigenvalues

$$(1 - \omega_1\omega_2) \pm \sqrt{[\omega_1(s_i^2 - c_i^2) - \omega_2]^2 + 4\omega_1^2 c_i^2 s_i^2} \tag{2.68}$$

Let $\beta = \omega_1 + \omega_2 =$ constant. Then the eigenvalues are

$$(1 - \beta) \pm \sqrt{4s_i^2(\omega_1^2 - \beta\omega_1) + \beta^2/4} \tag{2.69}$$

The modulus of this is minimized for every i when $\omega_1 = \beta/2$, in which case the eigenvalues are

$$1 - (1 \pm c_i)\beta \tag{2.70}$$

This in turn is minimized when $\beta = 1$, so $\omega_1 = \omega_2 = 1/2$ gives the minimal spectral radius of $c_1 = \cos\theta_1$ for Q_σ. The same result was obtained in

Method	$\omega_{\min}$	ρ	Eigenvalues
Unsymmetric Kaczmarz	$2/(1+\sin\theta_1)$	$(1-\sin\theta_1)/(1+\sin\theta_1)$	N
Symmetric Kaczmarz	1	$\cos^2\theta_1$	$N/2$
Cimmino	$1/2$	$\cos\theta_1$	N

Table 2.1: Comparison for $m = 2$ Case

[41] using a different approach, but again the advantage of using the CS decomposition is that the entire spectrum of Q_σ is known; viz.,

$$\sigma(Q_\sigma) = \left\{\pm c_1, \pm c_2, \ldots, \pm c_{N/2}\right\} \tag{2.71}$$

In particular, not only does the Cimmino iteration have half the rate of convergence of the Kaczmarz iteration because

$$\rho(Q_\sigma) = c_1 = \sqrt{\rho(Q_\pi)} \tag{2.72}$$

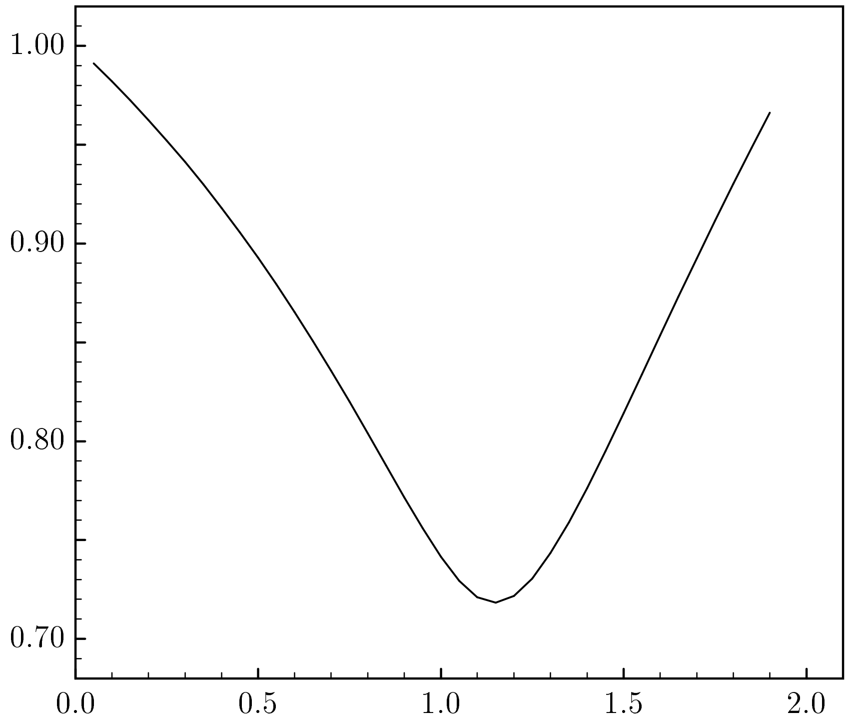
but when CG acceleration is applied twice the number of iterations are required. Table 2.1 summarizes some of these results.

Unfortunately no extension of the CS decompostion has been found for $m > 2$ that allows such a complete eigenvalue analysis of RP methods. Computational considerations require choosing $m > 2$ in many cases, and conjectures based on the two projector approach are often false. As one example, $\omega_1 = \omega_2 = \cdots = \omega_m = 1/m$ does not necessarily minimize the spectral radius of the Cimmino iteration matrix. As a more specific example, let

$$A = \begin{bmatrix} 2 & 1 & 8 \\ 0 & 8 & 8 \\ 5 & 8 & 5 \end{bmatrix} \tag{2.73}$$

When the Kaczmarz method (2.22) with $m = 3$ and $\omega_1 = \omega$ is applied to A, the value of ω that minimizes $\rho(Q_\pi)$ is larger than 1. Figure 2.3 shows the spectral radius as a function of ω for this example.

Figure 2.3: $\rho(Q_\pi)$ vs. ω for Example Problem



An important example is now presented showing that no direct relationship need exist between the spectra of A and its related RP matrices. Define

$$A = \begin{bmatrix} A_1^T \\ A_2^T \end{bmatrix} = \begin{bmatrix} 0 & D \\ S & C \end{bmatrix} \tag{2.74}$$

where each block is $N/2$, C and S satisfy the properties (2.57), and

$$D = diag(d_1, d_2, \ldots, d_{N/2}). \tag{2.75}$$

The projectors become

$$P_1 = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \quad P_2 = \begin{bmatrix} C \\ S \end{bmatrix} \begin{bmatrix} C & S \end{bmatrix} \tag{2.76}$$

and the eigenvalues of $I - Q_\pi$ are $\{s_1^2, s_2^2, \ldots, s_{N/2}^2, 1\}$ while those of A are

$$\left[c_i \pm \sqrt{c_i^2 + 4s_i d_i}\right]/2 \tag{2.77}$$

A is nonsingular iff each s_i and d_i is nonzero. If the s_i's are chosen to be close to 1 while the d_i's are chosen to be close to 0, $I - Q_\pi$ has eigenvalues clustered near 1 while A has many eigenvalues close to 0. Hence A is badly conditioned while $I - Q_\pi$ is very well-conditioned. Conversely if the d_i's are chosen to be large while the s_i's are close to 0 in such a way that $d_i s_i$ is near 1, then A is well conditioned while $I - Q_\pi$ is badly conditioned.

In summary, the distribution of eigenvalues of A and the induced RP system matrix may differ greatly. Although this example is contrived, among the test problems in the next chapter are two which yield identical spectra but have drastically differing spectra for their corresponding RP matrices.

2.5 Comparison of Kaczmarz and Cimmino Methods

A preliminary comparison of the product and sum RP methods can now be made. Both allow convergence for a wide choice of iteration parameters ω_i. When $\omega_i = \omega$ is fixed, Kaczmarz methods converge provided that $0 < \omega < 2$;

for Cimmino methods the condition becomes $0 < \omega < 1/\rho(P_1+P_2+\cdots+P_m)$. Since each P_i has norm 1, the last condition is satisfied if $0 < \omega < 1/m$.

Both approachs converge even in the inconsistent case where b is not in range(A). When $x_0 \in$ range(A^T), Kaczmarz methods converge to b premultiplied by a generalized inverse of A. Cimmino methods also converge in this case, but the limit is not so well characterized in terms of A. When the system is consistent, both classes converge to a minimum norm solution with no restrictions on A.

Now suppose that A is nonsingular. The simplest choice of iteration parameters becomes $\omega_i = 1$ for Kaczmarz methods and $\omega_i = 1/m$ for Cimmino methods. If $x_0 = 0$ is chosen, every iterate x_k generated by Kaczmarz methods satisfies the first block of equations exactly: $A_1^T x_k = b_1$. No such property holds for the Cimmino method. Furthermore, when A is an orthogonal matrix the Kaczmarz iteration matrix is 0 and convergence takes one step. The Cimmino iteration has a linear convergence rate for orthogonal matrices since $Q_\sigma = (1-2/m)I$. This last disability, however, disappears if an acceleration method is to be used and the system $(I - Q_\sigma)x = b_\sigma$ is considered instead because in this case the system becomes $(2/m)x = (2/m)A^{-1}b$.

When $m = 2$ both approaches can be completely analyzed in terms of the angles between range(A_1) and range(A_2). The Cimmino method has half the convergence rate of Kaczmarz, and the Cimmino system matrix has twice as many distinct eigenvalues.

Cimmino methods have two advantages over Kaczmarz methods. Their iteration matrices are automatically symmetric, while Kaczmarz methods require two sweeps through A to be symmetrized. Secondly, the projections in Cimmino forms can be computed in parallel and then added later, while the projections in the Kaczmarz forms must be computed one after another. As is shown in the next chapter, however, for structured matrices each individual projection can be computed with parallelism, allowing Kaczmarz forms to be implemented efficiently on multiprocessor and even vector computers.

In general then, Kaczmarz methods have better spectral and convergence properties while Cimmino methods have more natural parallelism in their computation.

Chapter 3

Implementation of Row Projection Methods

The convergence results in Chapter 2 allow any row partitioning of A to be used. The criteria for selecting a row partitioning can thus be based on computational considerations. This chapter outlines those criteria, defines and describes a set of test problems drawn from the important class of elliptic partial differential equations in three dimensions, and presents two row partitioning schemes satisfying the criteria for this class of problems. Furthermore, the approach to row partitioning is shown to be extensible to more general problems.

Although A. Björck and T. Elfving in [6] tested an accelerated Kaczmarz method, their work concentrated on the single row ($m = N$) case applied to weighted least squares problems. Previous work dealing with the practical implementation of block forms was by C. Kamath [38] and by C. Kamath and A. Sameh [39] in 1986. Using example problems drawn from two dimensional elliptic partial differential equations, they examined the issues of suitable row partitionings and methods for the numerical solution of the induced projections, primarily for the $m = 2$ case. They also proved and numerically verified that the optimal weighting parameter ω in this case is 1. By comparisons with Krylov subspace methods and CG applied to the normal equations they showed that the RP algorithm selected was more robust; unfortunately it was not as efficient as the other methods. This drawback was circumvented in [9] by using a preconditioned CG method for computing the projections; this approach in turn has the disagreeable property of duplicating work since the projections must be computed on every RP iteration. This chapter presents for three dimensional elliptic partial differential equations an implementation that uses only $\mathcal{O}(N)$ storage, provides a direct solver for computing the projections, and is both more robust and more efficient than other iterative solvers. Furthermore, the implementation is provably stable and allows efficient numerical monitoring of the conditioning of the least squares subproblems that define the projectors.

3.1 Row Partitioning Goals

The first criterion for a row partitioning is that the projections $P_i x = A_i(A_i^T A_i)^{-1} A_i^T x$ must be efficiently computable, since m or $2m-1$ projections must be found on every iteration. One way to achieve this is through parallelism; if A_i^T is the direct sum of separate blocks $C_i^{(j)}$, then P_i is block diagonal. The computation of $P_i x$ can then be done by assigning each block of P_i to a different processor on a multiprocessor machine, or by vectorizing the computations across the blocks on a vector machine.

The second criterion is storage efficiency. For example, in the general case actually forming each matrix P_i leads to storing m dense $N \times N$ matrices, far exceeding the requirements of a direct solver applied to A. The rule followed here is that the additional storage should be $\mathcal{O}(N)$, that is, a few extra vectors, the number of which must not grow with increasing problem size N. Another aspect of storage efficiency, *data locality*, is not as easy to quantify. Roughly speaking this means that if a block of data is accessed and placed on a small, fast memory such as a cache, as much work as possible should be done with that data before it is returned to the larger, slower memory medium. For vector machines this usually means that data placed in vector registers should be used as much as possible before being written back to memory.

The third criterion is some bound on the condition number of the subproblems induced by the projections. Computing $y = P_i x$ requires solving a system of the form

$$A_i^T A_i v = w \tag{3.1}$$

Hence the accuracy with which P_i can be computed depends on $\kappa(A_i^T A_i)$ and clearly RP methods can be unstable if large errors occur in the projections (this is verified and made more precise in Chapter 4.) Preferrably an *a priori* bound should be available for $\kappa(A_i^T A_i)$. Equally desirable is a practical and cheap numerical way of bounding $\kappa(A_i^T A_i)$ for a given problem. Previous work by C. Kamath and A. Sameh [39] on RP methods applied to two dimensional problems did not provide these bounds on the condition of the subproblems.

The fourth criterion is that the number m of projectors should be kept as small as possible, and should not depend on N. A full explanation of this must be deferred to Chapter 6, but for Kaczmarz methods one reason was presented in Section 2.4: N/m eigenvalues of the iteration matrix are exactly 0 when $\omega_i = 1$. Increasing m decreases the number of zero eigenvalues. For Cimmino methods the main reason follows from the form of the iteration matrix: m vectors must be summed on every iteration.

In summary, the row partitioning should allow efficiency in the computations, require at most $\mathcal{O}(N)$ storage, give well conditioned subproblems, and have m a small constant. These goals can be conflicting. Choosing $m = 1$ satisfies the fourth but violates the first and third since computing the projection is equivalent to solving the normal equations $A^T Ax = A^T b$. Choosing $m = N$ satisfies the second and third requirements since $A_i^T A_i$ is a scalar and hence perfectly conditioned and cheaply stored, but the first and fourth criteria are violated. One of the most surprising and useful results of this thesis is that all four goals can be achieved simultaneously for important classes of problems.

3.2 Test Problems

The test problems arise from using the seven point centered difference operator on elliptic partial differential equations

$$au_{xx} + bu_{yy} + cu_{zz} + du_x + eu_y + fu_z + gu = F \tag{3.2}$$

where $a - g$ are functions of (x, y, z) and the domain is the unit cube $[0,1] \times [0,1] \times [0,1]$. Dirichlet boundary conditions are imposed in a manner described later. For ellipticity $a, b, c > 0$ on the unit cube is required, but this is not necessary for the application of RP methods to (3.2). When discretized using $n+1$ grid points in each direction the resulting system has a coefficient matrix A of order $N = n^3$ and with a nonzero structure shown in Figure 3.1 for $n = 6$.

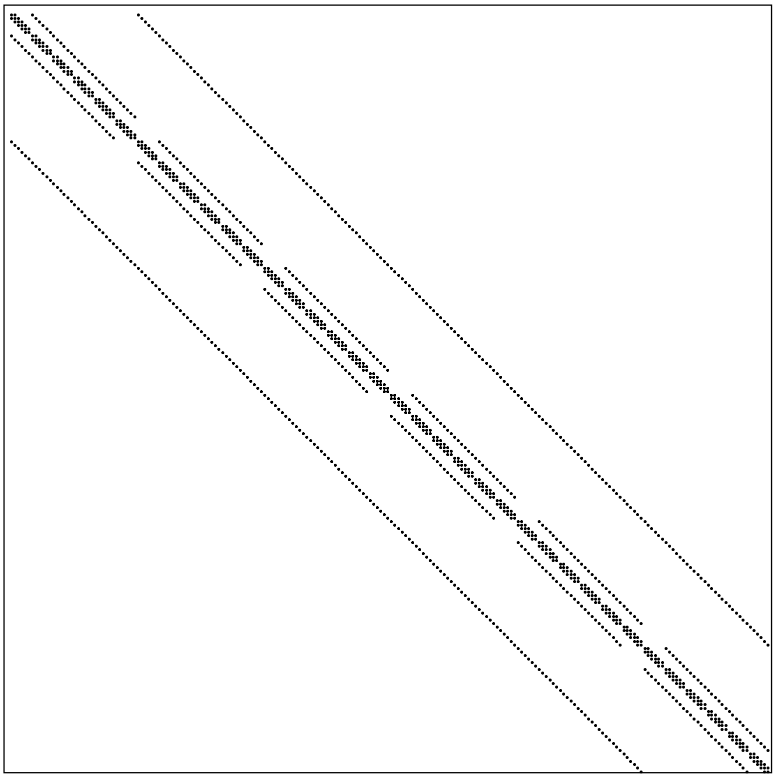


Figure 3.1: Nonzero Structure of the Test Matrices

This class of test problems is chosen for four reasons. First, by refining the mesh the problem size grows rapidly with n, allowing test problems that can be immediately scaled up to realistic sizes. Results are shown for problems of up to order 216000, corresponding to a $60\times 60\times 60$ mesh. Second, the class of problems of the form (3.2) include many important applications such as computational fluid dynamics and structural mechanics. Third, by manipulating the coefficients functions of the partial differential equation, matrices A can be created with any desired properties or lack of properties such as diagonal dominance, definite symmetric part, eigenvalues on both sides of the imaginary axis, or extreme ill conditioning. Fourth, the problems provide a specific example on how to select m, partition the rows into blocks, and analyze the subproblems that define the projectors.

In spite of the special form chosen for the test problems, the scheme presented for choosing the row partitioning also works for problems on irregular domains, 27 point difference operators, and mixed boundary conditions. With only slight modifications it can also be applied to finite element equations, coupled systems of partial differential equations, some periodic boundary conditions, and other problems.

Each test problem has boundary conditions chosen to give a predetermined solution to the partial differential equation, so that the norm of the error vector as well as that of the residual vector can be checked. This means in practice that A is created using finite differences, and then a solution vector x^* is multiplied by A to obtain the right hand side b. An objection to this process is that the contribution of the eigenvectors corresponding to eigenvalues near 0 are automatically damped out from b, creating a system that is unnaturally easier to solve. In the extreme case when A is singular, the system $Ax = b$ is guaranteed consistent. The process is acceptable here, however, for two reasons. First, RP methods work not on the system $Ax = b$, but on a derived system $(I - Q)x = \hat{b}$. The example in Section 2.4 shows that the spectra of these two systems can be totally unrelated; later this observation is demonstrated to actually hold for test problems 1 and 2. The second reason is that all of the other methods compared with RP algorithms work on $Ax = b$, so if anything the process gives an advantage to those other methods.

The collection of test problems and their predetermined solutions follows:

Problem 1:

$$u_{xx} + u_{yy} + u_{zz} + \beta u_x = F \tag{3.3}$$

with solution

$$u = xyz(1-x)(1-y)(1-z)$$

This is a prototype convection-diffusion equation with constant coefficients (c.f. [40].) The parameter β controls the degree of nonsymmetry of the system, and for large values of β the system loses the diagonal dominance which is important for many iterative methds. For most of the testing $\beta = 1000$ is used.

Problem 2:

$$u_{xx} + u_{yy} + u_{zz} + \beta u_y = F \tag{3.4}$$

with solution

$$u = xyz(1-x)(1-y)(1-z)$$

This is the same as (3.3) but with the convection term in the y-direction. Row projection methods can treat the variables x, y, and z anisotropically, and this problem helps demonstrate how to choose the partitioning of the matrix in order to enhance the performance of the methods. The spectrum of the matrix A is identical to that of (3.3), but the induced row projection matrices have different spectra.

Problem 3:

$$u_{xx} + u_{yy} + u_{zz} + 1000\exp(xyz)(u_x + u_y - u_z) = F \tag{3.5}$$

with solution

$$u = x + y + z$$

This is a three dimensional version of the two dimensional problem 4 in [39], for which Krylov subspace methods fails but the symmetrized RP method succeed.

Problem 4:

$$\begin{aligned} (\exp(xyz)u_x)_x &+ (\exp(-xyz)u_y)_y + (\exp(-xyz)u_z)_z \\ &- 250(x+y+z)u_x - 250[(x+y+z)u]_x - u/(1+x+y+z) = F \end{aligned} \tag{3.6}$$

with solution

$$u = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$$

This is a three dimensional analogue to the problem tested in [20], and can be proven to have a positive definite symmetric part $(A + A^T)/2$ for any mesh size, which is sufficient for the proof of convergence of some CG-like methods for nonsymmetric systems. It should be noted that even though the condition is sufficient, it is not necessary as the comparative testing with GMRES(k) in Chapter 6 shows.

Problem 5:

$$u_{xx} + u_{yy} + u_{zz} + 100xu_x - yu_y + zu_z + 100(x+y+z)u/xyz = F \tag{3.7}$$

with solution

$$u = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$$

This problem (c.f. [53]) has a pole of degree 2 at the origin and singularities along each coordinate plane in the u-term coefficient. The resulting system has an indefinite symmetric part and is extremely ill–conditioned, features that worsen as the mesh size gets smaller.

Problem 6:

$$u_{xx} + u_{yy} + u_{zz} - 10^5 x^2 (u_x + u_y + u_z) = F \tag{3.8}$$

with solution

$$u = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$$

This and the following problems are similar to the convection-diffusion type problems examined in [19]

Problem 7:

$$u_{xx} + u_{yy} + u_{zz} - 1000(1+x^2)u_x + 100(u_y + u_z) = F \tag{3.9}$$

with solution

$$u = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$$

Problem 8:

$$u_{xx} + u_{yy} + u_{zz} - 1000x^2 u_x + 1000u = F \tag{3.10}$$

with solution

$$u = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$$

Problem 9:

$$u_{xx} + u_{yy} + u_{zz} - 1000\left[(1-2x)u_x + (1-2y)u_y + (1-2z)u_z\right] = F \tag{3.11}$$

with solution

$$u = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$$

Problem	$\mathrm{Re}(\lambda) > 0$	$A + A^T$ Definite	$\kappa(A)$
1	Yes	Yes	9
2	Yes	Yes	9
3	Yes	No	57
4	Yes	Yes	28
5	No	No	40000
6	Yes	No	4786
7	Yes	No	36
8	No	No	6620
9	Yes	No	77

Table 3.1: Properties of A for $N = 1728$

These test problems exhibit many of the properties that cause difficulties for nonsymmetric iterative solvers. The eigenvalues of each test problem on a $12 \times 12 \times 12$ grid are shown in Appendix A. The properties are summarized in Table 3.1. 'Re $(\lambda) > 0$' indicates whether or not the eigenvalues lie in the right half plane of the complex plane; if this does not hold then residual polynomial methods will fail. '$A + A^T$ definite' indicates whether or not the symmetric part of A is definite; if the answer is yes then the sufficient conditions for GMRES(k) to converge hold. $\kappa(A)$ is an estimate of the condition number obtained from the eigenvalues of $A^T A$. Even though the properties are shown only for one relatively small problem size, many of them continue to hold for larger N. In particular, problems 5 and 8 have condition numbers that become dramatically worse with increasing N.

3.3 Row Partitioning of Test Problems

Row partitionings that give block diagonal projectors and hence achieve parallelism are best derived by considering the computational mesh. For convenience assume that n is a multiple of 3. For a seven point difference operator any two nodes separated by at least two other nodes on the line connecting them have computational molecules that do not intersect and so the corresponding rows of A have no nonzero entries in a shared column. If those two rows comprise the block row A_i^T, computing $P_i x$ can be done by finding the projection of x onto the range space of each row simultaneously. This is a three dimensional analogue of the approach to row partitioning proposed and implemented in [38], and the method below is a grid motivated version of a matrix partitioning used in [39].

Extending the idea, let A_1^T be those rows of A corresponding to nodes numbered $1, 4, 7, \ldots, n-2$ on the lines numbered $1, 4, 7, \ldots, n-2$ on the planes numbered $1, 4, 7, \ldots,\ n-2$. Here the natural ordering of nodes in the mesh is assumed. Next, let A_2^T be those rows of A corresponding to nodes numbered $2, 5, 8, \ldots, n-1$ on the lines numbered $1, 4, 7, \ldots, n-2$ on the planes numbered $1, 4, 7, \ldots, n-2$. Continuing, this ordering groups every third node on every third line in every third plane together to form A_i^T, yielding $m = 27$ block rows. Figure 3.2 shows the assignment of nodes to block rows A_i^T, with each node marked by the assigned i.

A_i^T consists of $n^3/27$ rows, and computing $P_i x$ can be done as $n^3/27$ simultaneous tasks since each row in A_i^T is orthogonal to all of the other rows in A_i^T.

This scheme is equivalent to the original Kaczmarz ($m = N$) approach applied to a row-permuted A; the special structure of A allows the row partitioning to be viewed as an $m = 27$ approach. All four of the criteria of Section 3.1 are satisfied. In particular $y = P_i x$ is computed as $n^3/27$ operations $y = (a_i^T a_i)^{-1}(a_i^T x)a_i$, which can be done efficiently if $\gamma_i = (a_i^T a_i)^{-1}$ is computed in a preprocessing stage before the iterations begin. The additional storage incurred is one vector, $g = (\gamma_1, \gamma_2, \ldots, \gamma_N)^T$.

This row partitioning approach can be generalized by using lines in the mesh rather than nodes, that is, A_i^T is comprised of equations corresponding to every third line in every third plane. This gives $m = 9$ block rows, with equations assigned to block rows as shown in Figure 3.3.

Each A_i^T is comprised of $n^2/9$ separate subblocks of n coupled equations each, so P_i can be computed as $n^2/9$ separate tasks. This satisfies the requirements of parallelism and small m for row partitionings. What of the other criteria? Let the subblocks of A_i^T be denoted $(C_i^{(j)})^T$ for $j = 1, 2, \ldots, n^2/9$. Dropping the superscripts and subscripts temporarily, a typical $(C_i^{(j)})^T$ has the form

$$C^T = [0, D_1, 0, D_2, T, D_3, 0, D_4, 0] \tag{3.12}$$

with each D_k a diagonal matrix and T a tridiagonal matrix. The matrix

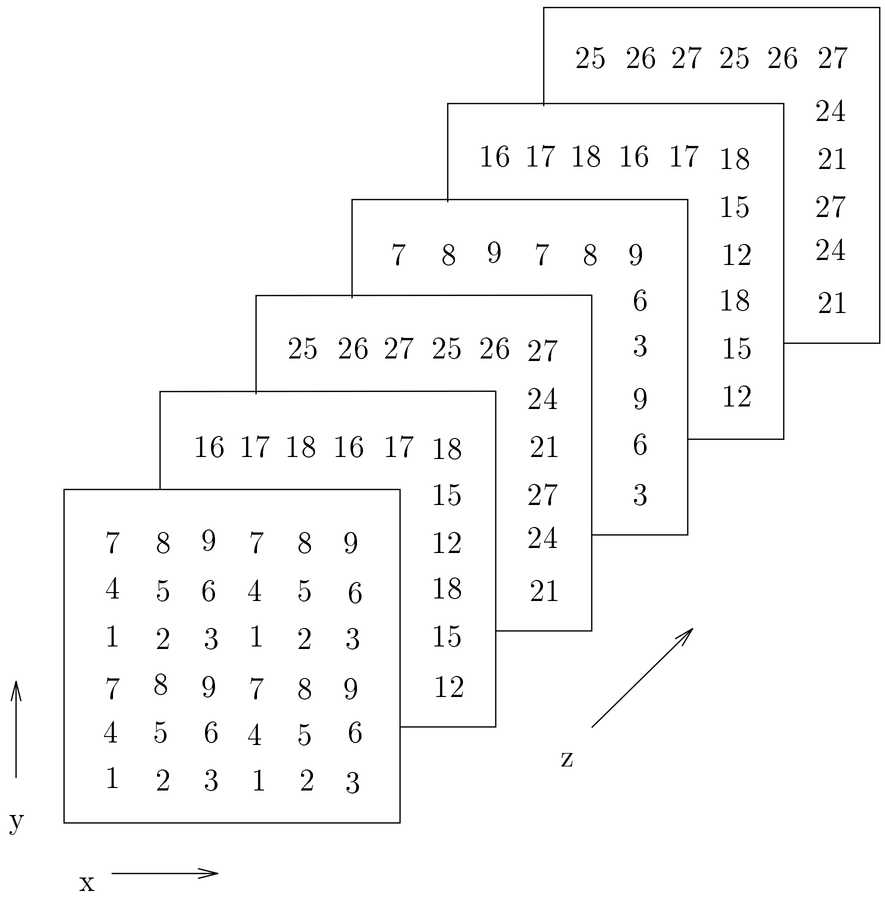


Figure 3.2: Row Partitioning by Nodes: $m = 27$

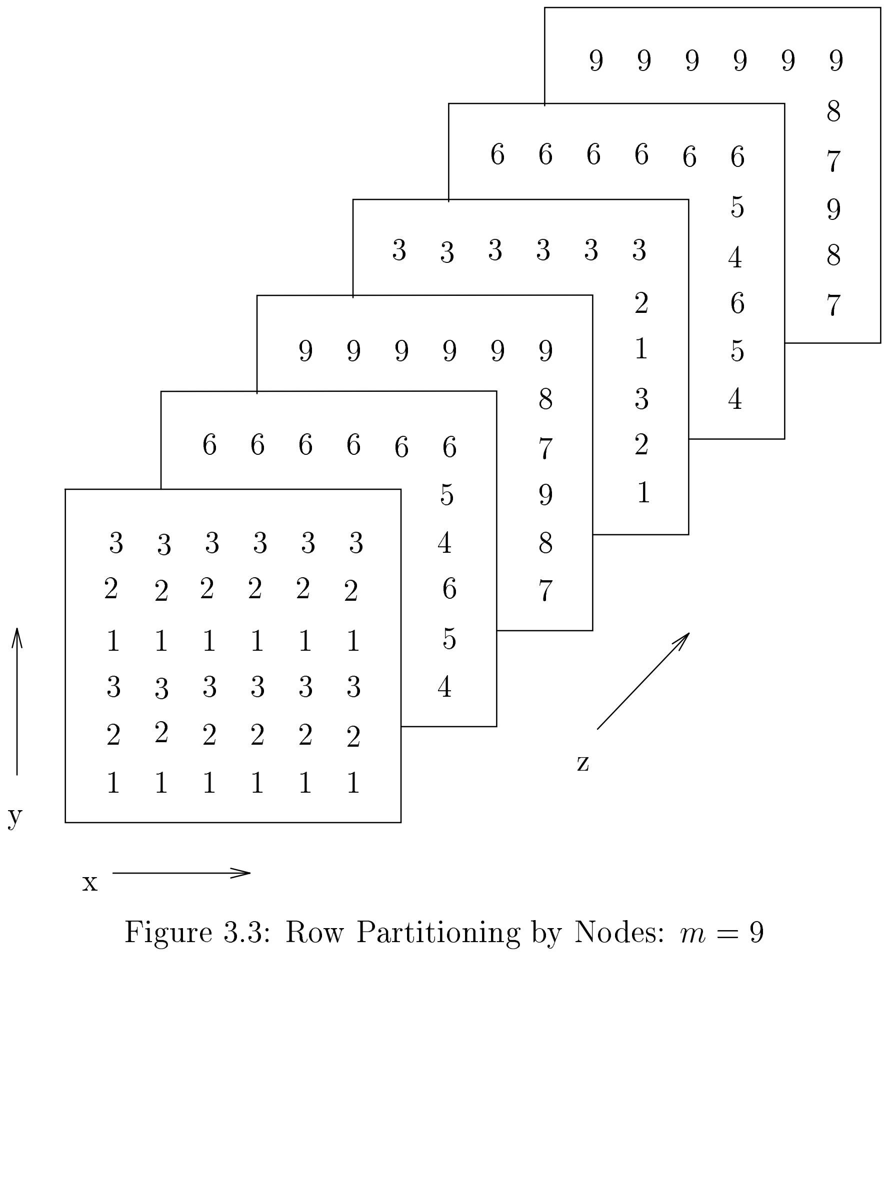


Figure 3.3: Row Partitioning by Nodes: $m = 9$

C^TC is positive definite because when A is nonsingular each subblock is necessarily of full rank n. More importantly, C^TC is pentdiagonal and hence narrow banded. This means that its Cholesky factor R has upper bandwidth of 2, and the Cholesky factors for all of the subblocks for all of the block rows A_i^T can be stored using only 3 additional vectors. Computing $y = P_i x$ can then be performed with operations of the form

$$y = C(R^TR)^{-1}C^Tx \tag{3.13}$$

Summarizing, the row partitioning with $m = 9$ satisfies all of the criteria with the possible exception of the third requirement that each subproblem be well-conditioned. Intuitively each C (and thus each A_i^T) is well-conditioned because C^TC consists of the normal equations of T in (3.12), with the squares of the diagonal blocks D_i added to the main diagonal of T^TT, making it strongly diagonally dominant. Since the singular values of C are the square roots of the eigenvalues of C^TC, C can be expected to be well-conditioned. A rigorous proof of this in terms of the coefficient functions of the partial differential equation (3.2), along with practical numerical methods for checking the conditioning of the subproblems, is provided in Chapter 4.

The next step in the row partitioning approach is to consider a planar version, placing all of the nodes in every third plane into the same block A_i^T. This yields $m = 3$ block rows A_i^T, each consisting of $n/3$ subblocks of n^2 coupled equations. Now difficulties arise; the typical C^T has the form $[0, D_1, T, D_2, 0]$ where now T is *block* tridiagonal. No theoretical assurance on the condition of the subproblems is readily available, and in fact C corresponds to a two dimensional elliptic partial differential equation which can be potentially extremely ill conditioned. The presence of the two diagonal blocks D_i mitigates this somewhat, but bounds analogous to those in Chapter 4 are not available.

More ominously, the Cholesky factor R of the normal equations of C has upper bandwidth n, so storing all such factors R requires $\mathcal{O}(N^{4/3})$ memory, violating the second criteria. This can be circumvented by using an iterative solver for the subproblems that define the projectors, an approach examined for two dimensional problems in [9]. One fault with this is that the subsystems must be solved once or twice on every RP iteration so work is implicitly duplicated. A second fault is that the tolerance which the iterative subsystem solver uses is not easily set *a priori*. Numerical testing on two dimensional problems indicated that this tolerance must be an order of magnitude smaller than that required for the system $Ax = b$, and this tighter requirement must hold for every RP iteration. This implies an exponentially increasing amount of work as the tolerance for the system $Ax = b$ decreases. While some work has been done on the use of similar inner-outer iterative processes (see [25]), for RP methods accuracy in computing the projections is extremely important. For these reasons the partitioning with $m = 3$ is not used for three dimensional problems.

3.4 Generalizing the Row Partitioning Approach for Other Problems

The testing results of Chapter 6 show that in fact the approach taken here for partitioning A is extremely effective. However, it is of limited use if applicable only to seven point difference operator matrices for systems on the unit cube with Dirichlet boundary conditions. The reasoning for selecting the partitioning is based on a natural decoupling induced by the computational molecule used. This decoupling with no changes necessary still occurs if, e.g., the domain is irregular and/or a 27-point difference operator is applied. Extensions to Neumann or periodic boundary conditions are also possible, again by considering the decoupling available on the mesh. Although more complicated, such a decoupling occurs with finite element methods applied to partial differential equations since usually the elements are chosen to have common support with only a few other elements.

For the $m = 9$ partitioning, what determined the storage requirements of $3N$ for all of the Cholesky factors was that the length of the longest line in the computational star is 3 nodes. More generally, if the longest line in the computational star is l, the Cholesky factors require only lN extra storage.

Clearly then the scheme used here for row partitioning can be extended to other, less simple, partial differential equations.

3.5 Choice of Iteration Parameters

The iteration parameters used in the numerical testing of the product forms are $\omega_i = 1$ for all i, giving a symmetric Kaczmarz iteration matrix

$$Q_\pi = (I - P_1)(I - P_2)\cdots(I - P_m)\cdots(I - P_2)(I - P_1), \tag{3.14}$$

The grounds for this choice are the following:

1. At least $1/m$ of the eigenvalues of Q_π are zero; this is because $(I - P_1)x = 0$ for $x \in \text{range}(P_1)$. More generally, the number of eigenvalues equal to zero is at least the same as $\text{rank}(A_1)$.

2. For $m = 2$, $\omega = 1$ minimizes the spectral radius of Q_π when all the weights are equal. Although not true for $m > 2$, practical experience shows that $\rho(Q_\pi)$ is not sensitive to changes in ω. This matches classical results for SSOR iterations which are not as sensitive to ω as the corresponding SOR methods. The small improvement that does result from choosing a $\rho(Q_\pi)$-minimizing ω is more than offset by the introduction of $1/m$ nonzero eigenvalues that occurs when $\omega \neq 1$.

3. If initialized by $x_0 = 0$, at the end of each multiplication by Q_π, $A_1^T x_k = b_1$ holds in exact arithmetic. This can be seen from (3.14). The last operation that Q_π applies is $I - P_1$, which projects the iterate onto the set of solutions to the first block row of equations. This also continues to hold when acceleration techniques such as Chebyshev or CG are used, provided that CG acceleration uses the initialization $x_0 = b_\pi$ (see Section 5.3.) For more general applications this also allows putting equations deemed more important into the first block, e.g., equations defined by constraints on the solution.

4. Since one of the goals of this thesis is a 'black-box' algorithm without parameter estimation, such an *a priori* choice is required.

5. For the new Kaczmarz RP method in Section 2.2.3, $\omega = 1$ allows an explicit reduction in the problem size. This occurs because the system matrix $I - V_\pi$ can be written as $S_V^T S_V$, where

$$\begin{aligned} S_V &= A^T(D + \omega L)^{-T} R^T \\ &= [Q_1, (I - \omega P_1)Q_2, (I - \omega P_1)(I - \omega P_2)Q_3, \\ &\quad \ldots, (I - \omega P_1)\cdots(I - \omega P_{m-1})Q_m] \end{aligned} \tag{3.15}$$

where Q_i is any matrix with orthonormal columns forming a basis for range(A_i). Hence the block (1,1) entry of $S_V^T S_V$ is $Q_1^T Q_1 = I$, and the (1,i) block is

$$Q_1^T (I - \omega P_1) \left(\prod_{j=2}^{i-1} (I - \omega P_j) \right) Q_i \qquad (3.16)$$

Since

$$Q_1^T (I - \omega P_1) = Q_1^T - \omega Q_1^T Q_1 Q_1^T = (1 - \omega) I, \qquad (3.17)$$

this is zero iff $\omega = 1$. Because $S_V^T S_V$ is symmetric, the same holds for the first block column entries (i,1), $i > 1$. For the case $m = 3$, e.g., this implies $I - V_\pi$ has the form

$$\begin{bmatrix} I & 0 & 0 \\ 0 & {Q_2}^T (I - P_1) Q_2 & {Q_2}^T (I - P_1)(I - P_2) Q_3 \\ 0 & {Q_3}^T (I - P_2)(I - P_1) Q_2 & {Q_3}^T (I - P_2)(I - P_1)(I - P_2) Q_3 \end{bmatrix} \qquad (3.18)$$

where each block has dimension $N/3 \times N/3$. Clearly the first $N/3$ unknowns require no computation to find, and so the iteration only need be applied to a system 2/3 (more generally, $(m-1)/m$) the size of the original system. An important point is that this reduction has no detrimental effect on the spectrum of the iteration matrix, since V_π is similar to Q_π and so has the same eigenvalues.

Summarizing, even if $\omega_i = 1$ is not the 'optimal' choice for product forms in terms of minimizing the spectral radius, for computational reasons it is overwhelmingly the best choice.

For the Cimmino forms $\omega_i = 1/m$ is used, but now the choice is not so clear-cut. The second and fourth grounds for $\omega_i = 1$ in the Kaczmarz methods still apply, but an equally important reason for this choice is simply a lack of theory to guide another selection. Considering the case $m = N$ where each row of A has norm 1 is illustrative. Let $\Omega = diag(\omega_1, \omega_2, \ldots, \omega_N)$. The Cimmino system becomes

$$A^T \Omega A x = A^T \Omega b \qquad (3.19)$$

and selecting 'optimal' weights is equivalent to solving

$$\min_{\Omega} \ \| I - A^T \Omega A \| \qquad (3.20)$$

The difficulty of doing this exceeds that of solving $Ax = b$.

For both the Kaczmarz and Cimmino forms the most important reason for choosing the weights as above is an *a posteriori* one; the resulting algorithms work well and achieve the goals set out earlier.

Chapter 4

Error Analysis and Stability of RP Methods

For the partitioning schemes proposed in Chapter 3 computing the projections $P_i x$ requires computing projections onto separate subblocks of A_i^T. To simplify notation these subblocks are labeled C_j^T in this chapter, and S_i is the set of indices j corresponding to block row i. In the $m = 9$ row partitioning $j = 1, 2, \ldots, n^2$ and each C_j^T consists of the n rows of A corresponding to a single line in the computational mesh. Except for lines on the first and last planes or for the first and last lines of a plane, each C_j^T has the form given by (3.12).

Finding the projections as suggested in Chapter 3 involves first explicitly forming the normal equations $C_j^T C_j$ and finding their Cholesky factors R_j, then performing triangular solves on every iteration. These operations are potentially detrimental if the condition numbers of the rectangular matrices C_j^T are large.

The first section in this chapter analyses the conditioning of the submatrices for the $m = 9$ row partitioning. Theoretical bounds on $\kappa(C_j)$ are derived, and the derivation also provides practical numerical methods for checking the condition numbers. Applied to the test problems these results show that in fact each subproblem is extremely well-conditioned.

The second section summarizes well-known error analyses for the matrix operations required by RP methods. Together with the bounds from the first section these results prove the stability of RP methods and show that for the class of test problems examined here, the projectors P_i are numerically well-defined.

4.1 Subproblem Conditioning

The error analysis of row projection methods in the next section shows that the numerical properties of the algorithms rely critically on the condition numbers $\kappa(C_j)$ of the subproblems. For the seven point difference operator applied to linear elliptic partial differential equations and the choice $m = 9$ estimates of the largest and smallest singular values of C_j are derived, providing bounds on the condition number. These estimates yield *a priori* estimates based on the properties of the coefficient functions of the partial differential equation, and equally important provide efficient computational methods for detecting potential ill conditioning in the subproblems. Furthermore, numerical results are presented showing that for the test problems the bounds obtained are usually accurate and the subproblems remain extremely well conditioned even when the original matrix A is ill conditioned. As an example, for Poisson's equation in three dimensions the largest possible condition number that can be encountered among all of the subproblems is bounded by 4.75; in practice it has not exceeded 3.63. This is true for any mesh size even though the overall condition number of A in this case becomes arbitrarily large as N increases.

Some bounds on the subproblem condition numbers can be made immediately. Since C_j is a submatrix of A, the minimax characterization of singular values [26, Cor. 8.3-3] implies that the smallest singular values σ_{min} satisfy $\sigma_{min}(C_j) \geq \sigma_{min}(A)$ and that the largest singular values σ_{max} satisfy $\sigma_{max}(C_j) \leq \sigma_{max}(A)$, so that $\kappa(C_j) \leq \kappa(A)$. In fact the subproblems can be extremely well conditioned even when the matrix A is ill conditioned. This can be seen by considering the case when $m = N$, i.e., when each projector is defined by a single row of A. In this case $C_j^T C_j$ is a nonzero scalar and so $\kappa(C_j) = 1$. For the test problems examined here the subproblems are also perfectly conditioned when $m = 27$ and the row partitioning of Figure 3.2 is used. On the other hand when $m = 1$, $\kappa(C_j) = \kappa(A)$.

Although the results shown here are particular for the $m = 9$ partitioning applied to matrices generated by the seven point central difference operator applied to elliptic problems on the unit cube, for more general matrices A and partitioning schemes a theorem from C. Johnson [36] is presented that provides lower bounds on the smallest singular values of the submatrices A_i. Combined with well-known upper bounds on the largest singular values of the A_i's this provides estimates on the condition numbers of the subproblems for arbitrary matrices.

4.1.1 Types of Subproblems

A typical row in the test problem matrices has nonzero elements

$$\begin{aligned}
z_- &= hf/2 - c \\
y_- &= he/2 - b \\
x_- &= hd/2 - a \\
x_0 &= 2(a+b+c) - h^2 g \\
x_+ &= -(hd/2 + a) \\
y_+ &= -(he/2 + b) \\
z_+ &= -(hf/2 + c)
\end{aligned} \tag{4.1}$$

in column-increasing order, where all of $a - f$ are evaluated at (x_0, y_0, z_0) and $h = 1/(n+1)$ is the mesh size. There are nine different types of C_j, corresponding to which line-block row j refers. To see this, recall the nonzero structure of A given in Figure 3.1. Most of the submatrices C_j have the form shown in (3.12). The submatrix corresponding to the first n equations, however, has the form

$$C^T = [T, D_3, 0, D_4, 0] \tag{4.2}$$

Nevertheless, for examining the condition number of C_1 it suffices to consider a matrix of the form (3.12) with $D_1 = D_2 = 0$ or equivalently, $z_- = y_- = 0$. Similarly, the blocks C_j corresponding to the n^{th} line in the grid can be considered as having $D_3 = 0$ or equivalently $y_+ = 0$. Proceding in this way the subproblems fall into nine categories as listed in Table 4.1. Bounds on $\lambda_{min}(C_j^T C_j)$ and $\lambda_{max}(C_j^T C_j)$ are derived using Gerschgorin's Disk Theorem and the minimax characterization of singular values. The analysis is shown for subproblems of type 9 since all but approximately $4n$ of the n^2 subproblems lie in this category, and the other cases give similar results. First note that for a given C_j the functions $a - g$ become functions of x alone since C_j is comprised of the set of equations for a single line in a single plane of the grid. For a given row in C_j^T, let (x_0, y_0, z_0) be the corresponding grid point.

Subproblem Type	Characteristics	j
1	$z_- = y_- = 0$	1
2	$z_- = 0$	$2, 3, \ldots, n-1$
3	$z_- = y_+ = 0$	n
4	$y_- = 0$	$n+1, 2n+1, \ldots, (n-2)n+1$
5	$y_+ = 0$	$n, 2n, \ldots, (n-1)n$
6	$y_- = z_+ = 0$	$n^2 - n + 1$
7	$z_+ = 0$	$n^2 - n + 2, \ldots, n^2 - 1$
8	$z_+ = y_+ = 0$	n^2
9		All Else

Table 4.1: Categories of Subproblems

Also let

$$\begin{aligned} a &= a(x_0, y_0, z_0) \\ a_+ &= a(x_0 + h, y_0, z_0) \\ a_{++} &= a(x_0 + 2h, y_0, z_0) \end{aligned} \tag{4.3}$$

and define

$$\begin{aligned} a_M &= \max_x |a(x, y_0, z_0)| \\ a_m &= \min_x |a(x, y_0, z_0)| \end{aligned} \tag{4.4}$$

Let similar definitions hold for the functions $b - g$. Bounds on the singular values of C_j are derived using the basic relationships $\sigma^2_{min}(C_j) = \lambda_{min}(C_j^T C_j)$ and $\sigma^2_{max}(C_j) = \lambda_{max}(C_j^T C_j)$.

4.1.2 Gerschgorin-type Estimates

The matrix $C_j^T C_j$ is pentadiagonal and symmetric, so it suffices to examine for this given subproblem the components

$$c_k = \left[C_j^T C_j\right]_{i,i+k}, \quad k = -2, -1, 0, 1, 2. \tag{4.5}$$

For type 9 subproblems, forming the normal equations yields

$$c_0 = 6(a^2+b^2+c^2)+8(ab+ac+bc)+h^2[(d^2+e^2+f^2)/2-4g(a+b+c)]+\mathcal{O}(h^4) \tag{4.6}$$

$$
\begin{aligned}
|c_1| &= |[2(a+b+c)-h^2g](hd_+-2a_+)- \\
&\qquad [2(a_++b_++c_+)-h^2g_+](hd+2a)|/2 \\
&\le [2(a_M+b_M+c_M)+h^2g_M](hd_M+2a_M)
\end{aligned} \tag{4.7}
$$

$$
\begin{aligned}
|c_2| &= |(a+hd/2)(a_{++}-hd_{++}/2)| \\
&\le a_M^2+ha_Md_M+h^2d_M^2/4
\end{aligned} \tag{4.8}
$$

The same bounds hold for $|c_{-1}|$ and $|c_{-2}|$ as for $|c_1|$ and $|c_2|$, respectively. Hence Gerschgorin's Disk Theorem implies that the largest eigenvalue of $C_j^TC_j$ satisfies

$$
\begin{aligned}
\lambda_{max} &\le 16a_M^2+6(b_M^2+c_M^2)+16a_M(b_M+c_M)+8b_Mc_M \\
&\quad +h^2[d_M^2+(e_M^2+f_M^2)/2-4g_m(a_m+b_m+c_m)+4g_Ma_M] \\
&\quad +h[4d_M(2a_M+b_M+c_M)]+\mathcal{O}(h^3) \\
&=: L_{max}
\end{aligned} \tag{4.9}
$$

When $a=b=c$,

$$
\begin{aligned}
\lambda_{max} &\le 68a_M^2+16hd_Ma_M+ \\
&\quad +h^2[d_M^2+(e_M^2+f_M^2)/2+4(g_Ma_M-3g_ma_m)]+\mathcal{O}(h^3)
\end{aligned} \tag{4.10}
$$

Thus for Poisson's equation $\lambda_{max}(C_j^TC_j)\le 68$. Similarly, the smallest eigenvalue of $C_j^TC_j$ is bounded by

$$
\begin{aligned}
\lambda_{min} &\ge 6(a_m^2+b_m^2+c_m^2)+8(a_mb_m+a_mc_m+b_mc_m) \\
&\quad -2a_M(5a_M+4b_M+4c_M)-2hd_M(3a_M+2b_M+2c_M) \\
&\quad +h^2[(d_m^2-d_M^2+e_m^2+f_m^2)/2-4g_M(2a_M+b_M+c_M)] \\
&=: L_{min}
\end{aligned} \tag{4.11}
$$

When $a=b=c$,

$$
\begin{aligned}
\lambda_{min} &\ge 42a_m^2-26a_M^2-14hd_Ma_M \\
&\quad +h^2[(d_m^2-d_M^2+e_m^2+f_m^2)/2-16g_Ma_M]
\end{aligned} \tag{4.12}
$$

For Poisson's equation this implies that $\lambda_{min}\ge 16$ and hence

$$\kappa(C_j^TC_j)\le 68/16=4.25.$$

Putting together the results above gives

Theorem 4.1 *For subproblems from category 9 as defined in Table 4.1, the condition number $\kappa(C_j^T C_j)$ is bounded by L_{max}/L_{min}, where L_{max} is defined in (4.9) and L_{min} is defined in (4.11).*

The arguments for the other categories of subproblems are similar to the one given above. In particular it is easy to show that for subproblem C_1, the second term in (4.9) is replaced by $5(b_M^2 + c_M^2)$ and the first term in (4.11) is replaced by $6a_m^2 + 5(b_m^2 + c_m^2)$. For Poisson's equation, this gives the bound

$$\kappa(C_1^T C_1) \leq 66/14 \leq 4.75.$$

4.1.3 Minimax Estimates

For some problems the Gerschgorin Disk Theorem provides an estimate $\lambda_{min}(C_j^T C_j) \leq 0$. In these cases a better bound on this smallest eigenvalue comes from an application of the minimax characterization of singular values:

Theorem 4.2 *Suppose that $C_j^T = [D_1 \ 0 \ D_2 \ T \ D_3 \ 0 \ D_4]$ where each $D_i \in \Re^n$ is diagonal, and let $\delta_i = \min_k |D_i|_{kk}$. Then*

$$\sigma_{min}^2(C_j) = \lambda_{min}(C_j^T C_j) \geq \delta_1^2 + \delta_2^2 + \delta_3^2 + \delta_4^2.$$

Proof: Let $z \in \mathcal{R}^n$, $\| z \| = 1$. Then

$$\begin{array}{rcl} \| C_j z \|^2 & = & \| D_1 z \|^2 + \| D_2 z \|^2 + \| Tz \|^2 + \| D_3 z \|^2 + \| D_4 z \|^2 \\ & \geq & \| D_1 z \|^2 + \| D_2 z \|^2 + \| D_3 z \|^2 + \| D_4 z \|^2 \\ & \geq & \delta_1^2 + \delta_2^2 + \delta_3^2 + \delta_4^2 \end{array} \tag{4.13}$$

Minimizing $\| C_j z \|$ over all such z gives the stated result. $\square$

For subproblems not in category 9 one or two of the δ_i's are missing, e.g., for subproblem C_1 the bound becomes

$$\sigma_{min}^2(C_1) \geq \delta_3^2 + \delta_4^2.$$

Theorem 4.2 gives bounds for any matrix with the same structure as that of the seven point difference operator. The result can be sharpened for uniformly elliptic partial differential equations. Recall that the definition of uniform ellipticity implies that there is some $\delta \geq 0$ such that $a, b, c \geq \delta > 0$ on the domain.

Corollary 4.1 *Suppose that $b(x, y, z) > \delta$ and $c(x, y, z) > \delta$, where $\delta > 0$. Then for subproblems in category 9, $\sigma_{min}(C_j) \geq 2\delta$*

Proof: The diagonal elements of the D_i's are given by

$$\begin{aligned} [D_1]_{jj} &= hf/2 - c \\ [D_2]_{jj} &= he/2 - b \\ [D_3]_{jj} &= -(he/2 + b) \\ [D_4]_{jj} &= -(hf/2 + c) \end{aligned}$$

where b, c, e, f are all evaluated at the point (jh, y_0, z_0). From the first inequality in the proof of Theorem 4.2,

$$\begin{aligned} \sigma^2_{min}(C_j) &\geq min\left[(hf - 2c)^2 + (hf + 2c)^2 + (he + 2b)^2 + (he - 2b)^2\right]/4 \\ &= min\left[2(b^2 + c^2) + h^2(e^2 + f^2)/2\right] \\ &\geq min\ 2(b^2 + c^2) \\ &\geq 4\delta^2 \end{aligned}$$

where the minimization is over $x \in (0, 1)$. □

As before, it is trivial to extend the result to the other categories of subproblems, and in fact for *all* subproblems $\sigma_{min}(C_j) \geq \sqrt{2}\delta$.

4.1.4 Discussion and Results for Test Problems

No assumption is made above about continuity of the coefficient functions of the elliptic partial differential equations. This is important for problems that result from linearization of some nonlinear problems that can develop shocks in the solution, e.g., Burger's equation.

Easier bounds result if $\mathcal{O}(h^2)$ terms are dropped, but the usual hypothesis that h is 'sufficiently small' has been avoided because in practice the $\mathcal{O}(h^2)$ terms are significant. As an example, for test Problem 8, $h^2 d_M^2 \approx \mathcal{O}(10^6)$

when $h = 1/48$. In order to reduce $h^2 d_M^2$ to $\mathcal{O}(10^{-2})$ by refining the mesh it is necessary to create a linear system of order $N \approx 10^{18}$, which exceeds the memory capacity of currently available computers.

The bounds also help reveal how to arrange the numbering of the nodes to achieve well conditioned subproblems. Consider the problem

$$\triangle u + \beta u_w = 0 \ , \ \beta \gg 1 \tag{4.14}$$

for $w = x, y,$ or z. Since the $\mathcal{O}(h)$ terms of L_{min} and L_{max} contain d_M but not e_M or f_M, the subproblems created when $w = x$ are expected to be more ill conditioned than those created when $w = y$ or z. Comparing the condition numbers for test problems 1 and 2 in Table 4.2 shows that in fact this occurs.

Bounds provided by Gerschgorin's Theorem are easily checked numerically as the problem is set up and this can be done in parallel across all of the subproblems at small cost. This is computationally more valuable than the *a priori* estimates and may provide better bounds since the actual component values are not replaced by the values a_M, a_m, etc. Table 4.2 gives three estimates for the maximal condition numbers encountered among the subproblems of the test problems. The first is based on the theoretical bounds on the largest and smallest singular values of the subproblems, as defined in (4.9), (4.11), and Corollary 4.1. This estimate is for submatrices in category 9 only. The second estimate is that obtained by applying Gerschgorin estimates numerically as the problem is generated, with the estimate for $\sigma_{min}(C)$ replaced by that of Corollary 4.1 whenever it is larger. The third estimate is computed using the Linpack routine DPOCO [15].

For Problems 2,3, and 5 the theoretical bound is actually smaller than the computed condition number. This is because the Gerschgorin numerical estimates and the Linpack condition numbers given in the table are the maximal ones over all subproblem types, while the theoretical bounds shown are for subproblems of type 9. Whenever the maximal condition number shown is larger than the estimate it does occur in a subproblem not in category 9. In any case the *a priori* bounds are within an order of magnitude of the true values, while the numerical bounds are usually extremely accurate.

In the event that a few of the subproblems have extremely large condition numbers, the QR decomposition can be used to generate the corresponding Cholesky factor. This approach has greater numerical stability, but at the

Problem	Theoretical for Category 9	Numerical for All Categories	Actual for All Categories
1	20.5	7.94	4.93
2	1.12	1.24	1.21
3	1.75	1.92	1.87
4	32.7	5.61	5.14
5	25.6	74.9	61.2
6	2088	1608	518
7	32.8	12.9	8.56
8	1480	1273	803
9	25.2	2.19	2.13

Table 4.2: Estimated $\max_j \kappa(C_j)$ for $n = 48$

cost of more flops. In particular, QR decomposition using Given's rotations requires $28n^2$ flops for each C_j, while forming the normal equations and using Cholesky factorization requires only $12n^2$ flops. However, this is a one time extra cost occuring in the preprocessing stage of the algorithm, and constitutes an acceptable price for stability. At no time in the testing has this been necessary, and none of the subproblems encountered has had a condition number greater than $\mathcal{O}(10^3)$.

4.1.5 Estimates for General Problems

As discussed earlier, depending on the partitioning used the condition number of the subproblems can in general range from 1 up to the condition number of A. Bounds on the smallest singular values of rectangular matrices allow choosing partitions of more general matrices that also are provably well conditioned without choosing $m = N$. Three such bounds are presented in [36], the best of which is

Theorem 4.3 *Let $C = (c_{ij})$ be an $s \times n$ matrix with $s \leq n$. Then*

$$\sigma_{min}(C) \geq \min_{1 \leq i \leq s} \left[|c_{ii}| - \frac{1}{2} \left(\sum_{j \neq i}^{s} |c_{ij}| + \sum_{j \neq i}^{s} |c_{ji}| \right) \right] \tag{4.15}$$

Since the singular values of a matrix are unchanged when it is postmultiplied by a permutation matrix, this theorem actually provides several bounds for a given matrix, corresponding to different permutations of the columns of C. These can be combined with an upper bound on the singular values to estimate the condition number of C. The simplest such bound is given by the Frobenius norm of C: $\sigma_{max}(C) = \| C \|_2 \leq \| C \|_F$

As might be expected, such estimates are usually not as sharp as those which rely on special structure of the matrix C. For example, for Poisson's equation and the partitioning with $m = 9$ the estimate is $\kappa(C) \leq 2.5\sqrt{n}$, which should be compared to the previously obtained estimate 4.75. Nevertheless these results are important since they can be applied to arbitrary matrices.

4.2 Error Analysis

This section presents an error analysis for the row projection methods. The analysis shows that if the smallest singular values of the submatrices C_j are bounded away from 0, the projector P_i is well-behaved subject to perturbations in the submatrices C_j. A rounding error analysis concentrating on those computations which are most suspect — the formation of the normal equations of each C_j^T and the subsequent Cholesky factorization — shows that for seven point difference matrices this particular approach yields small errors in the computation of the projectors.

This section does not present an error analysis of the acceleration methods used since such analysis has been presented for the conjugate gradient method in [8] and for Chebyshev acceleration in [25]. Block Stiefel acceleration is discussed in detail in the next chapter.

4.2.1 Perturbational Analysis

Let $A^T = [A_1, A_2, \ldots, A_m]$, and set $P_i =$ orthogonal projector onto $range(A_i)$. The effects of perturbations of A on P_i were analyzed thoroughly by G. Stewart in [56], and those results are simply summarized here and applied to the row projection methods. Let

$$B_i = A_i + E_i$$

and let $\tilde{P}_i$ = the orthogonal projector onto $range(B_i)$. Stewart's results divide into the two cases $rank(B_i) = rank(A_i)$ and $rank(B_i) \neq rank(A_i)$. The latter is not as tractable as the former; for example, it can be shown that when $rank(B_i) \neq rank(A_i)$,

$$\| B_i^{\dagger} \| \geq 1/ \| E_i \| \tag{4.16}$$

and so small perturbations in A_i *must* lead to large perturbations in the computed pseudo-inverse.

However, Corollary 4.1 shows that for elliptic problems with second derivative terms

$$a(x,y,z)u_{xx} + b(x,y,z)u_{yy} + c(x,y,z)u_{zz}$$

the smallest singular value of C_j is bounded below by

$$\sigma_{min}(C_j) \geq \sqrt{2}\,\delta,$$

where $b(x,y,z)$, $c(x,y,z) \geq \delta$. Since $\sigma_{min}(A_i)$ is the distance in the Frobenius norm between A_i and the nearest lower rank matrix, this implies that a pertubation E_i must be larger than $\sqrt{2}\,\delta$ to change the rank of A_i. For this reason only the case $rank(B_i) = rank(A_i)$ need be considered here.

In this case Stewart has shown that

$$\| A_i^{\dagger} - B_i^{\dagger} \| \leq 3 \| E_i \| \cdot \| B_i^{\dagger} \| \cdot \| A_i^{\dagger} \| \tag{4.17}$$

and

$$\| \tilde{P}_i - P_i \| = \| \tilde{P}_i(I - P_i) \| \leq \| E_i \| \min \left\{ \| B_i^{\dagger} \|, \| A_i^{\dagger} \| \right\} \tag{4.18}$$

As a historical note the first explicit statement and proof of the CS decomposition occurs in Stewart's 1977 paper, and was used to establish the last equality.

The inequality (4.18) shows that the projectors are locally continuous functions of the submatrices A_i in the full rank case. The triangle inequality immediately gives

$$\| (\tilde{P}_1 + \tilde{P}_2 + \cdots + \tilde{P}_m) - (P_1 + P_2 + \cdots + P_m) \| \leq \sum_{i=1}^{m} \| E_i \| \cdot \| A_i^{\dagger} \|$$

This implies that the Cimmino iteration matrix is numerically well behaved subject to perturbations of A_i that do not change the rank. For the Kaczmarz iteration the bound is the same:

Fact 4.1 *If P_i = projector onto range(A_i), $\tilde{P}_i$ = projector onto range$(A_i + E_i)$, and rank(A_i) = rank$(A_i + E_i)$ for $i = 1, 2, \ldots, m$ then*

$$\| \tilde{P}_1 \tilde{P}_2 \cdots \tilde{P}_m - P_1 P_2 \cdots P_m \| \le \sum_{i=1}^{m} \| E_i \| \cdot \| A_i^\dagger \|$$

Proof: : For $m = 1$ the result follows from (4.18). Suppose that the bound holds for $m - 1$. Then

$$\begin{aligned} & \| \tilde{P}_1 \tilde{P}_2 \cdots \tilde{P}_m - P_1 P_2 \cdots P_m \| \\ = \; & \| \tilde{P}_1 \tilde{P}_2 \cdots \tilde{P}_m - P_1 P_2 \cdots P_{m-1} \tilde{P}_m + P_1 P_2 \cdots P_{m-1} \tilde{P}_m - P_1 P_2 \cdots P_m \| \\ \le \; & \| \tilde{P}_1 \tilde{P}_2 \cdots \tilde{P}_{m-1} - P_1 P_2 \cdots P_{m-1} \| \| \tilde{P}_m \| + \| P_1 P_2 \cdots P_{m-1} \| \| \tilde{P}_m - P_m \| \\ \le \; & \textstyle\sum_{i=1}^{m-1} \| E_i \| \cdot \| A_i^\dagger \| + \| E_m \| \cdot \| A_m^\dagger \| \end{aligned}$$

because of the induction hypothesis, $\| \tilde{P}_m \| = 1$, and $\| P_1 P_2 \cdots P_{m-1} \| \le 1$. The fact now follows by induction. □

If the errors incurred in vector subtraction are neglected, the above bound certainly holds when P_i is replaced by $I - P_i$.

In summary, both the Cimmino and Kaczmarz iteration matrices are numerically well behaved under perturbations in A provided that the submatrices A_i have singular values bounded away from 0.

4.2.2 Rounding Error Analysis

The perturbational analysis does not provide information about the numerical properties of the row projection methods in relation to the particular numerical methods used to compute the projections. For this the errors incurred on the separate steps of forming the normal equations and computing the projections using their Cholesky decompositions must be examined. The rounding error analyses of [26] and [63] are used, and rounding errors incurred in vector subtraction and matrix-vector multiplications are omitted since the resulting relative errors are minor in comparison.

Let t be the machine precision, that is, let t be the number of bits in the mantissa. Set $\mu = 1.06 \cdot 2^{1-t}$; the number μ is a modification of the machine's unit roundoff error that allows simplified error bounds whenever the vector lengths n satisfy

$$n < 0.1 \cdot 2^{t-1}$$

This bound on n is assumed to hold throughout this chapter.

Formation of Normal Equations

When the normal equations $C_j^T C_j$ are formed, each element is found as an inner product of length less than or equal to the maximum number of nonzero elements in any row of A. Using the bounds on inner product computations in [55], the computed inner product $fl(x^T y)$ satisfies

$$fl(x^T y) = (1+\alpha)x^T y, \quad |\alpha| < \tilde{n}\mu$$

where $\tilde{n}$ is the length of the inner product. It follows that $C_j^T C_j + E$ is computed, where for the particular class of test problems examined here,

$$|E_{ii}| \leq 7\mu \parallel c_i \parallel^2 \leq 7\mu \parallel C_j \parallel^2$$

with c_i the i$^{\text{th}}$ column of C_j. In a similar fashion the off-diagonal elements of E satisfy

$$\begin{aligned} |E_{i,i+1}| &\leq 2\mu \parallel C_j \parallel^2 \\ |E_{i,i+2}| &\leq \mu \parallel C_j \parallel^2 \end{aligned}$$

Hence

$$\parallel E \parallel_2^2 \leq n \parallel E \parallel_F^2 \leq n \sum_{i=1}^{m} 54\mu^2 \parallel C_j \parallel^4$$

and so

$$\parallel E \parallel_2 \leq 8n\mu \parallel C_j \parallel^2 \tag{4.19}$$

Note that the factor $\parallel C_j \parallel^2$ results from replacing the norm of a row or part of a row with that of the overall matrix and is usually a generous overestimate.

Computing an Individual Projection

The computation of a projection relies on finding the Cholesky decomposition $C_j^T C_j = R^T R$ and then solving two triangular systems given by R and R^T. In [63] a complete analysis of the Cholesky decomposition is given. The results needed are summarized here:

Theorem 4.4 *Suppose that*

$$\kappa(C_j^T C_j) \leq n^{3/2}/(10\mu)$$

Then the computed Cholesky factor R of $C_j^T C_j$ satisfies

$$R^T R + E = C_j^T C_j$$

where

$$\| E \|_2 \leq \mu\gamma_n \ \| C_j^T C_j \|$$

and

$$\gamma_n = \left[1 + (\sqrt{n} - 2.2)\mu\right]^n \times \left[2(n+1)^{3/2}/3 + 1.1n\right]$$

Furthermore, if $10 < n < 0.1\mu^{-1}$*, then*

$$\| E \|_2 \leq 2n^{3/2}\mu \| C_j^T C_j \| \tag{4.20}$$

Proof: : See [63]. □

Note that it is n and not $N = n^3$ that appears in the above bounds. Since each projector P_i is defined in terms of the block rows A_i^T and each A_i is the direct sum of C_j's for $j \in S_i$, the bound for a projector follows by taking the maximum over all such C_j's:

Corollary 4.2 *Suppose that* $10 < n < 0.1\mu^{-1}$ *and*

$$\max_{j \in S_i} \kappa(C_j^T C_j) \leq n^{3/2}/(10\mu)$$

If the rounding errors from vector subtraction and matrix-vector multiplication are neglected, the computed version $\tilde{y}_i$ *of* $y_i = (I - P_i)x$ *is given by* $\tilde{y}_i = (I - P_i + E_i)x$*, where*

$$\| E_i \|_2 \leq 2n^{3/2}\mu \max_{j \in S_i} \| C_j^T C_j \|$$

Product of Projectors

First consider two projectors with errors introduced:

$$(I - P_1 + E_1)(I - P_2 + E_2) = (I - P_1)(I - P_2) + (I - P_1)E_2 + E_1(I - P_2) + E_1E_2$$

Since the norm of the last term above is $\mathcal{O}(\mu^2)$,

$$\begin{aligned}
&\| (I - P_1 + E_1)(I - P_2 + E_2) - (I - P_1)(I - P_2) \| \\
&\quad \leq \ \| (I - P_1)E_2 + E_1(I - P_2) \| + \mathcal{O}(\mu^2) \\
&\quad \leq \ \| E_1 \| + \| E_2 \| + \mathcal{O}(\mu^2) \\
&\quad \leq \ 2 \cdot 2n^{3/2}\mu \max_j \| C_j^T C_j \| + \mathcal{O}(\mu^2)
\end{aligned}$$

with the factor $2n^{3/2}$ arising from (4.20). By repeatedly omitting $\mathcal{O}(\mu^2)$ terms this argument can be extended to show:

Fact 4.2 *Suppose that* $10 < n < 0.1\mu^{-1}$. *If* $\tilde{y}$ *is the computed version of* $y = (I - P_m)(I - P_{m-1}) \cdots (I - P_1)x$, *then* $\tilde{y} = y + e$, *where*

$$\| e \| \leq 2mn^{3/2}\mu \max_j \| C_j^T C_j \| + \mathcal{O}(\mu^2).$$

As an example, for $n = 63$ and a machine with $t = 48$, the symmetric form of Kaczmarz's iteration with $m = 9$ computes the product of projectors for Poisson's equation with an error bounded by 10^{-11}.

4.3 Summary

The bounds show that the numerical properties of row projection methods rely primarily on the condition numbers of the underlying subproblems. The section on subproblem conditioning together with this section show that RP methods based on partitioning seven point difference matrices into 9 blocks and using the normal equations of the subproblems are stable methods applied to numerically well defined matrices.

All of the results in Section 4.2.2 are applicable to general matrices A with full rank submatrices A_i^T, with the exception of (4.19) which must have the factor 8 replaced by N, the order of A.

The bounds derived in Section 4.2.2 all depend on the subproblem size n and not on the overall problem size $N = n^3$.

The factor m in the bound of Fact 4.2 indicates that it may be best to avoid partitioning the matrix A into too many block rows. The numerical results of Chapter 6 also provide grounds for limiting m.

Chapter 5

Acceleration Methods

As the contrived example in Section 2.4 illustrates, the RP system matrices can have eigenvalues arbitrarily close to zero. This is also true for practical applications, as the spectra of the RP system matrices in the appendices show. For this reason acceleration of the RP methods is essential. Because the three dimensional problems of this thesis yield nonsingular matrices A, this is assumed to hold throughout this chapter, although generally the acceleration techniques can be applied provided that the system is consistent and $x_0 \in \mathrm{range}(A^T)$.

The RP methods and weight parameters have been chosen so that the spectra of the system matrices must lie in $(0, 1]$, allowing powerful acceleration techniques to be applied. Three such techniques are examined: Chebyshev, block Stiefel, and CG. In particular a thorough analysis of block Stiefel acceleration is provided, and an iterative version proposed.

For convenience the RP systems to be solved are listed here. Each is of the form

$$(I - Q)x = \tilde{b} \tag{5.1}$$

The *Kaczmarz* system has

$$Q = Q_\pi = (I - P_1)(I - P_2)\cdots(I - P_m)\cdots(I - P_2)(I - P_1). \tag{5.2}$$

The *V-form* system has $Q = V_\pi = I - S_V^T S_V$, with

$$\begin{aligned} S_V \;=\; & [Q_1, (I - P_1)Q_2, (I - P_1)(I - P_2)Q_3, \\ & \ldots, (I - P_1)\cdots(I - P_{m-1})Q_m] \end{aligned} \tag{5.3}$$

where Q_i has orthonormal columns forming a basis of A_i. Although the V-form system actually solves directly for an auxillary vector z and then finds x by means of (2.33), the distinction is important only in Section 5.3. Finally the *Cimmino* system has

$$Q = Q_\sigma = 2/m \sum_{i=1}^{m} P_i \tag{5.4}$$

When referred to as 'the' system $(I - Q)x = \tilde{b}$, all three RP methods are understood. Finally, the *pseudo-residual* is $\tilde{b} - (I - Q)x$, in contrast to the *true residual* $b - Ax$.

5.1 Chebyshev Acceleration

The properties and techniques for Chebyshev acceleration have been thoroughly examined [31, 43, 44] and are summarized here for RP methods primarily to establish notation and concepts used in Section 5.2. The largest eigenvalue of $I-Q$ is assumed to be 1, so that only an estimate ν of the smallest eigenvalue need be made. Let $\nu \in (0, 1)$, $x_0 \in \Re^N$, and a positive integer K be given, and set $\delta = (1 + \nu)/2$, $\gamma = (1 - \nu)/2$. The *Chebyshev(x_0, ν, K) process* is

---

Initialize:

$$\begin{aligned} r_0 &= \tilde{b} - (I - Q)x_0 \\ d_0 &= r_0/\delta \\ x_1 &= x_0 + d_0 \\ \alpha_1 &= 2\delta/\left(2\delta^2 - \gamma^2\right) \\ \beta_1 &= \delta\alpha_1 - 1 \end{aligned} \tag{5.5}$$

Repeat for $k = 1, 2, \ldots, K$:

$$\begin{aligned} r_k &= \tilde{b} - (I - Q)x_k \\ d_k &= \alpha_k r_k + \beta_k d_{k-1} \\ x_k &= x_{k-1} + d_k \\ \alpha_k &= \left[\delta - \gamma^2\alpha/4\right]^{-1} \\ \beta_k &= \delta\alpha_k - 1 \end{aligned} \tag{5.6}$$

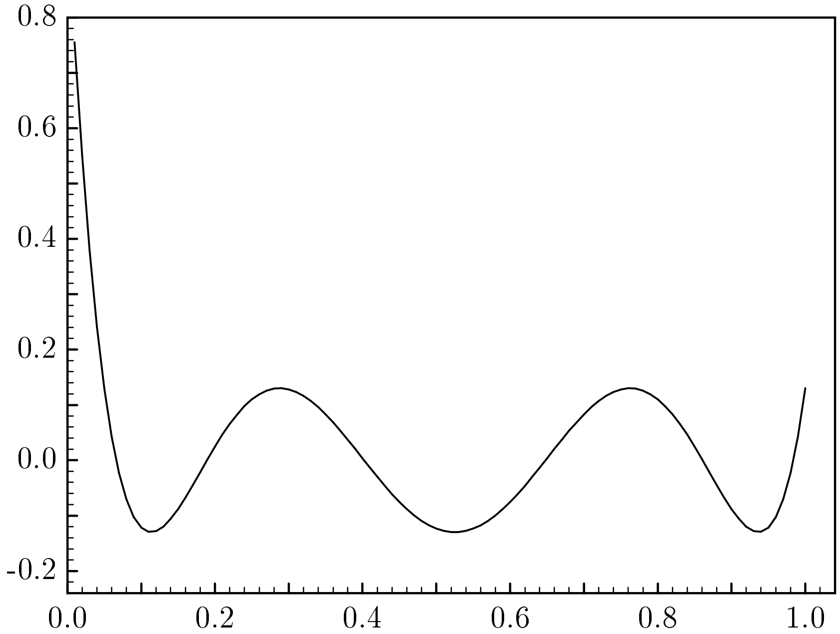


Figure 5.1: Chebyshev Polynomial

The above algorithm generates residuals r_k that satisfy

$$r_k = p_k(I - Q)r_0 \tag{5.7}$$

where

$$p_k(\lambda) = T_k\left(\frac{\delta - \lambda}{\gamma}\right) / T_k\left(\frac{\delta}{\gamma}\right) \tag{5.8}$$

and T_k is the k^{th} Chebyshev polynomial of the first kind on $[-1, 1]$. The shifted and normalized Chebyshev polynomial p_k is thus a residual polynomial as defined in Chapter 1, i.e., $p_k(0) = 1$. Figure 5.1 shows p_6 for $\nu = 0.05$. Let the eigenvalues of $I - Q$ be

$$0 < \lambda_1 \leq \lambda_2 \leq \cdots \lambda_N \leq 1 \tag{5.9}$$

with the corresponding orthonormal set of eigenvectors $v_1, v_2, \ldots, v_N$. Any vector w can be decomposed uniquely as

$$r = \sum_{i=1}^{N} \zeta_i v_i \tag{5.10}$$

The i^{th} *eigencomponent* of r is defined to be ζ_i. When $\nu \leq \lambda_1$ in the Chebyshev(x_0, ν, K) process, the pseudo-residuals r_k satisfy

$$\| r_k \| \approx \left[\frac{1-\nu}{(1+\sqrt{\nu})^2} \right]^k \| r_0 \| \tag{5.11}$$

and so the rate of convergence is best for $\nu = \lambda_1$. This provides also a formula to estimate the number K of iterations necessary to reduce the pseudo-residual by a factor ϵ for a given ν, namely by choosing K large enough so that the factor in (5.11) is less than ϵ.

Using Chebyshev acceleration for RP methods thus requires estimating one parameter, $\nu = \lambda_1$. Unfortunately, for the test problems λ_1 can be extremely small, e.g., $\lambda \approx 10^{-10}$. In this case even if ν is chosen to be equal to λ_1, reducing the residual by a factor of 10^{-6} calls for almost 6.9×10^5 iterations. Even though this is better than the 1.4×10^{11} iterations needed if acceleration is not applied, it is still unacceptably large for an iterative solver.

5.2 Block Stiefel Acceleration

The difficulty with Chebyshev acceleration is that if even one eigenvalue of $I - Q$ is small, convergence is slow regardless of the remaining spectral distribution. In 1985 Y. Saad, A. Sameh, and P. Saylor [49] proposed a scheme called *block Stiefel* (BST) acceleration to elude this obstacle.

Suppose that in Chebyshev(x_0, ν, K) acceleration ν is chosen larger than λ_1. Eigencomponents of r_k corresponding to eigenvalues in $[\nu, 1]$ are reduced even faster than if $\nu = \lambda_1$ were chosen, while eigencomponents of r_k corresponding to eigenvalues in $(0, \nu)$ are reduced at a much slower rate. Figure 5.2 illustrates this for p_6 when $\nu = 0.1$, and this graph should be compared to Figure 5.1.

If K were large enough so that r_K is effectively a multiple of the first eigenvector v_1, then a projection step (not to be confused with the projectors that define Q) can eliminate that eigencomponent from r_K, giving the solution. In general if the last s residuals are stored as $R = [r_{K-s+1}, \ldots, r_K]$ and span$(R) =$ span$[v_1, v_2, \ldots, v_s]$, then the projection step

$$x \leftarrow x + R(R^T(I-Q)R)^{-1}R^T r_k \tag{5.12}$$

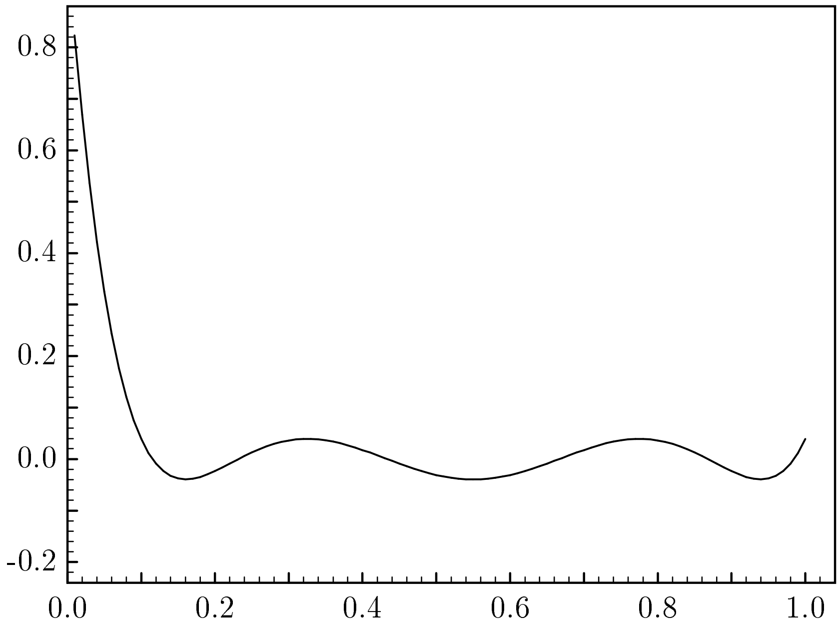


Figure 5.2: BST Polynomial

eliminates the corresponding s eigencomponents. This allows the Chebyshev acceleration to proceed as if the first s eigenvalues were absent from the problem. Because of storage limitations, possibly the number of eigenvalues less than ν is larger than s. In that case after the projection step $r_k \neq 0$ and further iterations may be required. This suggests an iterated BST method:

Iterated BST Acceleration

Initialize:

$$\begin{aligned} \nu_0 &= \text{given} \\ {x_0}^{(0)} &= \text{given} \\ K_0 &= \text{given} \\ j &= 0 \end{aligned} \tag{5.13}$$

Repeat:

Perform Chebyshev$({x_j}^{(0)}, \nu, K)$

Update $x_{j+1}{}^{(0)} = x_j{}^{(K)} + R_j y$, where y is chosen to minimize

$$\| (I - Q)(x_j{}^{(K)} + R_j y) - b \|_{(I-Q)} \tag{5.14}$$

and $R_j = \left[r_j{}^{(K)}, r_j{}^{(K-1)}, \ldots, r_j{}^{(K-s+1)} \right]$

Update ν and K

j = j + 1

As with Chebyshev acceleration, only one parameter, ν, need be estimated provided that (5.11) is used to update K. The next section derives a scheme for updating ν in BST acceleration.

5.2.1 Estimating ν

Suppose that no information is available initially about the spectrum of $I-Q$ except that it lies in $(0, 1]$. In this case the initial ν can be chosen somewhat arbitrarily from $(0, 1)$, for example, $\nu = 0.2$. The value of K is then selected to reduce the residual eigencomponents corresponding to large eigenvalues to the desired tolerance. To motivate the selection of the next value of ν, the following approximation is used when $| \xi | > 1$ and k large:

$$T_k(\xi) = \cosh(k \cosh^{-1}(\xi)) \approx \left(\xi + \sqrt{\xi^2 - 1} \right)^k / 2. \tag{5.15}$$

Also, define

$$\theta := \delta / \gamma = (1 + \nu)/(1 - \nu) > 1. \tag{5.16}$$

Once the first Chebyshev(x_0, ν, K) process has been executed, a good estimate of the smallest eigenvalue λ_1 is available in the following way. Suppose that the eigencomponents of r_k corresponding to eigenvalues in $(\nu, 1)$ are negligible. If there were only one eigenvalue $\lambda < \nu$, then

$$\frac{\| r_k \|}{\| r_{k-1} \|} \approx \frac{p_k(\lambda)}{p_{k-1}(\lambda)} \tag{5.17}$$

Since $\lambda \in (0, \nu)$, from the definition of Chebyshev polynomials

$$\frac{\| r_k \|}{\| r_{k-1} \|} \approx \frac{T_{k-1}(\theta)}{T_k(\theta)} \frac{T_k(\left(\frac{\delta-\lambda}{\gamma}\right))}{T_{k-1}(\left(\frac{\delta-\lambda}{\gamma}\right))}$$

$$\approx \frac{\cosh((k-1)\cosh^{-1}\theta)}{\cosh(k\cosh^{-1}\theta)} \frac{\cosh(k\cosh^{-1}\left(\frac{\delta-\lambda}{\gamma}\right))}{\cosh((k-1)\cosh^{-1}\left(\frac{\delta-\lambda}{\gamma}\right))}$$

Using the approximation $\cosh(\xi) = \left(e^{\xi} + e^{-\xi}\right)/2 \approx e^{\xi}/2$ and solving for λ gives

$$\lambda \approx \delta - \gamma \cosh\left[\cosh^{-1}\theta + \ln \| r_k \| - \ln \| r_{k-1} \|\right] \tag{5.18}$$

The approximation for $\cosh(\xi)$ is not useful in (5.18) because the expression in brackets may be close to 1. However, this can be approximated as

$$\lambda \approx \delta - \frac{\gamma}{2}\left[\frac{\| r_k \| \left(\theta + \sqrt{\theta^2 - 1}\right)}{\| r_{k-1} \|} + \frac{\| r_{k-1} \|}{\| r_k \| \left(\theta + \sqrt{\theta^2 - 1}\right)}\right] \tag{5.19}$$

The estimate above has proven computationally to give a good approximation to the smallest eigenvalue of $I - Q$. This may not provide a good value for the updated ν, for three reasons:

- Once the estimate of λ_1 is available, that eigencomponent may have been eliminated in the projection step. Hence the information may come too late to be of direct use.

- If the smallest eigenvalue is very small and is used as the next estimate of ν, then the next Chebyshev step may have a very large K assigned to it. This can be guarded against by setting $K \leq K_{\max}$.

- The desired goal is to have s eigenvalues less than ν. Knowing the smallest eigenvalue does not indicate how the next $s - 1$ eigenvalues are distributed.

However, note that in practice there may be more than one eigenvalue less than ν. In effect the λ in (5.17) will be a weighted average of the eigenvalues less than ν, so the estimates above can be expected to be slightly larger than the true λ_1, making (5.19) a good estimate for ν if not for λ_1.

5.2.2 The BST Projection Step

The feature that distinguishes BST from a restarted Chebyshev acceleration is the projection step. This section analyses the projection step in more detail, revealing a potential weakness of BST acceleration. Since a single projection step is considered, for convenience iteration indices are omitted and overbars indicate updated quantities, so that the projection step becomes

$$\begin{aligned} \bar{x} &= x + R[R(I-Q)R^T]^{-1}R^T r \\ R = R_j &= \left[{r_j}^{(K)}, {r_j}^{(K-1)}, \ldots, {r_j}^{(K-s+1)}\right] \end{aligned} \tag{5.20}$$

This implicitly assumes that R is full rank. The updated residual satisfies

$$\bar{r} = \left(I - (I-Q)R[(R^T(I-Q)R)]^{-1}R^T\right) r \tag{5.21}$$

or

$$\bar{r} = (I-Q)^{1/2}\tilde{Q}(I-Q)^{-1/2} r$$

with

$$\tilde{Q} = I - (I-Q)^{1/2} R \left[R^T(I-Q)R\right]^{-1} R^T (I-Q)^{1/2} \tag{5.22}$$

Since $\tilde{Q}$ is an orthogonal projector, it immediately follows that $\| \bar{r} \| \leq \| r \|$ and so the overall BST scheme is globally convergent, no matter how the parameter $\nu \in (0,1)$ is chosen.

Now let $V = [v_1, v_2, \ldots, v_N]$ and $D = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$. The eigendecomposition $r = \sum_{i=1}^N \zeta_i v_i$ can then be written as $r = Vz$, $z = (\zeta_1, \zeta_2, \ldots, \zeta_N)^T$. If $r_0 = Vz_0$ is the residual at the start of the Chebyshev(x_0, ν, K) process , then the k^{th} residual satisfies

$$r_k = p_k(I-Q)r_0 = Vp_k(D)z_0 \tag{5.23}$$

Define $Z = [p_{k-s+1}(D)z_0, p_{k-s+2}(D)z_0, \ldots, p_k(D)z_0]$ so that $R = VZ$. Using this notation,

$$\bar{r} = [I - VDZ(Z^TDZ)^{-1}Z^TV^T]r = V\bar{z}, \tag{5.24}$$

with

$$\bar{z} = [I - DZ(Z^TDZ)^{-1}Z^T]p_k(D)z_0. \tag{5.25}$$

The first question to be answered is if the projection step eliminates eigenvectors or eigenvalues from the residual, e.g., if λ is a repeated eigenvalue, does the projection step with $s = 1$ remove one eigenvector corresponding to λ or all such eigenvectors?

Lemma 5.1 *Let* $w = (Z^T D Z)^{-1} Z^T p_k(D) z_0$*. After the BST projection step the* i^{th} *eigencomponent of* $\bar{r}$ *is given by*

$$\bar{z}^{(i)} = \{p_k(\lambda_i) - \lambda_i [p_{k-s+1}(\lambda_i), \ldots, p_k(\lambda_i)] w\} z_0^{(i)} \tag{5.26}$$

Proof: Let e_i be the i^{th} coordinate unit vector. From (5.25),

$$\bar{z}^{(i)} = e_i^T \bar{z} = p_k(\lambda_i) z_0^{(i)} - \lambda_i e_i^T Z (Z^T D Z)^{-1} Z^T p_k(D) z_0 \tag{5.27}$$

Noting that $e_i^T Z$ is the i^{th} row of Z,

$$e_i^T Z = [p_{k-s+1}(\lambda_i), \ldots, p_k(\lambda_i)] z_0^{(i)} \tag{5.28}$$

and so

$$\bar{z}^{(i)} = \{p_k(\lambda_i) - \lambda_i [p_{k-s+1}(\lambda_i), \ldots, p_k(\lambda_i)] w\} z_0^{(i)} \tag{5.29}$$

□

Theorem 5.1 *If the BST projection step changes one eigencomponent of the residual corresponding to* λ *by a factor* η*, then all other eigencomponents corresponding to* λ *are also changed by a factor of* η*.*

Proof: Let

$$\eta_i = \{p_k(\lambda_i) - \lambda_i [p_{k-s+1}(\lambda_i), \ldots, p_k(\lambda_i)] w\} \tag{5.30}$$

as in the lemma, and note that if $\lambda_i = \lambda_j$, then $\eta_i = \eta_j$. □

Corollary 5.1 *If the projection step reduces the* i^{th} *eigencomponent of the residual to zero and* $\lambda_j = \lambda_i$*, then the* j^{th} *eigencomponent is also reduced to zero.*

What effect does the BST projection step have if two eigenvalues are close but not equal to each other? The next theorem shows that if they are within a distance ϵ of each other, then the factors by which the corresponding eigencomponents of the residual are reduced are bounded by a constant times ϵ.

Theorem 5.2 *Suppose that* $\mid \lambda_i - \lambda_j \mid < \epsilon$ *and* $\lambda_i \leq \lambda_j < \nu$. *Then for* $\epsilon < \phi^{-1}$ *where*

$$\phi = \frac{\sqrt{\nu - \lambda_j} + \sqrt{1 - \nu}}{2(1 - \nu)(\nu - \lambda_j)}, \tag{5.31}$$

$$\left| \frac{\bar{z}^{(i)}}{z_0^{(i)}} - \frac{\bar{z}^{(j)}}{z_0^{(j)}} \right| \leq \epsilon[1 + 2k\phi \cosh(k\phi\epsilon)](1 + \sqrt{s} \parallel w \parallel) \tag{5.32}$$

The proof of this result is long, uninstructive, and placed in Appendix G. The constant above grows rapidly with k, since larger values of k separate out the eigenvalues of $I - Q$ in $(0, \nu)$ more than small values do (see Figure 5.2.) The constant also grows with s, because larger values of s introduce older residuals into R which do not approximate the eigenspace corresponding to $(0, \nu)$ as well as later ones do. While these results are encouraging, there is no indication of whether the factor η_i by which the i^{th} eigencomponent is changed on the projection step has $|\eta_i| < 1$, giving a reduction, or $|\eta_i| > 1$, causing growth in that component. In order to analyse this further, first the motivating argument that the residuals tend to approximate $v_1, v_2, \ldots, v_x$ is quantified.

Lemma 5.2 *Let* $S = \text{span}\{v_i : \lambda_i = \lambda_1\}$ *and suppose that* $z_0^{(i)} \neq 0$ *for some* i *such that* $\lambda_i = \lambda_1$. *If* $\lambda_1 < \nu$, *then in the Chebyshev*(x_0, ν, K) *process*

$$\lim_{k \to \infty} r_k / p_k(\lambda_1) = v \in S \tag{5.33}$$

Proof: $V^T r_k = V^T p_k(I - Q) V z_0 = p_k(D) z_0$, so the i^{th} component of $V^T r_k / p_k(\lambda_1)$ is

$$\frac{p_k(\lambda_i)}{p_k(\lambda_1)} = \frac{T_k((\delta - \lambda_i)\gamma^{-1})}{T_k((\delta - \lambda_1)\gamma^{-1})} \tag{5.34}$$

If $\lambda_i \geq \nu$, then $|T_k((\delta - \lambda_i)\gamma^{-1})| \leq 1$ while $T_k((\delta - \lambda_1)\gamma^{-1}) \to \infty$ and so the corresponding eigencomponent of r_k goes to zero. For $\lambda_i < \nu$,

$$T_k((\delta - \lambda_1)\gamma^{-1}) = \cosh \left[k \ln \left(\frac{(\sqrt{1 - \lambda_i} + \sqrt{\nu - \lambda_i})^2}{1 - \nu} \right) \right] \tag{5.35}$$

The argument of the logarithm above is a decreasing function of i, so

$$\frac{T_k((\delta - \lambda_i)\gamma^{-1})}{T_k((\delta - \lambda_1)\gamma^{-1})} \to \begin{cases} 1, & if \lambda_i = \lambda_1 \\ 0, & if \lambda_1 < \lambda_i < \nu \end{cases} \tag{5.36}$$

Multiply $V^T r_k / p_k(\lambda_1)$ by V and use $VV^T = I$, (5.34), and (5.36) to find

$$r_k / p_k(\lambda_1) \rightarrow \sum_{\lambda_i = \lambda_1} z_0^{(i)} v_i \in S \tag{5.37}$$

$\Box$

Immediately it follows that r_k does tend to approximate v_1 as $k \rightarrow \infty$ if $\lambda_1 < \lambda_2$:

Corollary 5.2 *If* $\lambda_1 < \lambda_2$, *then*

$$\| r_k \| \rightarrow \begin{cases} p_k(\lambda_1) z_0^{(1)}, & if \;\; \lambda < \nu \\ 0, & if \;\; \lambda \geq \nu \end{cases} \tag{5.38}$$

Corollary 5.3 *If* $\lambda_1 < \lambda_2$, *then*

$$r_k / \| r_k \| \rightarrow \begin{cases} v_1, & if \;\; \lambda < \nu \\ 0, & if \;\; \lambda \geq \nu \end{cases} \tag{5.39}$$

Now consider the case where $s = 1$. The effect of the projection step on the eigencomponents in (5.27) becomes

$$\bar{z}^{(i)} = p_k(\lambda_i) \left[1 - \lambda_i \frac{r_k^T r_k}{r_k^T (I - Q) r_k} \right] z_0^{(i)} \tag{5.40}$$

The first factor gives the amount by which the Chebyshev(x_0, ν, K) process reduces the i^{th} eigencomponent, while the second factor shows the effect of the projection step. When $r_k = v_i$, the factor is zero and hence the i^{th} eigencomponent is eliminated, which is precisely the motivation given for BST in [49]. However, when $\lambda_1 < \lambda_2$ the last corollary shows that $r_k / \| r_k \| \approx v_1$ and so

$$\frac{r_k^T r_k}{r_k^T (I - Q) r_k} \approx \frac{1}{v_1^T (I - Q) v_1} = \lambda_1^{-1} \tag{5.41}$$

For $i = N$ in particular

$$\bar{z}^{(N)} \approx p_k(\lambda_N) \left[1 - \frac{\lambda_N}{\lambda_1} \right] z_o^{(N)} \tag{5.42}$$

and the projection step can increase the N^{th} eigencomponent of the residual by a factor close to $\kappa(I-Q)$. Furthermore, the better r_k approximates v_1, which is the goal of the Chebyshev(x_0, ν, K) process, the closer this growth factor comes to $\kappa(I-Q)$. In general,

$$\bar{z} \in \text{span}[z_k, Dz_k, Dz_{k-1}, \ldots, Dz_{k-s+1}] \tag{5.43}$$

with $z_k = p_k(D)z_0 \approx e_1$, the first unit vector. Since $D = (\lambda_1, \lambda_2, \ldots, \lambda_N)$, the coefficients of the vectors Dz_j must be extremely large in absolute value in order to eliminate the first few components of z_k. Such a linear combination necessarily will cause the last few components of $\bar{z}$ to be large, in practice by an amount close to $[\kappa(I-Q)]^s$. Thus there is a tradeoff in BST acceleration: If the first few eigenvectors are well approximated by r_k, the projection step greatly reduces the first few eigencomponents but simultaneously increases the last few eigencomponents by a factor close to $[\kappa(I-Q)]^s$. However, if the first few eigenvectors are poorly approximated the corresponding eigencomponents remain sizable after the projection step and the procedure can converge almost as slowly as simple iteration with Q.

In summary, iterated BST acceleration is convergent for any selection of the parameter $\nu \in (0,1)$. At worst no acceleration is provided and the method behaves like simple iteration. At best the convergence can match that of Chebyshev acceleration applied to a problem with spectrum in $(\nu, 1]$ rather than $(\lambda_1, 1]$. The projection step can remove up to s eigenvalues from the residual's eigendecomposition each time. Unfortunately the components corresponding to eigenvalues in $(\nu, 1]$ can actually increase on the projection step, by an amount as large as $\kappa(I-Q)$. However, for problems with a few extremely small eigenvalues, BST acceleration can potentially outperform Chebyshev acceleration.

5.3 CG Acceleration

CG acceleration for RP methods was proposed in [6] and tested in [38, 39, 9]. CG acceleration requires no parameter estimation and is an optimal process, implicitly finding the optimal residual polynomial. The basic algorithm is:

1) Initialize:

$$\begin{array}{rcl} x_0 & = & \text{given} \\ \epsilon = \epsilon_\rho & > & 0 \\ d_0 & = & r_0 = \tilde{b} - (I - Q)x_0 \\ \rho_0 & = & r_0^T r_0 \\ k & = & 0 \end{array} \tag{5.44}$$

2) Iterate:

$$\begin{array}{rrcl} & w_k & = & (I - Q)d_k \\ & \alpha_k & = & \rho_k / d_k^T w_k \\ & x_{k+1} & = & x_k + \alpha_k d_k \\ & r_{k+1} & = & r_k - \alpha_k w_k \\ & \rho_{k+1} & = & r_{k+1}^T r_{k+1} \\ \text{If} & \rho_{k+1} & < & \epsilon_\rho \ \text{ go to (4)} \end{array} \tag{5.45}$$

3) Update d_k:

$$\begin{array}{rcl} \beta_{k+1} & = & \rho_{k+1}/\rho_k \\ d_{k+1} & = & r_{k+1} + \beta_{k+1} d_k \\ & \text{Go to (2)} & \end{array} \tag{5.46}$$

4) Stopping Test:

If $\| b - Ax_{k+1} \|^2 < \epsilon$, stop.

Otherwise, set $\epsilon_\rho = \epsilon_\rho \rho_{k+1}/\epsilon$.

Return to (3).

The stopping test uses the true residual of the system, not the pseudo-residual. The special properties of the Kaczmarz system can be combined with those of the CG algorithm to reduce the amount of work per iteration and assure that x_k exactly satisfies the first block of equations on every iteration. This follows from:

Theorem 5.3 *Suppose that CG algorithm is applied to the Kaczmarz system*

$$(I - Q_\pi)x = b_\pi \tag{5.47}$$

If $x_0 = b_\pi$ is chosen as the starting vector, then r_k and d_k are in range$(I - P_1)$ for all $k \geq 0$.

Proof:

$$\begin{aligned}r_0 &= b_\pi - (I - Q_\pi)b_\pi = Q_\pi b_\pi \\ &= (I-P_1)(I-P_2)\cdots(I-P_m)\cdots(I-P_2)(I-P_1)b_\pi \in \mathrm{range}(I-P_1)\end{aligned} \tag{5.48}$$

Since $d_0 = r_0$, the same is true for d_0. Suppose the theorem holds for $k-1$. Then $d_{k-1} = (I - P_1)d_{k-1}$ and so

$$\begin{aligned}w_k &= (I - Q_\pi)d_{k-1} \\ &= (I-P_1)d_{k-1} \\ &\quad -(I-P_1)(I-P_2)\cdots(I-P_m)\cdots(I-P_2)(I-P_1)d_{k-1} \\ &\in \mathrm{range}(I-P_1)\end{aligned} \tag{5.49}$$

Since r_k is a linear combination of r_{k-1} and w_k, $r_k \in \mathrm{range}(I - P_1)$. Since d_k is a linear combination of d_{k-1} and r_k, $d_k \in \mathrm{range}(I - P_1)$. The result follows by induction. □

One projection per iteration can be saved, since the first multiplication by $(I - P_1)$ when forming $(I - Q_\pi)d_k$ can be omitted. This also keeps the first block of equations satisfied in exact arithmetic:

Corollary 5.4 *If $x_0 = b_\pi$ in the CG algorithm, then $A_1^T x_k = b_1$ for all $k \geq 0$.*

Proof: From the definition of b_π in (2.22)

$$b_\pi \in (A_1^T)^\dagger b_1 + \mathrm{range}(I - P_1) \tag{5.50}$$

Since $A_1^T(I - P_1) = A_1^T(I - A_1 A_1^\dagger) = 0$, $A_1^T x_0 = A_1^T(A_1^T)^\dagger b_1 = b_1$ because A_1 has full column rank. For $k > 0$, $d_{k-1} \in \mathrm{range}(I - P_1)$, so

$$A_1^T x_k = A_1^T(x_{k-1} + \alpha_k d_{k-1}) = A_1^T x_{k-1} = \cdots = A_1^T x_0 = b_1 \tag{5.51}$$

□

In [9] it was shown how to use these results in the $m = 2$ case to update the true residual and not require additional multiplications by A in doing so. That technique, however, is not as useful for $m = 9$ and larger because computing the true residual requires multiplying by the block rows A_i^T, $i = 3, 4, \ldots, m$. The corollary, as the unaccelerated Kaczmarz iteration does, allows equations deemed more important than others to be placed in the first block A_1^T and kept satisfied on every iteration.

The V-form system has already been shown to allow an explicit reduction of the problem size. The second major theoretical advantage of the V-form system is that when accelerated by CG, the resulting algorithm is error-minimizing in the two-norm:

Theorem 5.4 *Suppose CG acceleration is applied to the system $(I - V_\pi)z = b_V$ where V_π is given by (5.3) and b_V is defined by (2.29). Let x^* be the solution to $Ax = b$. Then the k^{th} iteration minimizes $\| x - x^* \|$ over all $x \in S_V^T * \text{span}[x_0 - x^*, \ldots, x_{k-1} - x^*]$*

Proof: From the relation between x_k and z_k given in (2.33),

$$x_k = A^T(D+L)^{-T}R^T z_k = S_V z_k, \tag{5.52}$$

where $V_\pi = I - S_V^T S_V$. The CG algorithm minimizes the quadratic form

$$\begin{aligned} z^T(I - V_\pi)z \quad - \quad & 2z^T b_V \\ = \quad & z^T S_V^T S_V z - 2z^T S_V^T A^{-1} b \\ = \quad & x^T x - 2x^T x^* \\ = \quad & \| x - x^* \|^2 \quad - \quad \| x^* \|^2 . \end{aligned} \tag{5.53}$$

Since $\| x^* \|^2$ is a constant, this is equivalent to minimizing $\| x - x^* \|^2$. The subspace over which this minimization occurs is

$$\begin{aligned} &\text{span}[r_0, \ldots, r_{k-1}] \\ &= \text{span}[S_V^T A^{-1} b - S_V^T S_V z_0, \ldots, S_V^T A^{-1} b - S_V^T S_V z_{k-1}] \\ &= S_V^T * \text{span}[x^* - x_0, \ldots, x^* - x_{k-1}] \end{aligned} \tag{5.54}$$

□

In contrast CG acceleration of the Kaczmarz system minimizes $\| S_V^T(x_k - x^*) \|$ over the subspace $S_V S_V^T * \text{span}[x_0 - x^*, \ldots, x_{k-1} - x^*]$, a result established in [6].

Chapter 6

Testing Results

The RP algorithms developed and proposed in this thesis have the goals of robustness and efficiency, in that order. This chapter presents numerical results that show both goals are achieved and provides the basis for further analysis of RP methods.

First the two machines that the testing is performed on are described. Next the RP algorithms are described in greater detail, and the other non-symmetric solvers to which the RP methods are compared are presented. The stopping tests used are defined and then the results comparing all of the methods are shown. A more detailed comparison between the RP methods is made, and this offers a theoretical explanation for their relative behaviour. The use of BST acceleration for the Kaczmarz RP algorithm is then described and test results are presented comparing it to CG acceleration. The partitioning scheme used here is the $m = 9$ scheme as defined in Chapter 3 and the primary acceleration scheme used is CG.

6.1 The Alliant FX/8 and Cray-2

The testing was mainly done on an Alliant X/8 at the Center for Supercomputing Research and Development of the University of Illinois, with additional tests run on one CPU of the Cray-2 at the National Center for Supercomputer Applications, also of the University of Illinois.

The FX/8 is a shared memory multiprocessor with 8 computational elements (CEs). Each CE has vector processing ability, with 8 vector registers containing 32 64-bit words each. Parallel execution of instructions is governed by a concurrency control bus, and the access to memory by the CE's is through a crossbar switch connected to a shared cache containing 16K 64-bit words. The maximum possible bandwidth between the cache and the CE's is 47 MW/sec. The cache in turn is connected to a 4MW main memory by a bus with a maximum possible bandwidth of 23.5 MW/sec. I/O and other noncomputational tasks are handled separately by 6 interactive processors which access memory through other caches.

The programs are written in Alliant's FX Fortran which permits automatically optimized code to be generated by the compiler, the user to specify where concurrency and/or vectorization should be allowed, or any combination of the two. Loops can be executed in several modes. Vector mode simply uses the vector registers and arithmetic units on one CE. Vector-concurrent mode divides iterations into blocks of 32 each and assigns them to individual CE's, where they run in vector mode simultaneously. Nested loops can run in concurrent-outer/vector-inner mode where the inner loop executes in vector mode and the outer loop in concurrent mode. Compiler directives also allow subroutine calls to be made concurrently.

Additional testing was performed on one processor of a four processor Cray-2 system. The Cray-2 has 128 Mwords of central memory arranged in four quadrants with 128 interleaved banks. A foreground processor controls data flow between memory and I/O devices. Each of the four computational processors has a single port to memory and can access all of memory. The total bandwidth to memory for all four processors is 1 GW/sec. Each processor has scalar and vector functional units, eight vector registers of length 64 words, and 2^{14} words of fast local memory used as a register file for scalar operands and temporary storage for vectors. Although the Cray-2 can be used as a multiprocessor, only one processor was used in the testing in order to demonstrate that RP methods can be vectorized efficiently.

Although the Alliant FX/8 has vector capabilities with a vector length of 32, the use of vectorization on nontrivial programs usually decreases execution time only by a factor of 2 or 3. Concurrent execution of programs, on the other hand, often decreases execution time by a factor of 5 or 6. Hence the parallelism available in an algorithm is of more importance than vectorization on the Alliant. For one processor of the Cray-2, however, vectorization is essential for good performance. The testing results show that RP methods can be implemented efficiently on both architectures.

6.2 Description of RP Algorithms

The RP algorithms and partitioning schemes are as presented in previous chapters, but a more detailed description of the implementations is given here. The crucial operations are the preprocessing stage, computation of the modified right hand side vectors, and forming the matrix-vector products needed for the acceleration schemes. The V-form system also requires a post-processing stage to recover the solution x from the auxillary unknown z. For convenience, the methods are referred as RP-9 for the Kaczmarz form with $m = 9$, V-RP for the V-form with $m = 9$, and SUM for the sum form with $m = 9$.

6.2.1 Preprocessing for RP Methods

Preprocessing for RP methods consists of two phases, computing the necessary Cholesky factors and then computing the necessary modifications of the right hand side vector b. As described in Chapter 3, each projection $y = P_i x$ is computed as $n^2/9$ simultaneous tasks, each task requiring the computation of a projection onto range(C_j) with the form of C_j^T given by (3.12). From the results of Chapter 4, each C_j is well-conditioned, so the procedure is to explicitly form the normal equations and then perform an LDL$^{\mathrm{T}}$ factorization. Recalling that S_i is the set of indices j for which $C_i^{(j)}$ is a subblock of A_i, this stage consists of:

Cholesky Factorization on Alliant:

For $i = 1, 2, \ldots, 9$ and $j \in S_i$ *(in parallel)*

> Form $R_i^{(j)} = (C_i^{(j)})^T (C_i^{(j)})$ *(vectorized)*
> Perform LDL$^{\mathrm{T}}$ factorization on $R_i^{(j)}$, overwriting $R_i^{(j)}$ with the result. *(sequential)*

End For

In practice $R_i^{(j)}$ is stored as three vectors, and the one defining the diagonal is stored as $(D_i^{(j)})^{-1}$ so that the backsolves can be performed using multiplication rather than division. The LDL$^{\mathrm{T}}$ factorization is an inherently sequential process, but this is not a serious detriment on the Alliant FX/8. For the Cray-2, however, it is important that this be vectorizable and so the order of the loops is reversed on that machine. Let n_{ne} be the number of operations needed to form the normal equations for one subblock $(C_i^{(j)})$, and let n_{ldl} be the number of operations required to perform the LDL$^{\mathrm{T}}$ factorization on one subblock.

Cholesky Factorization on Cray-2:

For $k = 1, 2, \ldots, n_{ne}$ *(sequential)*

> Perform the k^{th} operation for forming $R_i^{(j)} = (C_i^{(j)})^T (C_i^{(j)})$ *(vectorized across all i and j)*

End For

For $k = 1, 2, \ldots, n_{ldl}$ *(sequential)*

> Perform the k^{th} operation for performing a LDL$^{\mathrm{T}}$ factorization on $R_i^{(j)}$ *(vectorized across all i and j)*

End For

Three important points must be made about the Cray-2 vectorized implementation. If the matrix is set up and stored by diagonals with the natural ordering of nodes so that the nonzero structure is given by Figure 3.1, the stride between elements forming the vectors of the vectorized are constant until a plane boundary is reached, when a jump in the vector index occurs. This is remedied by storing the matrix with the node numbering induced by the ordering shown in Figure 3.3, or equivalently, by performing the same permutation on the columns of A as is performed on the rows when defining the row partitioning. The second point is not as easily handled. When more general problems such as elliptic partial differential equations on irregular domains are solved, the number and size of the blocks defined by lines and planes varies through the matrix. In this case the vectorization proposed above is no longer valid and other schemes must be used. The final point is that for the seven point difference operator matrices used here, the vector lengths are $n^2/9$. For efficient vector processing on a Cray the lengths should be at least 64, which occurs here when a $24 \times 24 \times 24$ grid is used.

Care must be used in computing the modified right hand side vector $\tilde{b}$ from the original right hand side vector b since, e.g., a naive implementation using (2.22) can require $(3m^2 + m)/2 = 126$ projections and 10 vectors for temporary storage. Let

$$R_i = \mathrm{diag}(R_i^{(j)}), \quad j \in S_i \tag{6.1}$$

By careful overwriting the computation of the modified right hand side b_π can be implemented using only one temporary vector $w^T = (w_1^T, w_2^T, \ldots, w_m^T)$ of length N and $4m - 2$ triangular backsolves with R_i or R_i^T, $2m - 2$ multiplications by R_i or R_i^T, $3m$ multiplications by A_i and $m^2 - m$ by A_i^T. The predominant expense is the triangular backsolves, which is the same as that required by one iteration with the Kaczmarz system. Furthermore, as in the Cholesky decomposition stage each of the matrix-vector operations is implemented with $n^2/9$ concurrency on the Alliant or vectors of length $n^2/9$ on the Cray-2 so that in this and all of the following algorithms each operation involving A_i or $R_i^{(j)}$ is actually performed in parallel on the subblocks $C_i^{(j)}$ and $R_i^{(j)}$.

Computation of b_π from b:

$w = b$

For $i = 1, 2, \ldots, m$

$w_i = (R_i^T R_i)^{-1} w_i$
$b_\pi = A_i w_i$
For $j = i + 1, i + 2, \ldots, m$

$w_j = w_j - A_j^T b_\pi$

End For

End For

For $i = 1, 2, \ldots, m-1$

$w_i = R_i^T R_i w_i$

End For

For $i = m, m-1, \ldots, 1$

$b_\pi = A_i w_i$

For $j = 1, 2, \ldots, i-1$

$w_j = w_j - A_j^T b_\pi$

End For

If $i > 1$, $w_{i-1} = (R_{i-1}^T R_{i-1})^{-1} w_{i-1}$

End For

$b_\pi = A_1 w_1$

For $i = 2, 3, \ldots, m$

$b_\pi = b_\pi + A_i w_i$

End For

The V-form modified right hand side b_V can be computed at a slightly lower cost, requiring $3m-2$ triangular backsolves by R_i^T or R_i. Let $Q_i = A_i R_i^{-1}$, and where Q_i is used in the following algorithm it should be implemented as a backsolve using R_i followed by a multiplication by A_i.

Computation of b_V from b:

For $i = 1, 2, \ldots, m$

$$(b_V)_i = R_i^{-T} b_i$$

End For

$w = Q_m (b_V)_m$

For $i = m-1, m-2, \ldots, 2$

$$(b_V)_i = (b_V)_i - Q_i^T w$$
$$w = w + Q_i (b_V)_i$$

End For

$(b_V)_1 = (b_V)_1 - Q_1^T w$

Again, each multiplication by A_i, A_i^T, R_i^{-1}, or R_i^{-T} is implemented with $n^2/9$ concurrency or vector length.

Computing the modified right hand side b_σ for the sum form is more straightforward than for the product forms, and requires only $2m$ triangular backsolves and m multiplications by A_i:

Computation of b_σ from b:

$b_\sigma = A_1 (R_1^T R_1)^{-1} b_1$

For $i = 2, 3, \ldots, m$

$$b_\sigma = b_\sigma + A_i (R_i^T R_i)^{-1} b_i$$

End For

6.2.2 Computation of Matrix-Vector Products for CG

For the Kaczmarz system, products $v = (I - Q_\pi)d$ are computed with a cost of $4m - 2$ triangular solves and $2m - 1$ multiplications by A_i or A_i^T.

Computation of $v = (I - Q_\pi)d$:

$v = d$

For $i = 1, 2, \dots, m, m-1, \dots, 1$

$$v = v - A_i(R_i^T R_i)^{-1} A_i^T v$$

End For

$v = d - v$

The cost for the V-form system is slightly less, requiring only $4m - 4$ triangular solves and $2m - 2$ multiplications by A_i or A_i^T.

Computation of $v = (I - Q_V)d$:

$w = Q_m d_m$

For $i = m-1, \dots, 3, 2$

$$w = w + Q_i(d_i - Q_i^T w)$$

End For

$w = w - A_1(R_1^T R_1)^{-1} A_1^T w$

For $i = 2, \dots, m-1$

$$v_i = Q_i^T w$$
$$w = w - Q_i v_i$$

End For

$v_m = Q_m^T w$

Note that this only uses the block components numbered from 2 to m of the search direction vector d and the product vector v since the V-form has an explicit reduction in the problem size.

The matrix-vector product needed for the sum form is simple to implement, but an important point about the following algorithm is that only one additional vector for temporary storage is used. If 9 vectors are used, each of the projections can be computed in parallel, as well as having $n^2/9$ parallelism available within the computation of each projector.

Computation of $v = (I - Q_\sigma)d$:

$w = A_1(R_1^T R_1)^{-1}d$

For $i = 2, 3, \ldots, m$

$w = w + A_i(R_i^T R_i)^{-1}d$

End For

The cost of this multiplication is only half that of the product forms, requiring $2m$ triangular solves and m multiplications by A_i.

6.2.3 Postprocessing

Both the Kaczmarz and sum forms find x directly and require no postprocessing. The V-form must retrieve x from the auxillary vector z, and uses $2m - 1$ triangular solves and $2m - 2$ multiplications by A_i or A_i^T to do this.

Retrieval of x from z for V-form:

$z_1 = (b_V)_1$

$x = Q_m z_m$

For $i = m-1, \ldots, 1$

$$x = x + Q_i(z_i - Q_i^T x)$$

End For

6.3 Description of Other Methods Tested

The RP algorithms are compared to two other basic methods, GMRES(10) and CG on the normal equations $A^T Ax = A^T b$. The latter is refered to as CGNE. GMRES(10) is from the PCGPAK library and has been optimized for the Alliant FX/8 by E. Anderson and for the Cray-2 by Scientific Computing Associates. More details about these implementations of GMRES(10) can be found in [52]. The fixed value of $k = 10$ is chosen because larger values do not significantly improve the robustness of GMRES(k) until k becomes a significant fraction of the problem size, and because iterative algorithms with a small fixed number of vectors for additional storage are of primary interest in this thesis. For simplicity, whenever GMRES is refered to without an argument, GMRES(10) is understood. The reasons for selecting GMRES(k) instead of another Krylov subspace method are first the popularity of GMRES(k), and secondly that Orthomin(k) and GCR(k) give similar results to GMRES(k).

GMRES(10) is implemented also with ILU and MILU preconditioners. The versions of these preconditioners are the ones that require an additional amount of storage equal to the number of nonzeros in the matrix A, as opposed to the versions that require only one additional vector of storage. See [28] for a description of these preconditioners, where the high versus low memory requirement versions are distinguished by using asterisk, for example, ILU versus ILU*. Because only GMRES(k) is implemented with these preconditioners, in the sequel the combination of GMRES(10) with ILU preconditioning is abbreviated ILU here, and mutatis mutandi for MILU. Although these two preconditioners can be shown to exist for M-matrices [45], that need not be the case for the test problems. Even when the preconditioners exist, the preconditioned system may actually be more difficult to solve than the unpreconditioned one. This is shown to occur in the test results. Finally, PCGPAK allows the user to pass a parameter to the preconditioning routine that helps guard against failure of the preconditioner. The parameter corresponds to an amount to be added to diagonal elements of the preconditioner during the factorization to increase the diagonal dominance. This parameter is set to 0 in the experiments because only routines with automatic parameter estimation are considered here, and in practice several tries at the incomplete factorization may be necessary to find a workable value for the parameter.

CGNE is also implemented with a preconditioner, found by performing an ILU* factorization on A to get $A \approx LU$. Then CG is applied to the normal equations of the left-preconditioned system $(LU)^{-1}Ax = (LU)^{-1}b$. When combined with this preconditioner CGNE is denoted simply as ICCG. Note that this is not the same as forming the normal equations A^TA and then performing an incomplete Cholesky factorization of A^TA. Earlier work [9] with such a preconditioner for CGNE applied to two dimensional problems showed that it also suffered robustness problems.

All of the methods tested, along with their additional memory requirements, are listed in Table 6.1. With the exception of GMRES(10), all of the methods work on a modified right hand side vector and so storage must also be provided for it.

6.4 Stopping Tests

The primary stopping test used is

$$\| Ax_k - b \|^2 \leq \epsilon \tag{6.2}$$

Acronym	Method	Storage
RP-9	CG accelerated Kaczmarz ($m = 9$)	$7N$
V-RP	CG accelerated V-form ($m = 9$)	$8N$
SUM	CG accelerated Sum form ($m = 9$)	$8N$
GMRES	GMRES(10) from PCGPAK	$13N$
ILU	GMRES(10) with ILU precond	$21N$
MILU	GMRES(10) with MILU precond	$21N$
CGNE	CG applied to $A^T Ax = A^T b$	$5N$
ICCG	CGNE with ILU* precond	$6N$

Table 6.1: Methods Tested

where $\epsilon = 10^{-9}$. For the CG accelerated methods, initially the tolerance for the pseudo-residual is set to equal that on the residual, i.e., $\epsilon_\rho = 10^{-9}$. Recall that the pseudo-residual is $r_k = b_\pi - (I - Q_\pi)x_k$ while the true residual is $b - Ax_k$. If the pseudo-residual satisfies $\| r_k \|^2 \leq \epsilon_\rho$, the true residual is computed (at a cost of a multiplication with A) and (6.2) is checked. If (6.2) does not hold, then the CG tolerance is adjusted by the assignment

$$\epsilon_\rho \leftarrow (0.7\epsilon_\rho / \| Ax_k - b \|^2)\epsilon. \tag{6.3}$$

Thus if the pseudo-residual's norm is 10 times smaller than the true residual's norm, the tolerance is decreased by a corresponding factor of 0.07, where the additional factor 0.7 is present to prevent the adjustment being made too often. Once this is done, the CG iteration resumes. The preconditioned PCGPAK routines use the residual of the preconditioned system as the stopping test, so the same tolerance adjustment procedure is used with them. If the PCGPAK routines must be resumed this way, the preconditioner is not recomputed.

Additional stopping tests, corresponding to failure conditions, are also imposed. A maximum number of iterations is set to $\max\{N, 4001\}$ for the product forms RP-9 and V-RP, the preconditioned GMRES routines ILU and MILU, and the preconditioned CG on the normal equations routine ICCG. SUM and GMRES(k) are allowed $\max\{N, 8001\}$ iterations because they require only one sweep throught the matrix per iteration while the others require 2 or more sweeps. Unpreconditioned CGNE is also allowed up to 8001 iterations. These limits are generous and usually if a method fails because the maximum allowed iterations are reached, iterating further has no effect. The only exception to this is CGNE for problem 5 with $N = 1728$, which can succeed after 3066 iterations.

For the test problems there is no guarantee that the preconditioners will exist or provide a better system. The preconditioner is called *unstable* if a zero pivot occurs during the factorization or if the norm of the initial preconditioned residual is 30 orders of magnitude larger than that of the unpreconditioned residual. In the latter case experience has shown that performing the iterations always fails to sufficiently reduce the residual. An additional case labeled unstable preconditioning occurs when the iterations experience overflow because of the use of a preconditioner.

The PCGPAK routines have one more stopping test. If the residual does not chang by a difference of at least 10^{-6} over the course of 10 successive iterations, the residual is said to *stagnate*. Experience has shown that loosening this restriction by, e.g., requiring only a 10^{-9} change within 100 iterations, simply changes the reason for failure from stagnated residual to maximum iterations reached.

Finally, for the tests run on the Cray-2 a maximum CPU time limit of 20 minutes is set for all of the methods. This is done for budgetary reasons. These error conditions are summarized and given the following codes in the tables of results:

- MX: Maximum iterations reached
- RS: Residual stagnates
- UP: Unstable preconditioner
- TM: Maximum allowed CPU time reached

6.5 Comparison of RP Methods with Other Methods

The results comparing the RP methods with the other methods are tabulated in Appendix A. Each table shows the number of iterations, the amount of CPU time required, the norms of the true residual and error vectors at the end of the solution process, and a two-letter code from the last section indicating the kind of failure encountered if one occurred. Problems of orders $N = 1728$ and $N = 13824$ were tested on the Alliant FX/8, and of order $N = 216000$ on the Cray-2.

Robustness results are summarized in Tables 6.2 - 6.4.

The preconditioned methods failed over half the time, and problems with large off-diagonal elements were the source of most of those failures. This is because the ILU and ICCG methods ignore the fill-in positions when forming the incomplete factors and for matrices with large off-diagonal entries those positions are large. The surprising result is that MILU failed 74% of the time, because the MILU incomplete factorization accounts for the fill-in

Robustness of Methods for N = 1728									
	Problem								
Method	1	2	3	4	5	6	7	8	9
RP-9									
V-RP									
SUM									
GMRES					RS	RS		RS	
ILU			RS		RS	RS		RS	RS
MILU			MX		RS	RS	RS	RS	RS
CGNE						MX			
ICCG			MX		MX	UP	MX	UP	MX

Table 6.2: Robustness of Methods for $N = 1728$

Robustness of Methods for N = 13824									
	Problem								
Method	1	2	3	4	5	6	7	8	9
RP-9									
V-RP									
SUM									
GMRES					RS	RS		RS	
ILU			MX		RS	UP		RS	RS
MILU	RS	RS	UP		RS		RS	RS	RS
CGNE					MX	MX		MX	
ICCG			UP		UP	UP		MX	MX

Table 6.3: Robustness of Methods for $N = 13824$

Robustness of Methods for N = 216000									
	Problem								
Method	1	2	3	4	5	6	7	8	9
RP-9								MX	
V-RP					MX			MX	
SUM								MX	
GMRES					RS	TM		RS	
ILU			UP		UP	UP		TM	
MILU	RS	TM	UP		RS		RS	RS	UP
CGNE					MX	MX		MX	
ICCG			UP		TM	UP		TM	TM

Table 6.4: Robustness of Methods for $N = 216000$

positions by moving their contributions to the main diagonal. For the test problems, however, this procedure creates incomplete factors that are not close approximations to the factors of A, which is revealed by noting that the initial residual of the MILU preconditioned system often is many orders of magnitude larger than that of the unpreconditioned system.

Unpreconditioned GMRES failed consistently on Problems 5, 6, and 8, for which A has an indefinite symmetric part. GMRES succeeds on Problems 3, 7, and 9 which also have indefinite symmetric parts, and this confirms that the positive definiteness of $A + A^T$ is sufficient but not necessary for convergence. Unpreconditioned CGNE failed on a total of 7 combinations of problems and sizes, primarily for Problems 5, 6, and 8. Graphs of the spectrum of A^TA for all of the problems with $N = 1728$ are in Appendix D, and an examination of those graphs shows Problems 5, 6, and 8 to be the ones with the poorest distribution of eigenvalues for the conjugate gradient algorithm, viz., there are many small eigenvalues. In spite of these failures CGNE is the most robust of the non-RP solvers, failing only 26% of the time. The total number and percentage of failures over all 27 combinations of problems and sizes is shown in Table 6.5.

Clearly RP-9 and SUM are the most robust of the methods, with V-RP close behind. The test problems were generally chosen to be especially challenging to existing iterative solvers, and may not be a representative mixture

Method	Number of Failures	Percent Failures
RP-9	1	3.7
V-RP	2	7.4
SUM	1	3.7
GMRES	9	33.3
ILU	14	51.9
MILU	20	74.1
CGNE	7	25.9
ICCG	16	59.3

Table 6.5: Number of Failures Among Methods

of realistic applications. Problem 4 was selected because for any mesh size it has a positive definite symmetric part, and as expected GMRES, ILU, and MILU succeed in solving it for each N. However, so does every other iterative method tested here. At the other extreme, Problem 8 was artificially created in an attempt to find problems for which RP-9 fails. Although this happens when N reaches 216000, every other iterative solver also fails in this case; this includes not only the solvers already discussed, but also GMRES(40) and Orthomin(10) from PCGPAK. Furthermore, after 4001 iterations with a final residual norm of 1.2×10^{-3} RP-9 comes closer to solving this problem than the next competitor, CGNE with a final residual of 3.5×10^{-3} after 8001 iterations. Additionally, all of the other non-RP methods fail on Problem 8 even for the coarser meshes.

V-RP failed on both Problems 5 and 8 when $N = 216000$. A more detailed analysis of these failures is presented in the next two sections, but for now it should be noted that on both of those problems V-RP produced a smaller error norm than RP-9, even though the residual norm is larger.

Can the robustness of the GMRES(k)-based methods be improved by increasing the number k of vectors stored? To answer this the algorithms were run on the smaller test problems for which they failed. The results are summarized in Tables 6.6 - 6.8.

Unpreconditioned GMRES(k) continues to fail on Problems 5, 6, and 8. ILU succeeds on Problem 9 with $k = 30$, and MILU succeeds on Problems

Effect of k on GMRES(k)									
N = 1728					N=13824				
	k					k			
Prob	10	20	30	40	Prob	10	20	30	40
5	RS	RS	RS	RS	5	RS	RS	RS	RS
6	MX	RS	RS	RS	6	MX	RS	RS	RS
8	RS	RS	RS	RS	8	RS	RS	RS	RS

Table 6.6: Effect of Increasing k on GMRES Robustness

Effect of k on ILU									
N = 1728					N=13824				
	k					k			
Prob	10	20	30	40	Prob	10	20	30	40
3	RS	RS	RS	RS	3	MX	RS	RS	RS
5	RS	RS	RS	RS	5	RS	RS	RS	RS
6	RS	RS	RS	RS	6	UP	UP	UP	UP
8	RS	RS	RS	RS	8	RS	RS	RS	RS
9	RS	RS			9	RS	RS	RS	RS

Table 6.7: Effect of Increasing k on ILU Robustness

Effect of k on MILU									
N = 1728					N=13824				
	k					k			
Prob	10	20	30	40	Prob	10	20	30	40
3	MX	RS	RS	RS	1	RS	RS	RS	RS
5	RS	RS	RS	RS	2	RS	RS	RS	RS
6	RS				5	RS	RS	RS	RS
7	RS	RS	RS	RS	7	RS	RS	RS	RS
8	RS	RS	RS	RS	8	RS	RS	RS	RS
9	RS	RS			9	RS	RS	RS	RS

Table 6.8: Effect of Increasing k on MILU Robustness

6 and 9 when $N = 1728$. Out of the 28 failure cases shown in the tables, however, increasing k improves robustness in only three cases, and only for the smaller problem size.

Why are RP methods more robust than the other algorithms? An explanation for CGNE is deferred to the next section, but for GMRES(10) a simple explanation is that there is no reason to expect the method to work unless $A + A^T$ is positive definite, and the results simply confirm that this condition is sufficient but not necessary for convergence. This explanation says more about the state of convergence theory for GMRES(k) than it does about the method itself. Each iteration of GMRES(k) minimizes the components of the residual contributed by the eigenvectors of A restricted to a Krylov subspace. If the residual is orthogonal to the Krylov subspace, the method stalls. Even if the residual is not orthogonal to the Krylov subspace, the minimization is over a subspace of dimension k, which becomes negligible for large N. RP methods, however, reduce the norm of the error by at least $\rho(Q) < 1$ on every iteration. For the product forms, moreover, every projection minimizes the residual on a subspace of dimension $N/9$, a fixed fraction of the problem size. Even though each iteration of RP algorithms requires more work than one iteration of GMRES(10), in balance a more robust method results.

The explanation for the preconditioned methods is equally simple: there is no reason to expect the factorization to exist, and even when it does exist only a heuristic argument suggests that the preconditioned system has a better eigenvalue distribution. The tests demonstrate that if A is not an M-matrix, the heuristic argument is invalid.

The robustness property is expected, since the RP methods were designed to achieve it. The surprising result is that in spite of the greater amount of work done per iteration, RP methods are the fastest of the solvers tested on 20 of the 27 experiments. Figure 6.1 shows the execution times of the methods for Problems 1, 4, and 7 with $N = 13824$. These problems are ones that most of the methods solved, although MILU fails on Problem 1 and ICCG fails on Problem 7. All three RP methods are faster than the remaining methods, and the Kaczmarz form V-RP is the fastest of the row projection methods. Both parallelism and data locality play a role in the speed of RP methods. Roughly 85% of the computation time of RP-9 is consumed in computing the matrix-vector products needed for the CG acceleration, and this has a natural parallelism of $n^2/9$ where $N = n^3$. Furthermore, the vector lengths for the Alliant FX/8 are n and for the Cray-2 implementation, $n^2/9$. GMRES computes the matrix-vector product with a parallelism of N, where each row can be computed separately. The vector length, however, is only 7. So, even though more parallelism is available for GMRES, the parallel tasks are small and synchronization costs predominate. This is partly because of the data structure that PCGPAK uses to store A, and GMRES can run faster if the matrix-vector product is performed by diagonals rather than rows.

Multiplication is performed by diagonals for CGNE, yet it was the fastest method on only 2 of the 27 combinations of problems and sizes. Part of the reason is the large number of iterations required, a consequence of the poor spectral distribution of A^TA compared to A. An equally important reason is data locality. The RP methods pull a block of $7n$ entries from A belonging to some submatrix $C_j^{(i)}$, and an additional block of $3n$ entries belonging to the corresponding Cholesky factor, into the CPU and/or cache and then perform two submatrix-vector products and solving two triangular systems before needing to access memory for the next block. CGNE, however, must stream all $7n^3$ entries of A through the CPU and/or cache twice for each iteration. Although RP methods actually need to make more data memory references on each iteration ($36n^3$ for RP-9 vs. $14n^3$ for CGNE), they naturally block the data into smaller fragments and then perform much more work with each fragment before writing it back to memory. This advantage can only

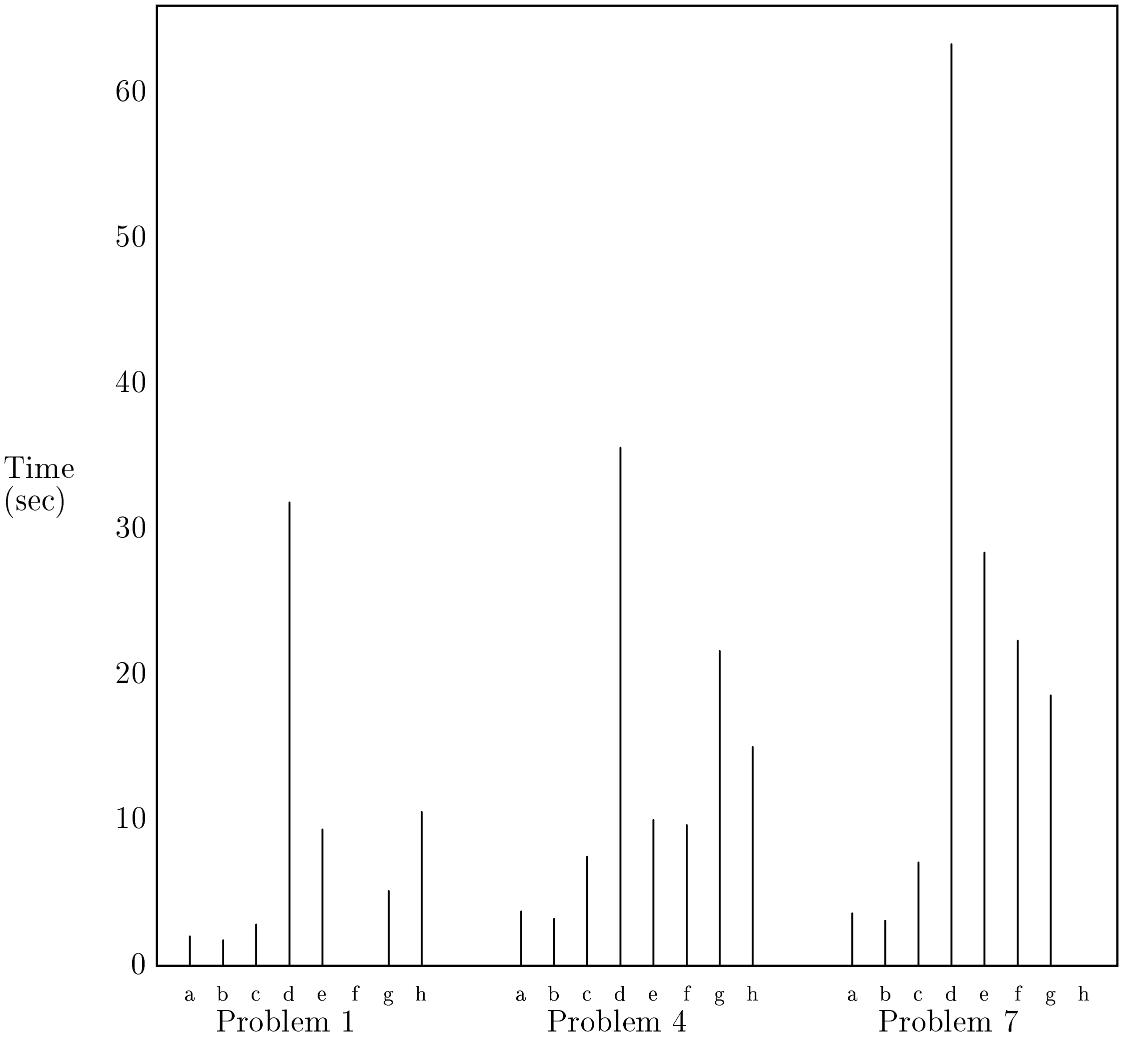


Figure 6.1: Execution Times for Problems 1, 4, and 7 with $N = 13824$; a:RP-9,b:V-RP,c:SUM,d:GMRES,e:ILU,f:MILU,g:CGNE,h:ICCG

be expected to increase with problem size. Figure 6.2 shows the ratio of execution time on 1 processor of the Alliant FX/8 to that on p processors for RP-9, averaged over all 9 problems. For comparison, a dashed line with slope 1 is included showing the ratio corresponding to perfect parallelism. On 8 processors RP-9 runs between 5 and 6 times faster, and the efficiency of 63% - 75% is because of the small size of the cache.

The main reason for the slowness of the incomplete factorization preconditioners is that they are inherently sequential processes, and not suitable for the computers used here. The second reason has already been mentioned, that for these problems the preconditioners do not closely approximate the matrix A.

6.6 Comparison of RP Methods with Each Other and CGNE

The preceding comparisons do not explain why in general RP-9 is faster than SUM which in turn is faster than CGNE. Heuristically an ill conditioned matrix A has some linear combination of rows approximately equal to the zero vector. For a row partitioned matrix, the near linear dependence can come *within* the blocks, that is, some linear combination of the rows within a particular block is approximately zero, or *between* the blocks, that is, the linear combination close to 0 must draw from more than one block row A_i^T. Now let $A_i = Q_i R_i$ be the QR decomposition of A_i, and note that both Q_i and $P_i = Q_i Q_i^T$ have the perfect condition number of 1, where condition number is defined as the ratio of largest to smallest nonzero singular values. It can be shown that RP-9, SUM, V-RP, and CGNE all apply the CG algorithm to a system with coefficient matrix $W^T W$, where the corresponding W^T is shown in Table 6.9.

Examining the matrices W shows that CGNE works on $A^T A$, and the ability to form a near linear dependence from both within and between blocks enters into the system matrix. SUM, however, replaces each A_i with Q_i, which has orthonormal columns. Hence SUM removes the near linear depen-

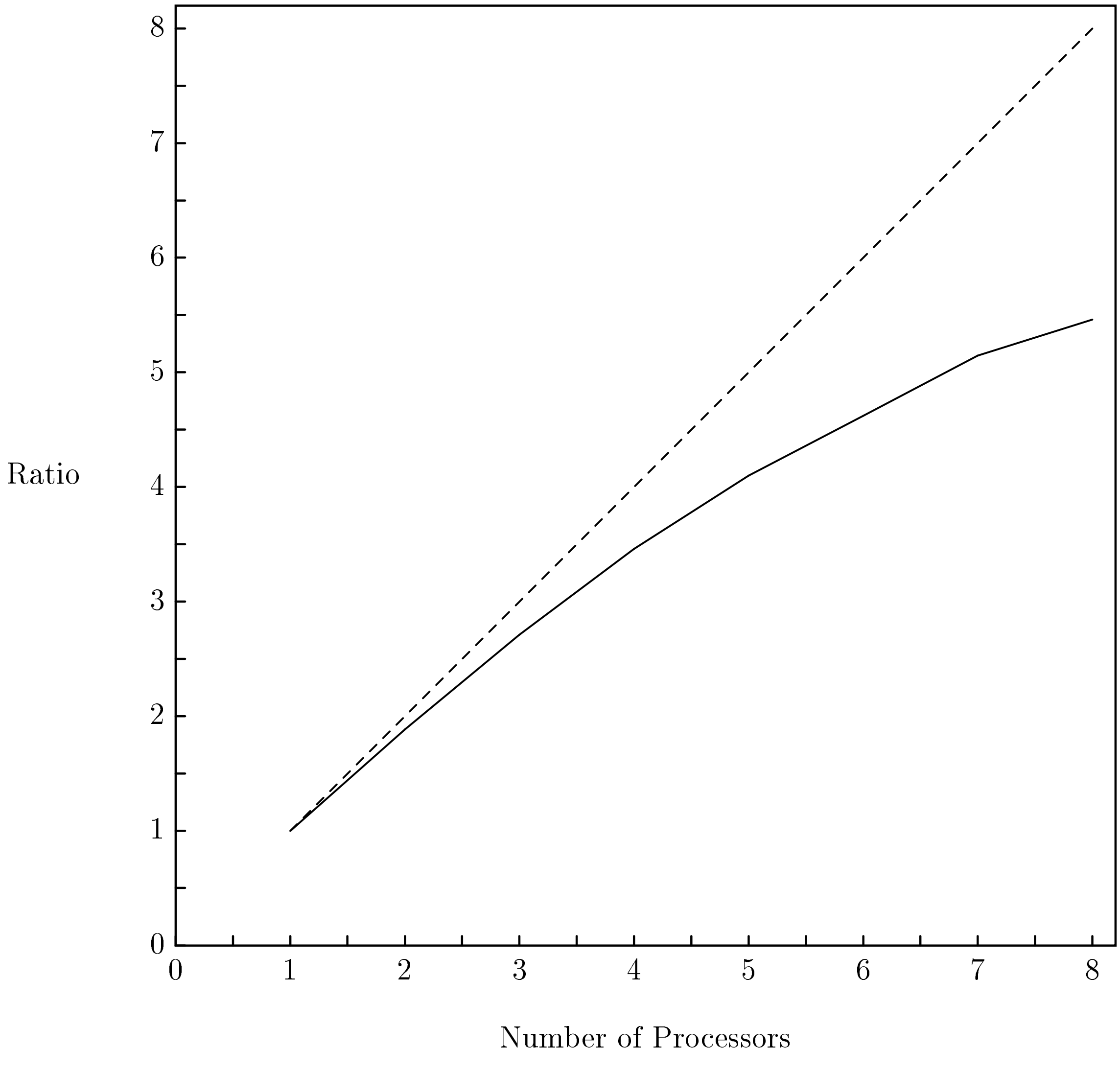


Figure 6.2: Execution Time Ratios for RP-9 on p Processors

Method	W^T
CGNE	$[A_1 \ \ A_2 \ \ \cdots A_m]$
SUM	$[Q_1 \ \ Q_2 \ \ \cdots Q_m]$
RP-m	$\left[P_1 \ \ (I-P_1)P_2 \ \ (I-P_1)(I-P_2)P_3 \ \ \cdots \prod_{i=1}^{m-1}(I-P_i)P_m\right]$
V-RP	$\left[Q_1 \ \ (I-P_1)Q_2 \ \ (I-P_1)(I-P_2)Q_3 \ \ \cdots \prod_{i=1}^{m-1}(I-P_i)Q_m\right]$

Table 6.9: Comparison of System Matrices for Four Methods

dence within the blocks, but has no effect on that between the blocks. RP-m also performs an orthornormalization within each block, 'replacing' A_i with the perfectly conditioned projector P_i. RP-m then goes a step further by recursively orthogonalizing between blocks in the following way: The first block column of W^T is orthogonal to the others since $P_1(I - P_1) = 0$. If the orthogonalizing $I - P_1$ factor is removed from the other block columns of W^T, then the second block column is orthogonal to the remaining ones since $P_2(I - P_2) = 0$. This process continues until the last block is reached. The recursive nature of this 'partial orthogonalization' can be seen by rewriting W^T as

$$W^T = [P_1 \quad (I-P_1)[P_2 \quad (I-P_2)[P_3 \quad (I-P_3)\cdots[P_{m-1} \quad (I-P_{m-1})P_m]\cdots]]] \tag{6.4}$$

Finally, V-RP has the same effect as RP-9, since $Q_i^T(I - P_i) = 0$ in the same way that $P_i(I - P_i) = 0$.

How accurate is this heuristic argument? One implication of it is that the system matrix spectrum for RP-9 should be better than that of SUM which in turn should be better than that of CGNE, where 'better' means having fewer small eigenvalues and many more eigenvalues near the maximal one. Figure 6.3 shows the spectra of the system matrices for these 3 methods for Problem 5 with $N = 1728$. Each spectrum has been normalized to have $\lambda_{\max} = 1$, and the normalized eigenvalues are plotted in increasing order versus the eigenvalue number. Clearly from the Figure RP-9 gives a spectrum favorable to CG acceleration, with SUM less favorable and CGNE the worst of the three. CGNE has such a poor distribution for this problem that it fails by reaching the maximum allowed iterations. A comparison of the spectra shown in the Appendices for these three methods demonstrates that this

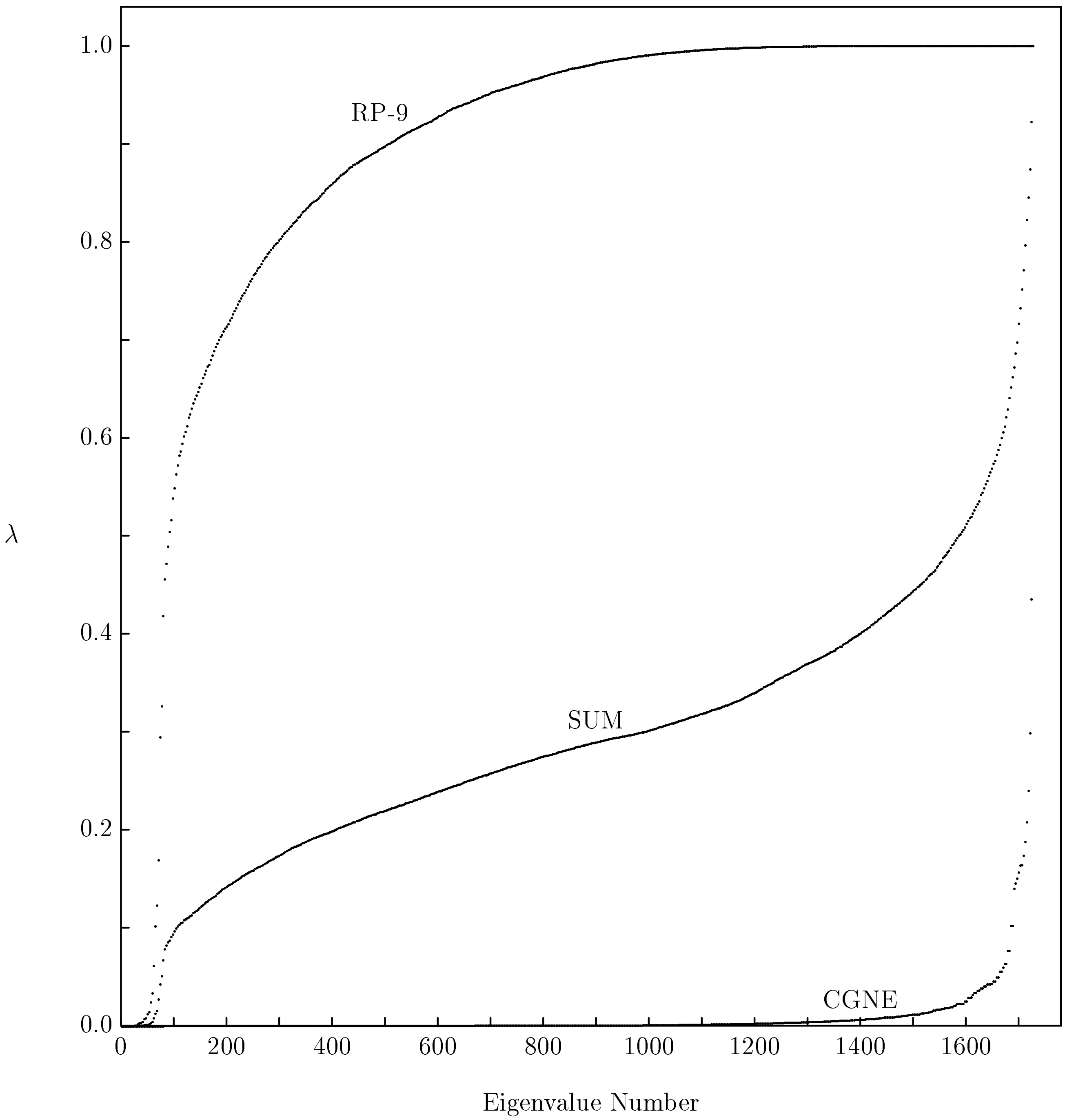


Figure 6.3: Comparison of Spectra for RP-9, SUM, and CGNE

relationship holds for all 9 of the test problems. Note that this result is for the entire spectral distribution and not just for the condition numbers of the system matrices.

A second implication of this argument is that RP methods will require fewer iterations for matrices A where the near linear dependence comes primarily from within a block row rather than between block rows. This is the reason for the difference in performance of the RP algorithms on Problems 1 and 2, even though the spectra of the matrices A are identical. A typical row i for both problems has, in the notation of Chapter 4, $x_0 = 6$, $z_+ = z_- = -1$. Problem 1 then has

$$x_- = 500h - 1, \quad x_+ = -(500h + 1), \quad y_- = y_+ = -1 \tag{6.5}$$

while Problem 2 reverses the role of x and y and so has

$$y_- = 500h - 1, \quad y_+ = -(500h + 1), \quad x_- = x_+ = -1 \tag{6.6}$$

The importance of this distinction becomes clear when the angles between row i and the other rows are computed, since the angles measure how close to linear dependence the rows are. Let a_i be the i^{th} row of A. For a mesh size with $h = 1/25$, Table 6.10 shows the expression for $\| a_i \|^2 * \cos\theta_j$, where θ_j is the angle between row i and row j, $j > i$. Only rows j not orthogonal to a_i are shown. The only value of $\cos\theta_j$ not close to 0 is 0.47. For Problem 1 this relatively small angle occurs *within* the same block row as a_i, while for Problem 2 it occurs for rows in other block rows. This explains the difference in performance and also indicates how to arrange the grid in a given problem, as well as for which problems RP methods are particularly suited. For CFD problems the grid should be arranged so that the predominant flow is in the x-direction, which is the case for Problem 1 but not for Problem 2. RP methods like all other iterative solvers work well if the system is diagonally dominant; furthermore, if there are large off-diagonal elements in the diagonal closest to the main diagonal, RP methods still work well. Although many workers in the field of RP methods have recognized the importance of having the angles beteen the block rows large, this seems to be the first guideline on how to achieve this for practical problems without having to actually compute the angles involved.

A third implication of the heuristic argument for the relative performance of RP methods is that the number of block rows should be kept as small as

Block Location	Row #	$\| a_i \|^2 * \cos\theta_j$	$\cos\theta_j$ Prob 1	Prob 2
Within	$i+1$	$x_0(x_- + x_+)$	0.014	0.014
Within	$i+2$	$x_- x_+$	0.470*	0.001
Between	$i+n-1$	$y_- x_- + y_+ x_+$	0.002	0.002
Between	$i+n$	$x_0(y_- + y_+)$	0.014	0.014
Between	$i+n+1$	$y_- x_+ + y_+ x_-$	0.002	0.002
Between	$i+2n$	$y_- y_+$	0.001	0.470*
Between	$i+n^2-n$	$y_- z_- + y_+ z_+$	0.002	0.002
Between	$i+n^2-1$	$x_- z_- + x_+ z_+$	0.002	0.002
Between	$i+n^2$	$x_0(z_- + z_+)$	0.010	0.014
Between	$i+n^2+1$	$x_+ z_- + x_- z_+$	0.002	0.002
Between	$i+n^2+n$	$y_+ z_- + y_- z_+$	0.002	0.002
Between	$i+2n^2$	$z_- z_+$	0.001	0.001

Table 6.10: Comparison of Angles to Row i for Problems 1 and 2

possible. The reason is that the partial orthogonalization between blocks in the W of Table 6.9 becomes less effective as more block rows appear. In terms of the heuristic, progressively more orthogonalizing factors $I - P_i$ must be stripped away before the orthogonalization effect between block row i and the succeeding block rows is felt. Keeping the number of block rows small can also be seen to be important because, e.g., for the $m = N$ case all of the ablility to form a near linear dependency between rows of the system matrix occurs between the block rows, where the outer CG acceleration method must deal with it. Finally, the error analysis also indicates that the number of block rows should be small since a factor of m occurs in the error bounds.

6.7 Comparison of RP-9 and V-RP

None of the arguments in the last section explain the difference in performance between RP-9 and V-RP, since both have system matrices with identical spectra. More importantly, why does V-RP apparently fail on two problems by going to the maximum number of iterations? The reason can be found in Theorem 5.4. V-RP minimizes the true error $x_k - x^*$ and not the residual $b - Ax_k$. However, the CG iteration must use the residual norm as a stopping test since clearly the error vector is not available. The distinction is especially acute for ill conditioned problems for which the error and residual can differ drastically. Figure 6.4 plots the error and residual norms for RP-9 and V-RP against the iteration number for Problem 5 when $N = 13824$.

Although after the first few iterations the V-RP error norm remains consistently smaller than the RP-9 error norm, the V-RP residual norm remains larger than the RP-9 residual norm. Hence RP-9 successfully terminates after 666 iterations because the residual is small, while V-RP continues iterating even though it has given a better solution after 666 iterations. V-RP apparently fails on some problems not because of a weakness in the method, but paradoxically because of its strength. The figures in Appendix B show similar curves to those in Figure 6.4 for other problems with $N = 13824$. Similar curves occur for the two failures of V-RP, and in both cases V-RP actually gives a better solution than RP-9 for the same number of iterations. V-RP generally runs for a few more iterations than RP-9 because of the difference between the residual and error norms, but gives smaller errors and requires less time per iteration. Less time is required because V-RP explicitly removes the $N/9$ eigenvalues of the system matrix that are equal to 1, and the vector operations needed by CG are for vectors of correspondingly shorter length. These efficiencies are offset only slightly by the work required to retrieve the solution x from the auxillary vector z at the end of the iterations.

In summary V-RP is a superior method for well conditioned or moderately ill conditioned systems. It is still superior for extremely ill conditioned systems in that it yields smaller error norms than RP-9, but unfortunately using the same stopping test for both will terminate RP-9 before V-RP. One way to avoid this problem is to recognize the error-minimizing property of V-RP and allow a less restrictive tolerance on the residual in the stopping test.

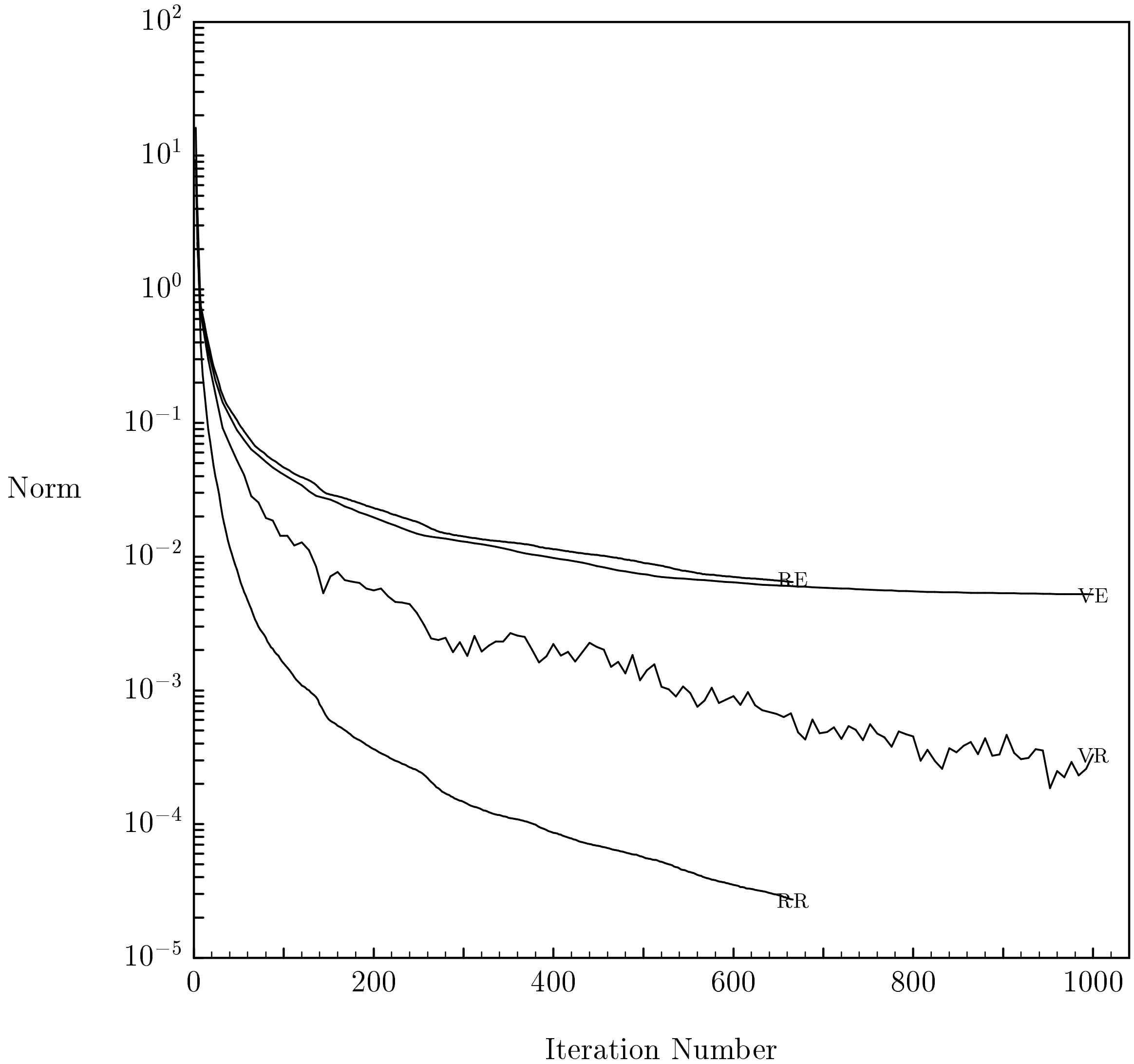


Figure 6.4: Error and Residual Norms for Problem 5 with VE = V-RP Error Norm, VR = V-RP Residual Norm, RE = RP-9 Error Norm, RR = RP-9 Residual Norm.

6.8 BST Acceleration

RP-9 was also tested with iterated block Stiefel acceleration, using one residual vector ($s = 1$), on the Alliant FX/8. As shown in Chapter 5 this can be implemented with only one parameter to be estimated, the initial estimate ν that defines the lower bound of the interval on which the first Chebyshev(x, ν, K) process is performed. In the tests this is arbitrarily set to be $\nu = 0.2$; after the first Chebyshev(x, ν, K) process the adaptive procedures are used. Additionally, the calculated ν is constrained to lie in the interval $[10^{-5}, 0.8]$ to guard against the adaptive procedure generating a ν outside of the interval $(0, 1)$. The results are tabulated in Tables 6.11 - 6.12. In every case BST acceleration requires more iterations and more time to solve the problem, and on Problem 8 with $N = 13824$ BST acceleration fails.

For the extremely ill conditioned Problems 5, 6, and 8 the poor performance of BST is explained in Section 5.2; when the Chebyshev(x, ν, K) process isolates an eigenvector of $I - Q_\pi$ corresponding to a small eigenvalue, the following projection step removes it but simultaneously causes growth in the other eigencomponents of the residual. Figure 6.5 shows the residual norm for each iteration for Problem 5 with $n = 13824$. The curve levels off as the eigencomponents in $[\nu, 1]$ become small. Then the projection step does not cause a large decrease in the residual because of the concommittent increase in the eigencomponents corresponding to the large eigenvalues. The estimate ν is adjusted and then the process starts over.

For well conditioned problems the method behaves like Chebyshev acceleration. For example, on Problem 1 the initial estimate $\nu = 0.2$ is less than the smallest eigenvalue of $I - Q_\pi$, so the projection step is never made and ν is not adjusted. Since CG acceleration implicitly finds the optimal interval containing the spectrum of $I - Q_\pi$, it requires fewer iterations in this case.

Finally, one of the best advantages of Chebyshev or BST over CG acceleration is the avoidance of inner products within the iteration, since inner products do not vectorize well. This advantage is obviated for RP methods because usually about 85% of the computation time occurs in the computation of the projections. Hence at most a 15% reduction in execution time can be achieved. When computed for the results in Table 6.12, the time per iteration for BST is 2-15% less than that for CG. This savings is negligible in view of the increased number of iterations.

Prob	Accel	Iter	Time	Residual	Error
1	CG	3	1.802E-01	2.060E-05	1.717E-06
	BST	15	4.181E-01	1.709E-05	6.628E-07
2	CG	12	2.818E-01	2.299E-05	1.506E-06
	BST	41	8.077E-01	1.908E-05	1.857E-06
3	CG	107	1.700E+00	1.921E-05	1.624E-06
	BST	184	3.090E+00	2.480E-05	2.895E-06
4	CG	8	2.218E-01	6.711E-06	6.082E-07
	BST	18	4.069E-01	3.142E-05	1.573E-06
5	CG	263	4.036E+00	2.899E-05	1.253E-03
	BST	1257	2.002E+01	2.375E-05	1.332E-03
6	CG	318	4.855E+00	1.755E-05	9.167E-08
	BST	835	1.333E+01	2.221E-05	2.872E-07
7	CG	19	3.898E-01	1.438E-05	1.521E-06
	BST	25	5.137E-01	3.007E-05	6.437E-06
8	CG	181	2.812E+00	1.325E-05	4.388E-04
	BST	775	1.234E+01	2.717E-05	1.022E-03
9	CG	50	8.470E-01	2.724E-05	3.558E-06
	BST	109	1.875E+00	2.634E-05	4.164E-06

Table 6.11: Comparison of BST and CG Acceleration for $N = 1728$

Prob	Accel	Iter	Time	Residual	Error
1	CG	8	2.012E+00	3.049E-05	6.189E-06
	BST	14	2.962E+00	1.399E-05	1.837E-06
2	CG	35	5.683E+00	1.191E-05	3.227E-06
	BST	90	1.364E+01	2.226E-05	8.889E-06
3	CG	109	1.595E+01	2.373E-05	3.422E-06
	BST	183	2.677E+01	2.432E-05	5.385E-06
4	CG	21	3.728E+00	1.603E-05	5.241E-06
	BST	34	5.969E+00	1.753E-05	1.543E-05
5	CG	666	9.337E+01	2.718E-05	6.476E-03
	BST	2375	3.280E+02	2.466E-05	6.562E-03
6	CG	715	1.001E+02	2.775E-05	1.170E-06
	BST	2063	2.852E+02	2.243E-05	1.387E-06
7	CG	20	3.584E+00	1.110E-05	1.964E-06
	BST	25	4.510E+00	1.379E-05	5.593E-06
8	CG	2656	3.698E+02	1.539E-05	1.241E-02
	BST	4001	5.500E+02	1.303E-02	2.543E+01
9	CG	33	5.401E+00	1.791E-05	1.739E-06
	BST	58	9.264E+00	2.087E-05	2.212E-06

Table 6.12: Comparison of BST and CG Acceleration for $N = 13824$

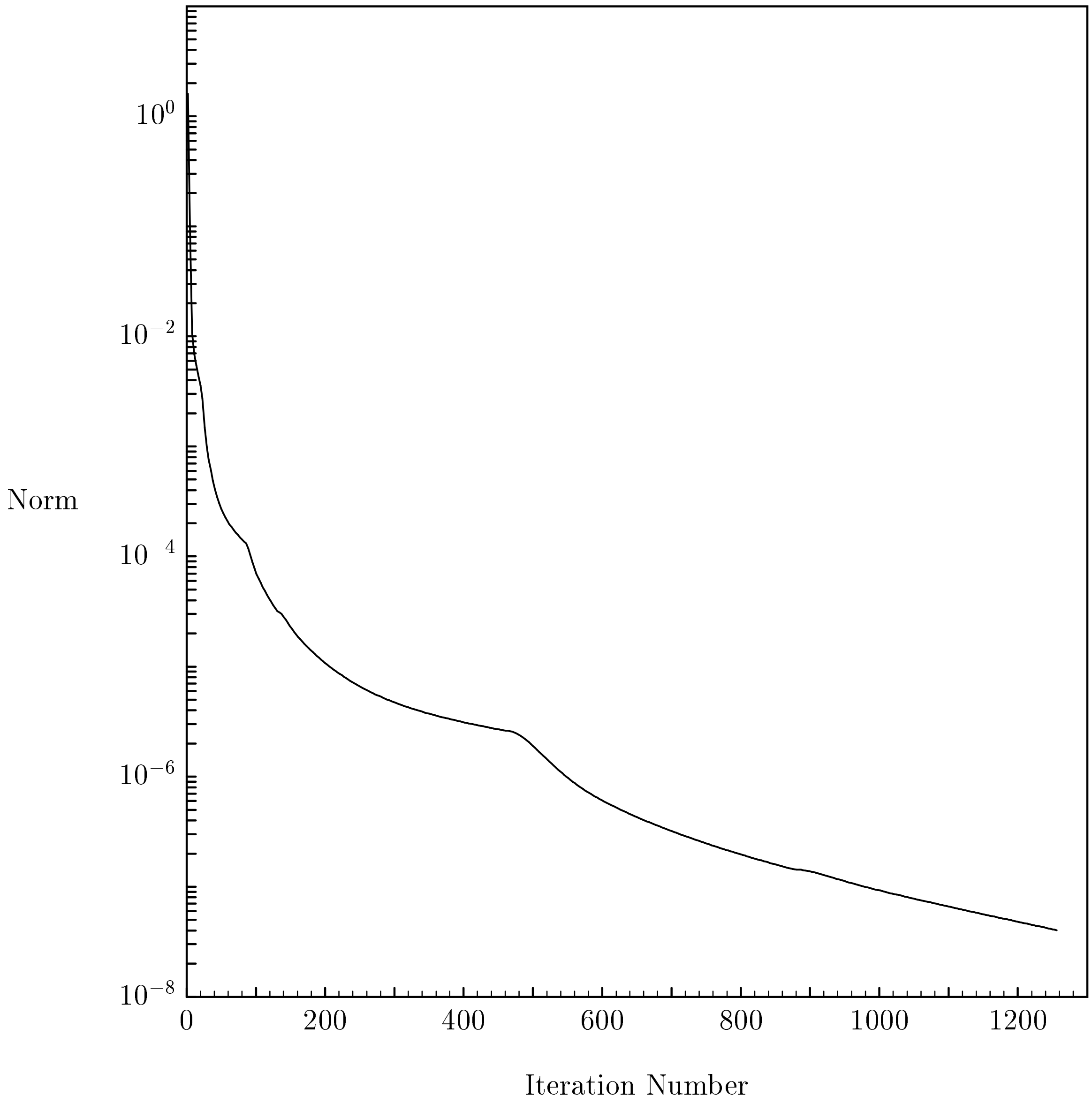


Figure 6.5: Residual Norm vs. Iteration Number for Problem 5 with $N = 13824$

The conclusion is that BST acceleration is not suitable for RP methods. However, for sparse positive definite matrices A where the entries of A are explictly stored and the system is ill-conditioned with only a few small eigenvalues, each iteration is much cheaper than with RP methods. In such cases BST acceleration may be more effective.

Chapter 7

Summary and Conclusions

After categorizing row projection methods into the two classes of product and sum forms, this thesis summarizes previous convergence results and presents two theorems that extend and generalize those results. The convergence does not rely on special spectral properties of the matrix A, and even holds when the system is inconsistent or A is rectangular. A new product RP form called V-RP is introduced which has both theoretical and computational advantages over the symmetrized product form. In particular, when CG acceleration is used the two norm of the error is minimized on each iteration, an explicit reduction in problem size can be made, and an amount of work roughly equivalent to one projection is eliminated from each iteration. The testing shows that these properties do lead to significant improvement over the symmetrized product form.

An analysis is made of block Stiefel (BST) acceleration and an adaptive iterated version proposed. The analysis concentrates on the BST projection step for eliminating the smallest eigenvector components from the problem, and shows that this occurs at the cost of increasing the other eigenvector components of the residual. Although numerical testing shows BST acceleration is inferior to CG acceleration for RP problems on shared memory machines, BST may be suitable for other linear problems.

The properties of product RP methods and CG acceleration when combined with a special starting estimate of x are shown to allow the elimination of one projection per iteration. The iterates x_k in this case will always satisfy the first block of equations, allowing, for example, constraint equations to remain satisfied.

A particular row partitioning scheme is proposed for elliptic partial differential equations in three dimensions. The resulting implementation allows direct solution methods for computing the necessary projections, uses only $\mathcal{O}(N)$ storage, provides large scale parallelism suitable both for multiprocessing and vectorization, and can be generalized to other problems. An error analysis shows that RP methods are stable, and that good bounds on the condition numbers of the subproblems created by the implementation can be easily obtained theoretically and numerically.

Testing shows that RP methods are numerically robust and can solve systems on which every other method tested fails. The particular implementation used here is also fast and efficient in comparison, because it allows $\mathcal{O}(n^2)$ parallelism for speed and works on subproblems of size $\mathcal{O}(n)$, providing data locality.

The underlying connection between the three RP methods tested and CGNE is revealed. All four consist of the CG applied to a system of the form W^TW for some W. From this the overall spectral distribution is expected to be best for the product RP methods, followed by the sum form RP method, with CGNE worst. Numerically the difference can be striking, and leads to CGNE failing on problems for which RP methods succeed. This analysis also shows why RP methods work better on some problems than others, even when the problems matrices A have identical spectra. This in turn identifies problems for which RP methods are especially suitable and indicates how to order the nodes for such problems.

In conclusion, RP methods are remarkably robust, well suited for parallel and vector processing, and competitive with if not better than other iterative solvers in speed and efficiency.

Appendix A

Tables for Comparison of Methods

This appendix summarizes the testing results of Chapter 6. The error codes are as defined in that chapter and consist of:

- MX: Maximum iterations reached
- RS: Residual stagnates
- UP: Unstable preconditioner
- TM: Maximum allowed CPU time reached

Comparison of Methods for $N = 1728$						
Prob	Meth	Iters	Time	Resid	Error	Code
1	RP-9	3	1.802e-01	2.060e-05	1.717e-06	
	V-RP	3	1.609e-01	2.101e-05	1.704e-06	
	SUM	5	1.844e-01	2.404e-05	5.717e-07	
	GMRES	112	1.934e+00	2.598e-05	2.616e-06	
	ILU	9	1.105e+00	1.886e-05	5.466e-07	
	MILU	491	3.783e+01	1.577e-05	8.082e-07	
	CGNE	32	1.888e-01	7.397e-07	6.770e-08	
	ICCG	15	1.406e+00	2.519e-05	8.371e-07	
2	RP-9	12	2.818e-01	2.299e-05	1.506e-06	
	V-RP	13	2.813e-01	2.410e-05	7.510e-07	
	SUM	22	3.604e-01	2.630e-05	2.285e-06	
	GMRES	112	1.831e+00	2.598e-05	2.616e-06	
	ILU	9	9.861e-01	1.886e-05	5.466e-07	
	MILU	537	4.113e+01	1.568e-05	8.046e-07	
	CGNE	32	1.813e-01	7.397e-07	6.770e-08	
	ICCG	15	1.401e+00	2.026e-05	7.236e-07	
3	RP-9	107	1.700e+00	1.921e-05	1.624e-06	
	V-RP	111	2.044e+00	2.013e-05	6.187e-07	
	SUM	300	4.361e+00	1.803e-05	2.161e-06	
	GMRES	1015	1.660e+01	3.147e-05	2.745e-06	
	ILU	75	6.146e+00	9.294e+03	1.981e+02	RS
	MILU	1729	1.533e+02	2.346e+03	6.522e+01	MX
	CGNE	411	2.206e+00	9.695e-07	1.183e-07	
	ICCG	1729	1.392e+02	4.075e+16	4.714e+14	MX

Comparison of Methods for $N = 1728$						
Prob	Meth	Iters	Time	Resid	Error	Code
4	RP-9	8	2.218e-01	6.711e-06	6.082e-07	
	V-RP	8	2.019e-01	7.273e-06	6.022e-07	
	SUM	16	2.793e-01	8.467e-06	1.036e-06	
	GMRES	334	5.451e+00	3.054e-05	3.822e-06	
	ILU	9	9.886e-01	1.758e-05	1.354e-06	
	MILU	8	9.503e-01	1.869e-05	1.610e-06	
	CGNE	202	1.089e+00	1.489e-06	2.434e-07	
	ICCG	16	1.487e+00	8.518e-06	4.100e-07	
5	RP-9	263	4.036e+00	2.899e-05	1.253e-03	
	V-RP	331	5.998e+00	2.435e-05	2.567e-04	
	SUM	156	2.293e+00	2.887e-05	9.783e-04	
	GMRES	142	2.322e+00	3.999e+00	2.022e+00	RS
	ILU	186	1.457e+01	4.194e+02	3.222e+02	RS
	MILU	76	6.056e+00	2.174e+02	2.215e+01	RS
	CGNE	1729	1.733e+01	3.128e-04	3.273e-03	MX
	ICCG	1729	1.393e+02	2.247e+02	1.892e+01	MX
6	RP-9	318	4.855e+00	1.755e-05	9.167e-08	
	V-RP	330	5.978e+00	2.050e-05	4.023e-08	
	SUM	790	1.142e+01	2.341e-05	4.027e-07	
	GMRES	1244	2.044e+01	2.311e+02	4.837e+00	RS
	ILU	53	4.425e+00	5.469e+04	6.853e+01	RS
	MILU	183	1.417e+01	1.171e+00	1.319e-02	RS
	CGNE	1729	1.721e+01	1.889e+00	1.708e-01	MX
	ICCG					UP

Comparison of Methods for $N = 1728$						
Prob	Meth	Iters	Time	Resid	Error	Code
7	RP-9	19	3.898e-01	1.438e-05	1.521e-06	
	V-RP	19	3.885e-01	2.009e-05	1.394e-06	
	SUM	47	7.200e-01	2.858e-05	4.406e-06	
	GMRES	438	7.157e+00	3.077e-05	2.567e-06	
	ILU	51	4.014e+00	1.946e-05	1.155e-06	
	MILU	43	3.548e+00	5.328e+02	2.026e+01	RS
	CGNE	258	1.386e+00	1.591e-06	2.512e-07	
	ICCG	1729	1.393e+02	3.777e-04	2.913e-05	MX
8	RP-9	181	2.812e+00	1.325e-05	4.388e-04	
	V-RP	316	5.772e+00	2.121e-05	4.796e-05	
	SUM	286	4.163e+00	1.033e-05	4.962e-04	
	GMRES	362	5.913e+00	2.291e-02	1.301e-02	RS
	ILU	62	5.066e+00	9.222e+01	2.009e+01	RS
	MILU	164	1.415e+01	2.298e+02	1.894e+01	RS
	CGNE	1556	8.315e+00	4.170e-06	4.589e-04	
	ICCG					UP
9	RP-9	50	8.470e-01	2.724e-05	3.558e-06	
	V-RP	53	1.001e+00	2.319e-05	1.454e-06	
	SUM	140	2.058e+00	1.972e-05	3.921e-06	
	GMRES	427	6.973e+00	2.809e-05	2.393e-06	
	ILU	225	1.774e+01	3.476e+02	1.805e+01	RS
	MILU	61	5.117e+00	6.435e+02	2.090e+01	RS
	CGNE	222	1.195e+00	2.511e-06	5.564e-07	
	ICCG	1729	1.394e+02	1.125e+07	2.253e+05	MX

Comparison of Methods for $N = 13824$						
Prob	Meth	Iters	Time	Resid	Error	Code
1	RP-9	8	2.012e+00	3.049e-05	6.189e-06	
	V-RP	9	1.760e+00	9.522e-06	1.644e-06	
	SUM	17	2.826e+00	2.210e-05	5.233e-06	
	GMRES	269	3.184e+01	3.157e-05	1.125e-05	
	ILU	10	9.370e+00	8.345e-06	4.317e-07	
	MILU	127	7.999e+01	2.256e+00	3.651e-01	RS
	CGNE	90	5.135e+00	2.617e-06	8.422e-07	
	ICCG	16	1.057e+01	1.627e-05	1.437e-06	
2	RP-9	35	5.683e+00	1.191e-05	3.227e-06	
	V-RP	36	5.280e+00	1.571e-05	1.863e-06	
	SUM	73	1.001e+01	1.267e-05	4.056e-06	
	GMRES	269	3.081e+01	3.157e-05	1.125e-05	
	ILU	10	8.741e+00	8.345e-06	4.317e-07	
	MILU	120	7.507e+01	2.501e+00	4.113e-01	RS
	CGNE	90	5.107e+00	2.617e-06	8.422e-07	
	ICCG	16	1.048e+01	1.662e-05	1.449e-06	
3	RP-9	109	1.595e+01	2.373e-05	3.422e-06	
	V-RP	113	1.583e+01	2.258e-05	1.357e-06	
	SUM	343	4.579e+01	2.504e-05	5.845e-06	
	GMRES	757	8.706e+01	3.080e-05	2.174e-06	
	ILU	4001	3.446e+03	3.304e+03	2.028e+02	MX
	MILU					UP
	CGNE	677	3.782e+01	1.571e-06	4.199e-07	
	ICCG					UP

Comparison of Methods for $N = 13824$						
Prob	Meth	Iters	Time	Resid	Error	Code
4	RP-9	21	3.728e+00	1.603e-05	5.241e-06	
	V-RP	21	3.227e+00	2.425e-05	4.745e-06	
	SUM	54	7.503e+00	1.468e-05	6.183e-06	
	GMRES	310	3.562e+01	3.104e-05	7.833e-06	
	ILU	12	1.002e+01	1.140e-05	1.790e-06	
	MILU	10	9.678e+00	1.936e-05	2.635e-06	
	CGNE	387	2.165e+01	3.581e-06	1.761e-06	
	ICCG	24	1.503e+01	3.062e-05	5.443e-06	
5	RP-9	666	9.337e+01	2.718e-05	6.476e-03	
	V-RP	2659	3.645e+02	2.387e-05	3.695e-03	
	SUM	571	7.603e+01	2.547e-05	5.225e-03	
	GMRES	703	8.058e+01	2.166e+01	1.379e+01	RS
	ILU	88	5.740e+01	1.977e+05	1.724e+05	RS
	MILU	86	5.475e+01	1.056e+06	5.421e+05	RS
	CGNE	8001	9.652e+02	3.942e-05	5.538e-03	MX
	ICCG					UP
6	RP-9	715	1.001e+02	2.775e-05	1.170e-06	
	V-RP	813	1.111e+02	2.230e-05	8.381e-08	
	SUM	1996	2.647e+02	2.428e-05	7.137e-07	
	GMRES	3708	4.290e+02	2.989e+02	8.676e+00	RS
	ILU					UP
	MILU	647	4.399e+02	1.556e-05	8.093e-07	
	CGNE	8001	9.800e+02	8.348e+00	1.048e+01	MX
	ICCG					UP

Comparison of Methods for $N = 13824$						
Prob	Meth	Iters	Time	Resid	Error	Code
7	RP-9	20	3.584e+00	1.110e-05	1.964e-06	
	V-RP	20	3.086e+00	1.586e-05	1.757e-06	
	SUM	51	7.102e+00	1.925e-05	5.147e-06	
	GMRES	548	6.336e+01	3.162e-05	9.187e-06	
	ILU	41	2.839e+01	1.866e-05	1.198e-06	
	MILU	32	2.234e+01	5.825e+03	3.013e+02	RS
	CGNE	325	1.857e+01	2.459e-06	8.221e-07	
	ICCG	594	3.389e+02	2.087e-05	3.820e-06	
8	RP-9	2656	3.698e+02	1.539e-05	1.241e-02	
	V-RP	3111	4.271e+02	2.473e-05	2.051e-03	
	SUM	4511	6.155e+02	1.493e-05	1.415e-02	
	GMRES	175	2.019e+01	6.105e+00	2.448e+00	RS
	ILU	98	6.198e+01	3.715e+00	8.406e+01	RS
	MILU	142	8.910e+01	1.839e+02	5.288e+01	RS
	CGNE	8001	9.817e+02	1.499e-03	2.023e+00	MX
	ICCG	4001	2.337e+03	8.349e-02	2.508e+01	MX
9	RP-9	33	5.401e+00	1.791e-05	1.739e-06	
	V-RP	33	4.884e+00	1.921e-05	1.492e-06	
	SUM	92	1.255e+01	1.902e-05	2.837e-06	
	GMRES	238	2.727e+01	3.030e-05	3.786e-06	
	ILU	186	1.266e+02	3.688e+02	5.031e+01	RS
	MILU	428	2.731e+02	3.689e+02	5.060e+01	RS
	CGNE	114	6.556e+00	1.484e-06	2.314e-07	
	ICCG	4001	2.328e+03	2.687e+05	9.099e+03	MX

Comparison of Methods for $N = 216000$						
Prob	Meth	Iters	Time	Resid	Error	Code
1	RP-9	29	8.789e+00	2.280e-05	2.753e-05	
	V-RP	32	9.236e+00	1.918e-05	9.251e-06	
	SUM	76	1.220e+01	2.283e-05	3.728e-05	
	GMRES	211	8.959e+01	2.930e-05	1.292e-05	
	ILU	1	2.727e+00	2.867e-05	5.865e-06	
	MILU	713	7.941e+02	2.404e+00	1.821e+00	RS
	CGNE	183	1.017e+01	1.168e-05	1.971e-05	
	ICCG	22	2.246e+01	2.772e-05	1.146e-05	
2	RP-9	72	2.087e+01	2.361e-05	3.307e-05	
	V-RP	78	2.160e+01	2.796e-05	1.214e-05	
	SUM	169	2.664e+01	1.024e-05	1.691e-05	
	GMRES	211	8.958e+01	2.930e-05	1.292e-05	
	ILU	1	2.727e+00	2.867e-05	5.865e-06	
	MILU					TM
	CGNE	183	1.007e+01	1.168e-05	1.971e-05	
	ICCG	22	2.244e+01	2.437e-05	1.128e-05	
3	RP-9	189	5.357e+01	2.057e-05	1.211e-05	
	V-RP	197	5.365e+01	2.166e-05	3.816e-06	
	SUM	665	1.045e+02	2.309e-05	2.543e-05	
	GMRES	669	2.877e+02	3.043e-05	2.392e-06	
	ILU					UP
	MILU					UP
	CGNE	1591	8.654e+01	4.853e-06	6.349e-06	
	ICCG					UP

Comparison of Methods for $N = 216000$						
Prob	Meth	Iters	Time	Resid	Error	Code
4	RP-9	100	2.866e+01	2.843e-05	5.841e-05	
	V-RP	107	2.953e+01	2.124e-05	1.155e-05	
	SUM	295	4.658e+01	2.049e-05	5.307e-05	
	GMRES	363	1.566e+02	3.026e-05	2.541e-05	
	ILU	2	3.785e+00	7.655e-06	3.608e-06	
	MILU	1	2.766e+00	2.113e-05	9.291e-06	
	CGNE	908	5.085e+01	1.351e-05	3.999e-05	
	ICCG	80	7.622e+01	2.509e-05	3.621e-05	
5	RP-9	1292	3.589e+02	2.729e-05	2.536e-02	
	V-RP	4001	1.076e+03	4.544e-04	2.180e-02	MX
	SUM	4477	7.057e+02	2.380e-05	2.548e-02	
	GMRES	902	3.881e+02	1.091e+02	1.020e+02	RS
	ILU					UP
	MILU	318	3.588e+02	1.091e+02	6.688e+06	RS
	CGNE	8001	4.406e+02	6.780e-04	4.283e-02	MX
	ICCG					TM
6	RP-9	644	1.792e+02	2.250e-05	1.907e-06	
	V-RP	758	2.041e+02	2.404e-05	2.711e-07	
	SUM	2090	3.226e+02	2.272e-05	2.691e-06	
	GMRES					TM
	ILU					UP
	MILU	256	2.824e+02	2.875e-05	1.228e-06	
	CGNE	8001	4.377e+02	1.544e+01	7.757e+01	MX
	ICCG					UP

Comparison of Methods for $N = 216000$						
Prob	Meth	Iters	Time	Resid	Error	Code
7	RP-9	48	1.397e+01	2.249e-05	2.460e-05	
	V-RP	51	1.420e+01	2.108e-05	7.645e-06	
	SUM	130	2.007e+01	2.471e-05	3.490e-05	
	GMRES	413	1.785e+02	3.106e-05	2.237e-05	
	ILU	8	1.023e+01	1.635e-05	3.037e-06	
	MILU	21	3.259e+01	2.372e+02	4.720e+03	RS
	CGNE	550	3.034e+01	8.548e-06	1.353e-05	
	ICCG	128	1.204e+02	1.976e-05	1.980e-05	
8	RP-9	4001	1.114e+03	1.196e-03	3.779e+00	MX
	V-RP	4001	1.071e+03	3.590e-01	2.893e+00	MX
	SUM	8001	1.260e+03	1.259e-03	4.374e+00	MX
	GMRES	138	6.022e+01	1.155e+02	1.315e+02	RS
	ILU					TM
	MILU	152	1.765e+02	1.155e+02	2.220e+03	RS
	CGNE	8001	4.359e+02	3.533e-03	7.761e+00	MX
	ICCG					TM
9	RP-9	48	1.395e+01	2.475e-05	7.683e-06	
	V-RP	49	1.378e+01	2.399e-05	4.510e-06	
	SUM	160	2.486e+01	2.381e-05	1.236e-05	
	GMRES	365	1.577e+02	2.983e-05	1.572e-05	
	ILU	33	4.009e+01	2.296e-05	1.148e-05	
	MILU					UP
	CGNE	316	1.740e+01	5.275e-06	3.852e-06	
	ICCG					TM

Appendix B

Graphs of V-RP and RP-9 Error and Residual Norms

This appendix summarizes the iteration histories for RP-9 and V-RP for Problems 3-4 and 6-9. The curves are labeled using the scheme

- RE: RP-9 Error Norm
- RR: RP-9 Residual Norm
- VE: V-RP Error Norm
- VR: V-RP Residual Norm.

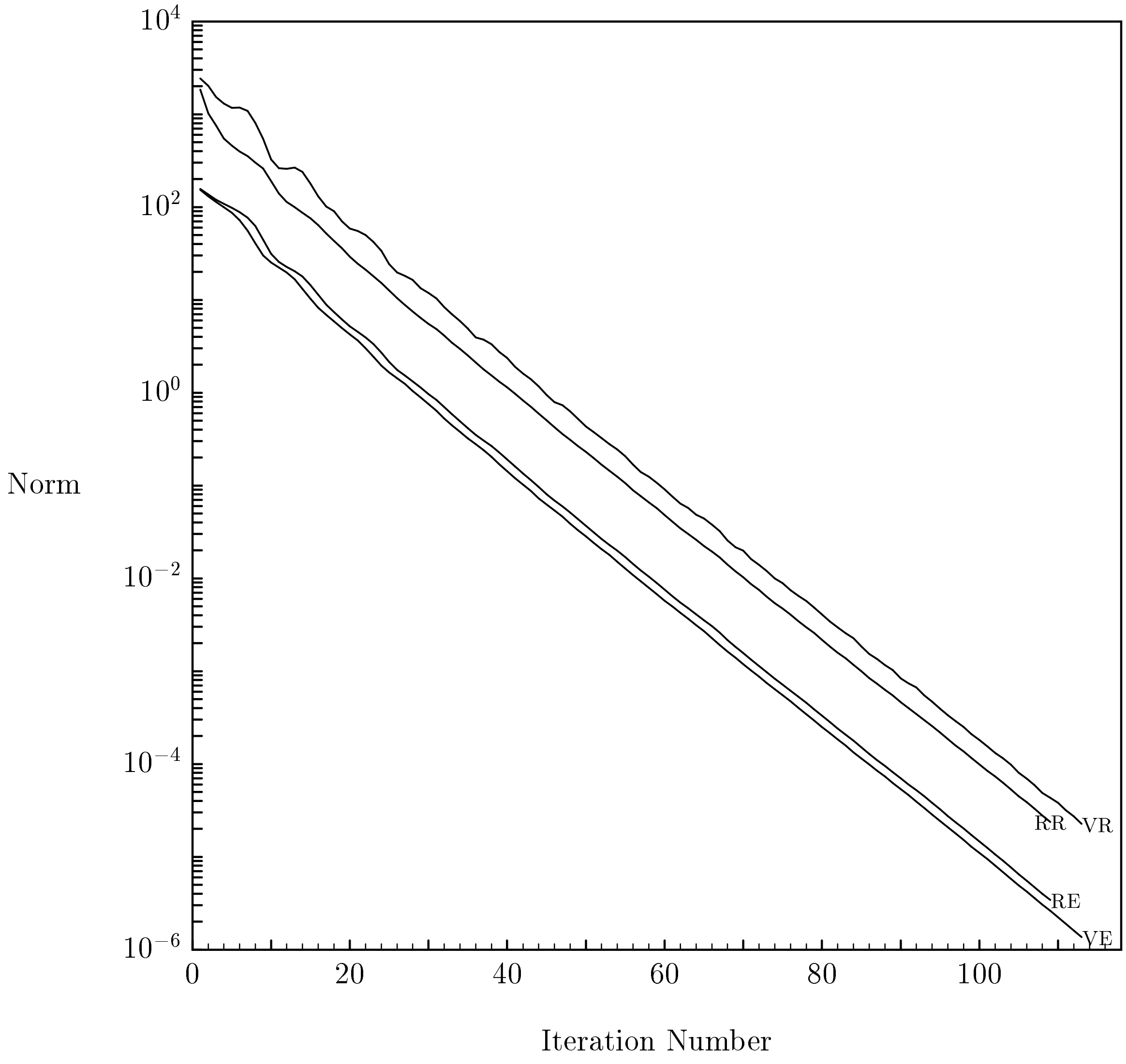


Figure B.1: Error and Residual Norms for Problem 3

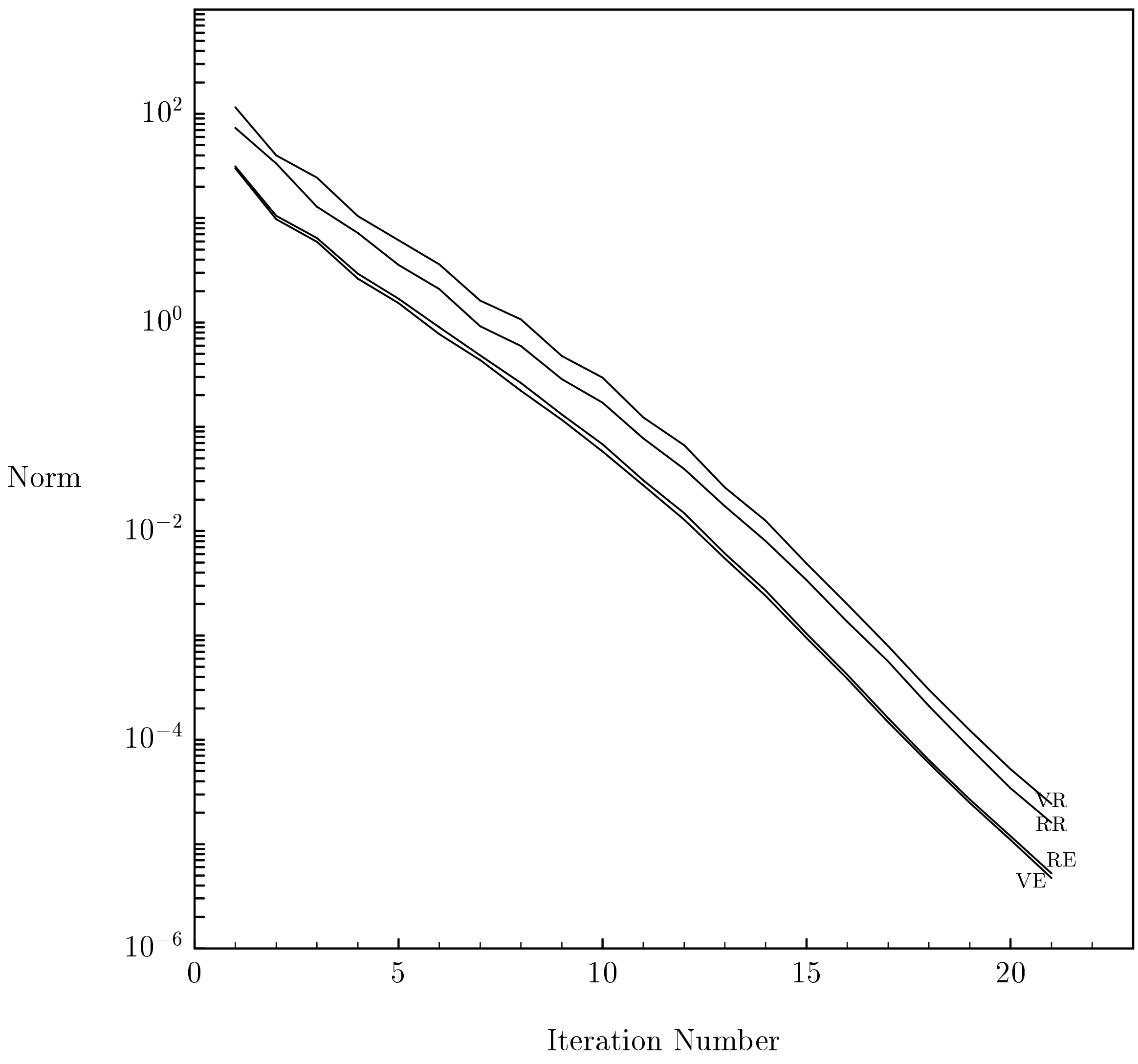


Figure B.2: Error and Residual Norms for Problem 4

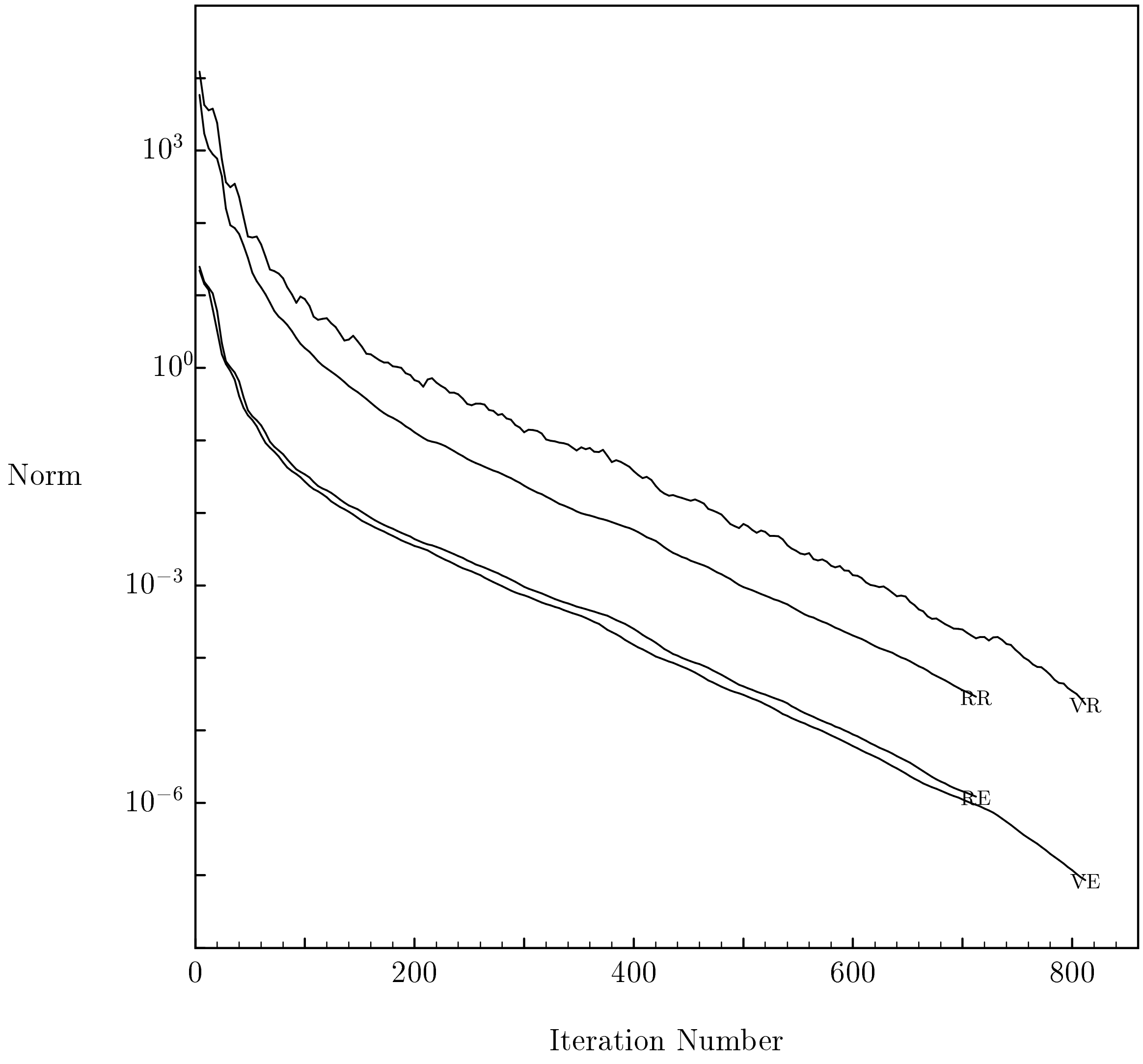


Figure B.3: Error and Residual Norms for Problem 6

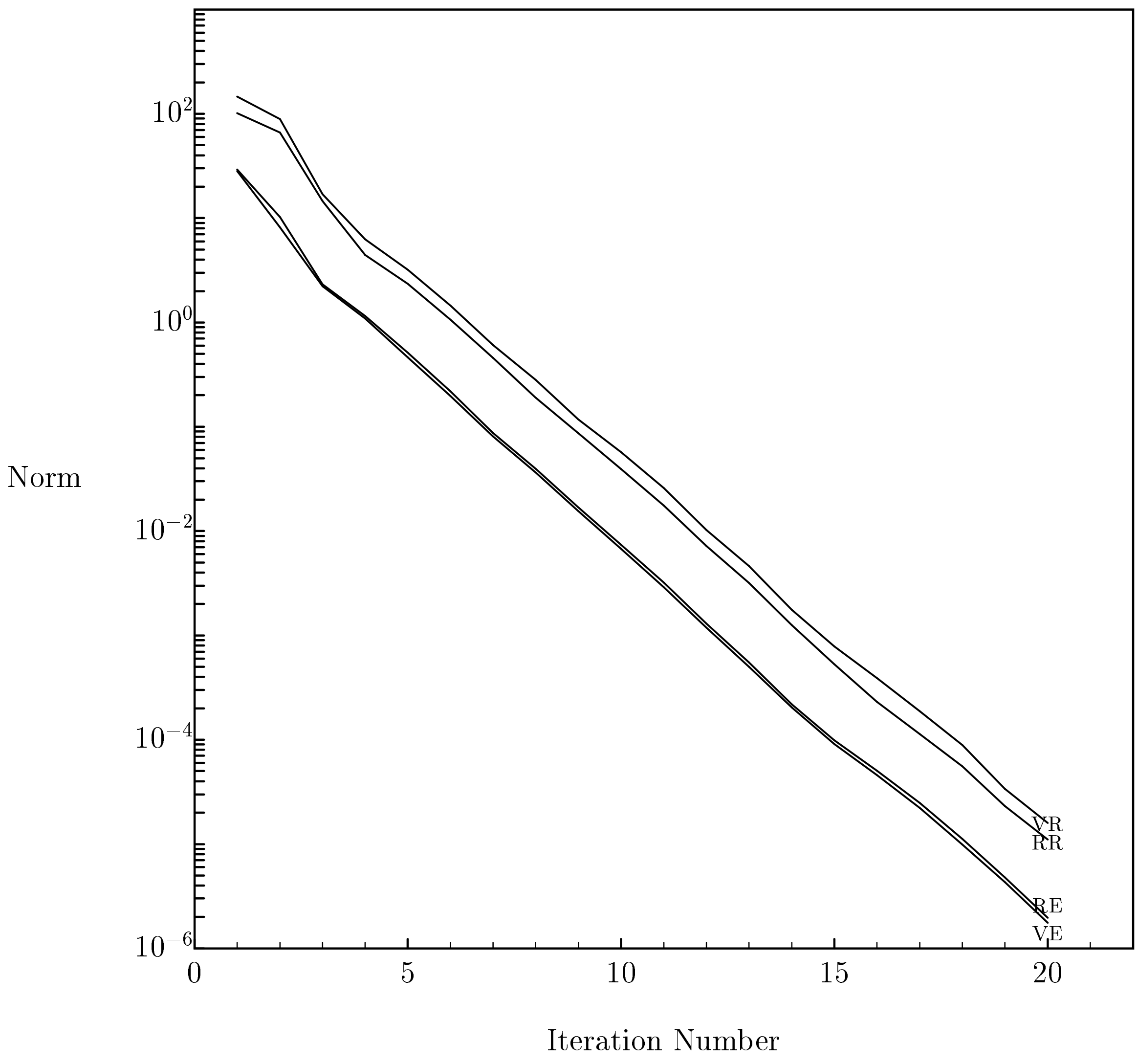


Figure B.4: Error and Residual Norms for Problem 7

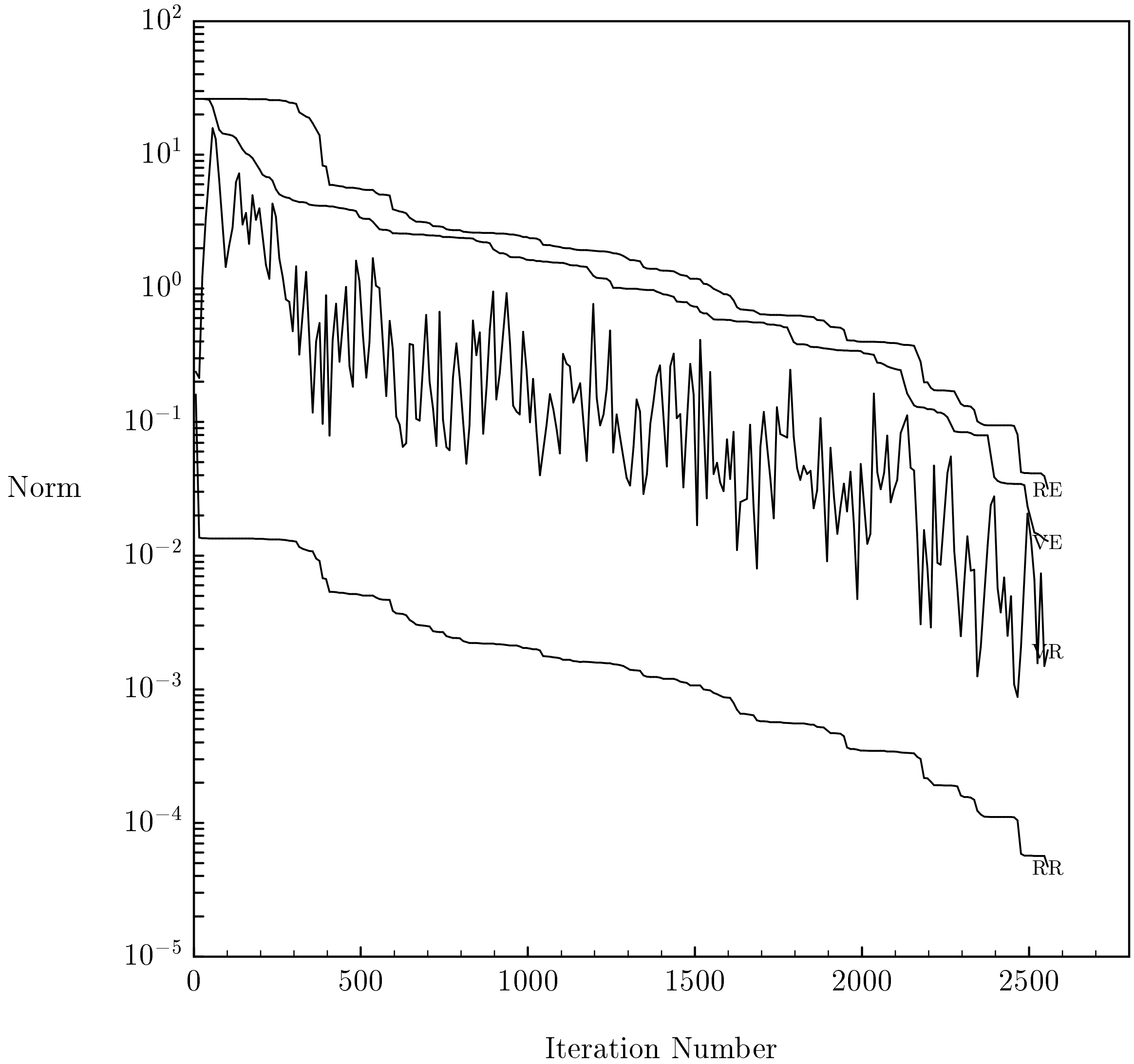


Figure B.5: Error and Residual Norms for Problem 8

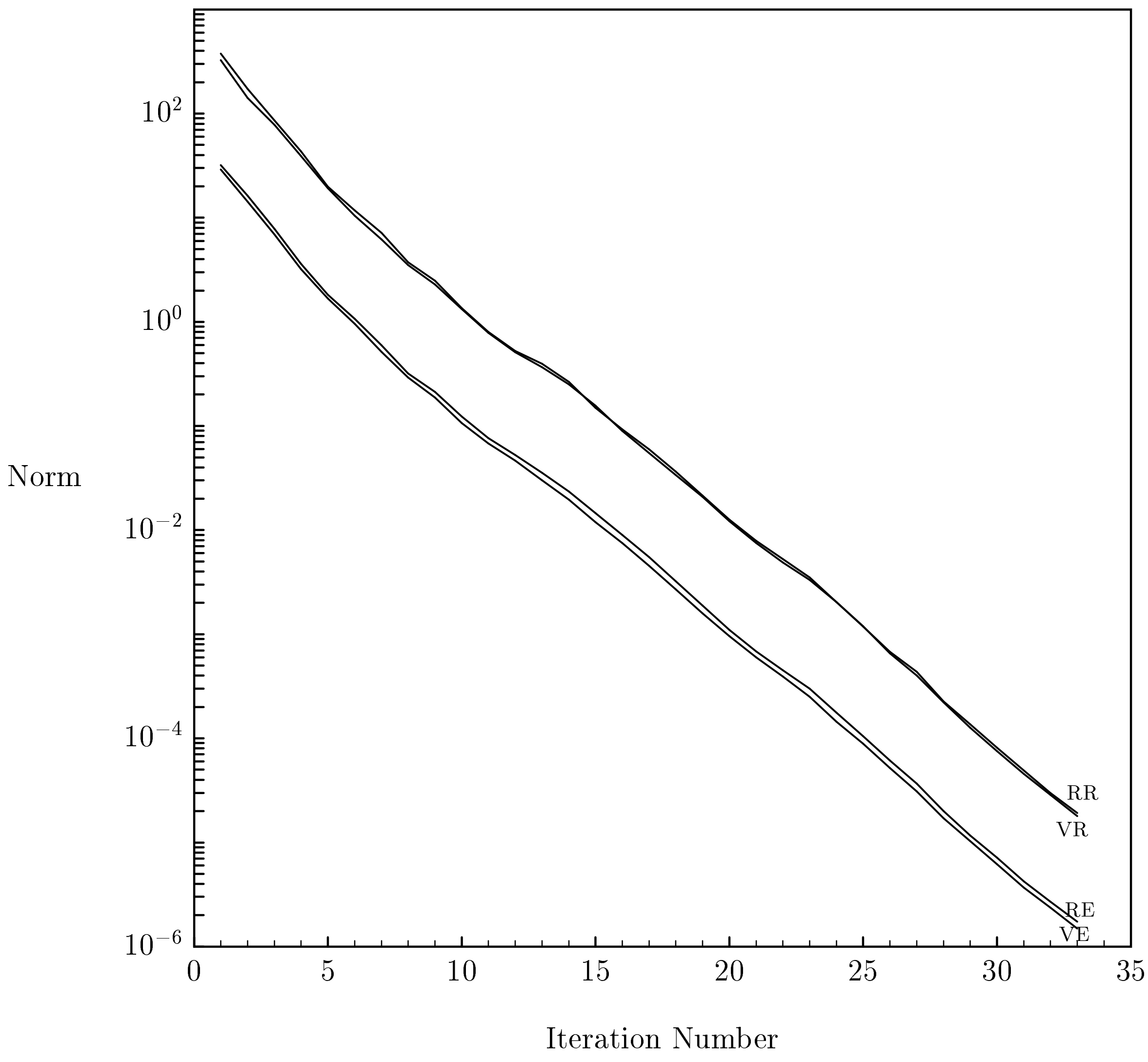


Figure B.6: Error and Residual Norms for Problem 9

Appendix C

Spectra of A for Test Problems with $N = 1728$

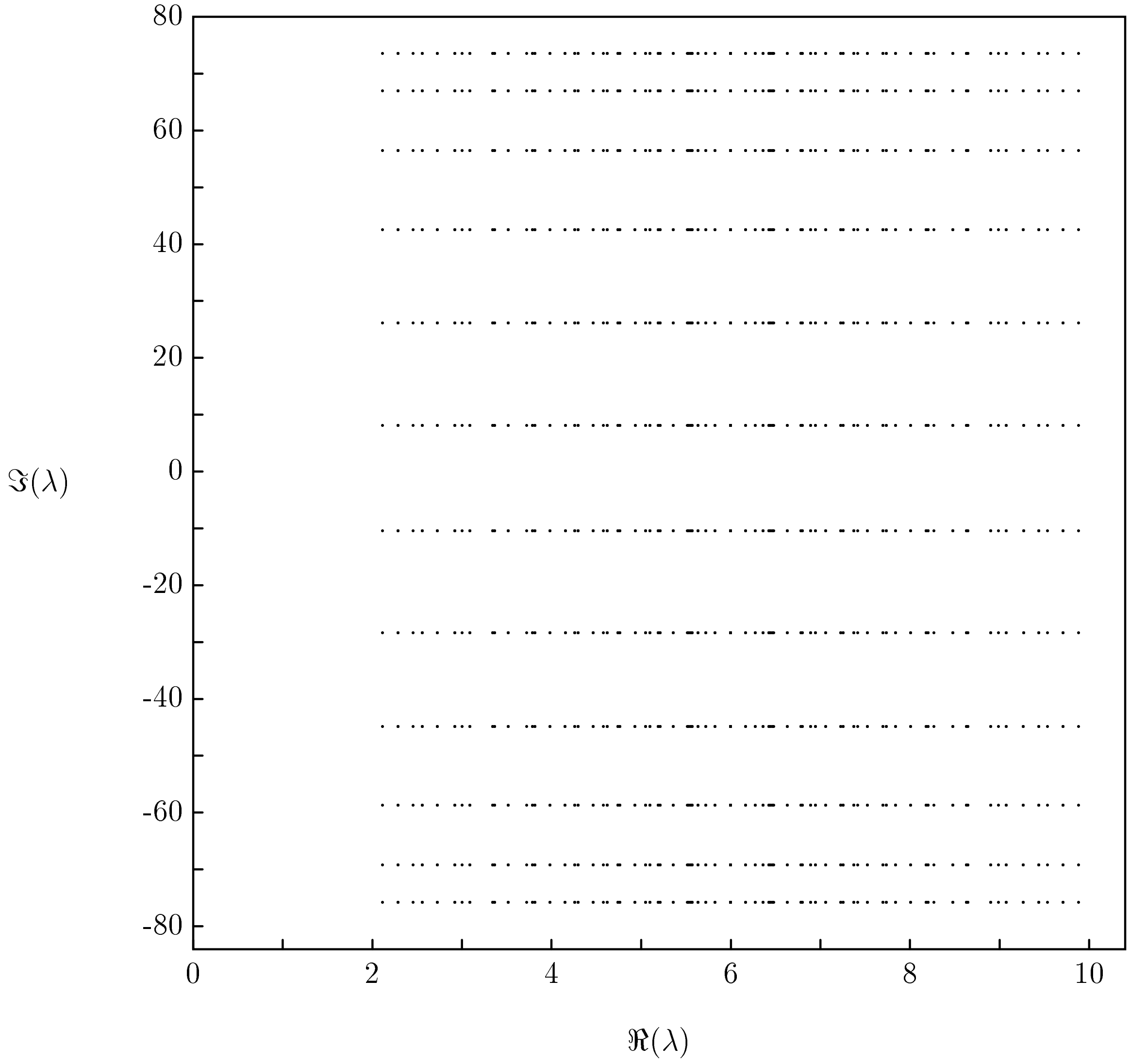


Figure C.1: Spectrum of A for Problem 1 with $N = 1728$

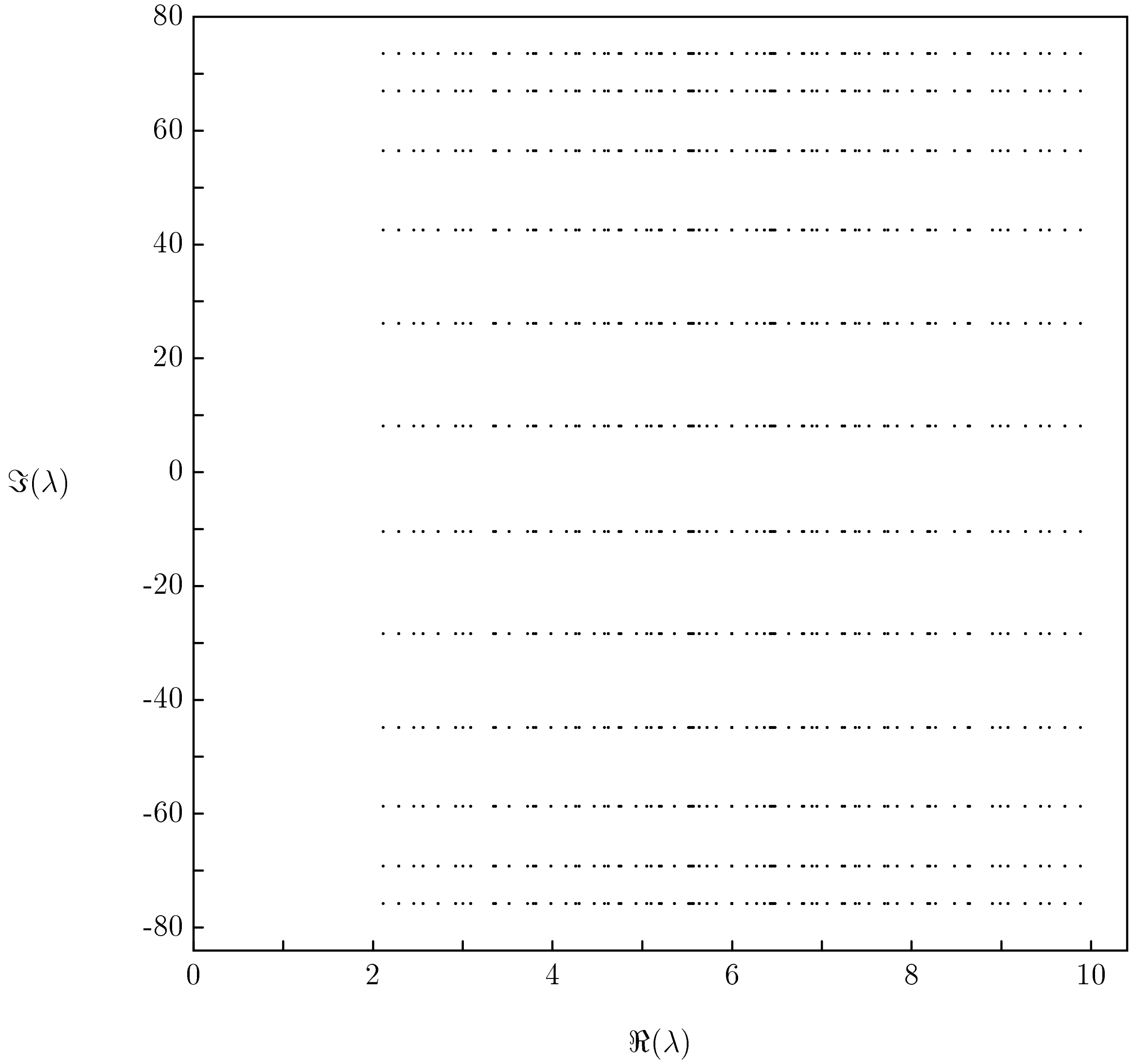


Figure C.2: Spectrum of A for Problem 2 with $N = 1728$

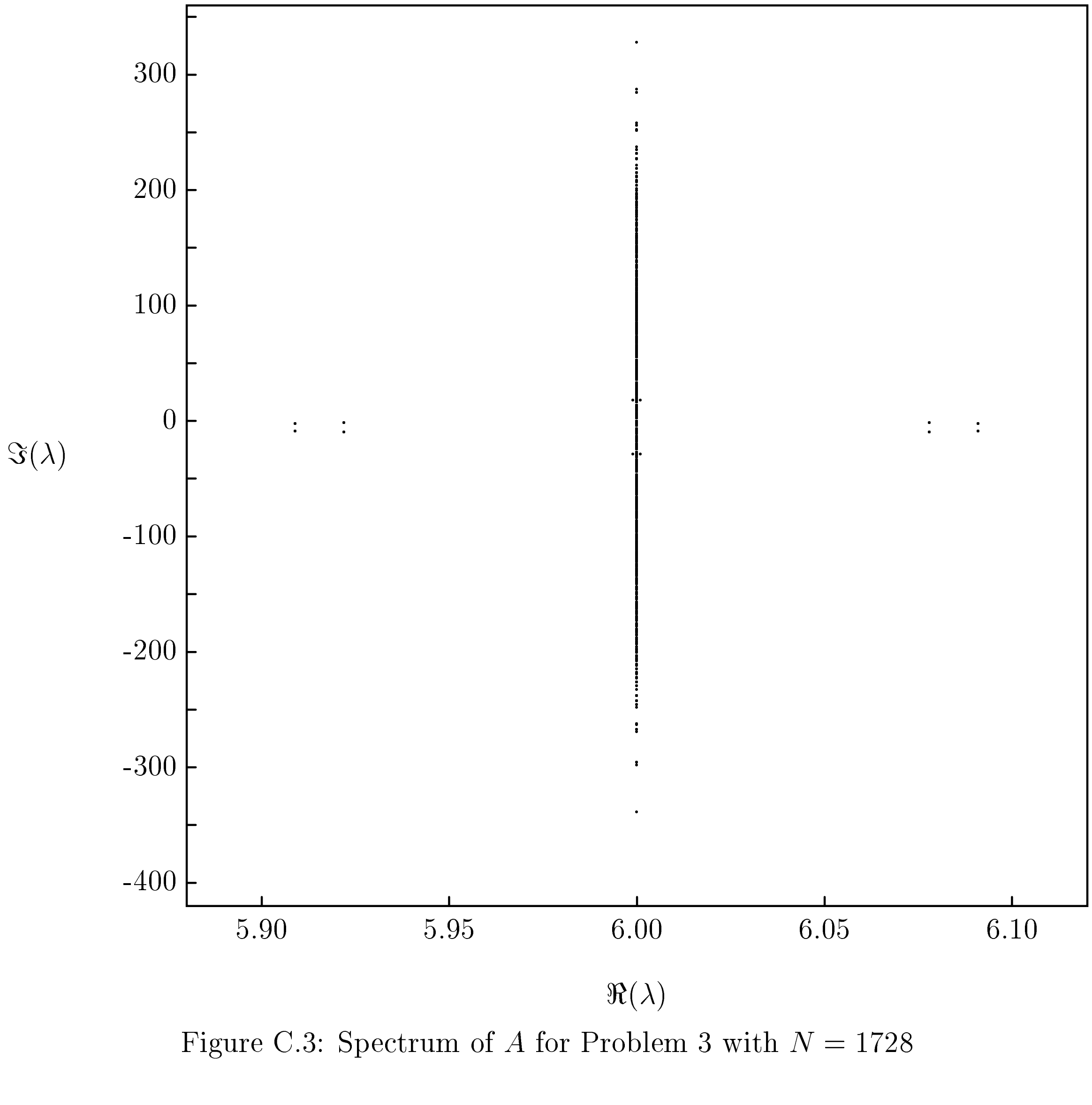


Figure C.3: Spectrum of A for Problem 3 with $N = 1728$

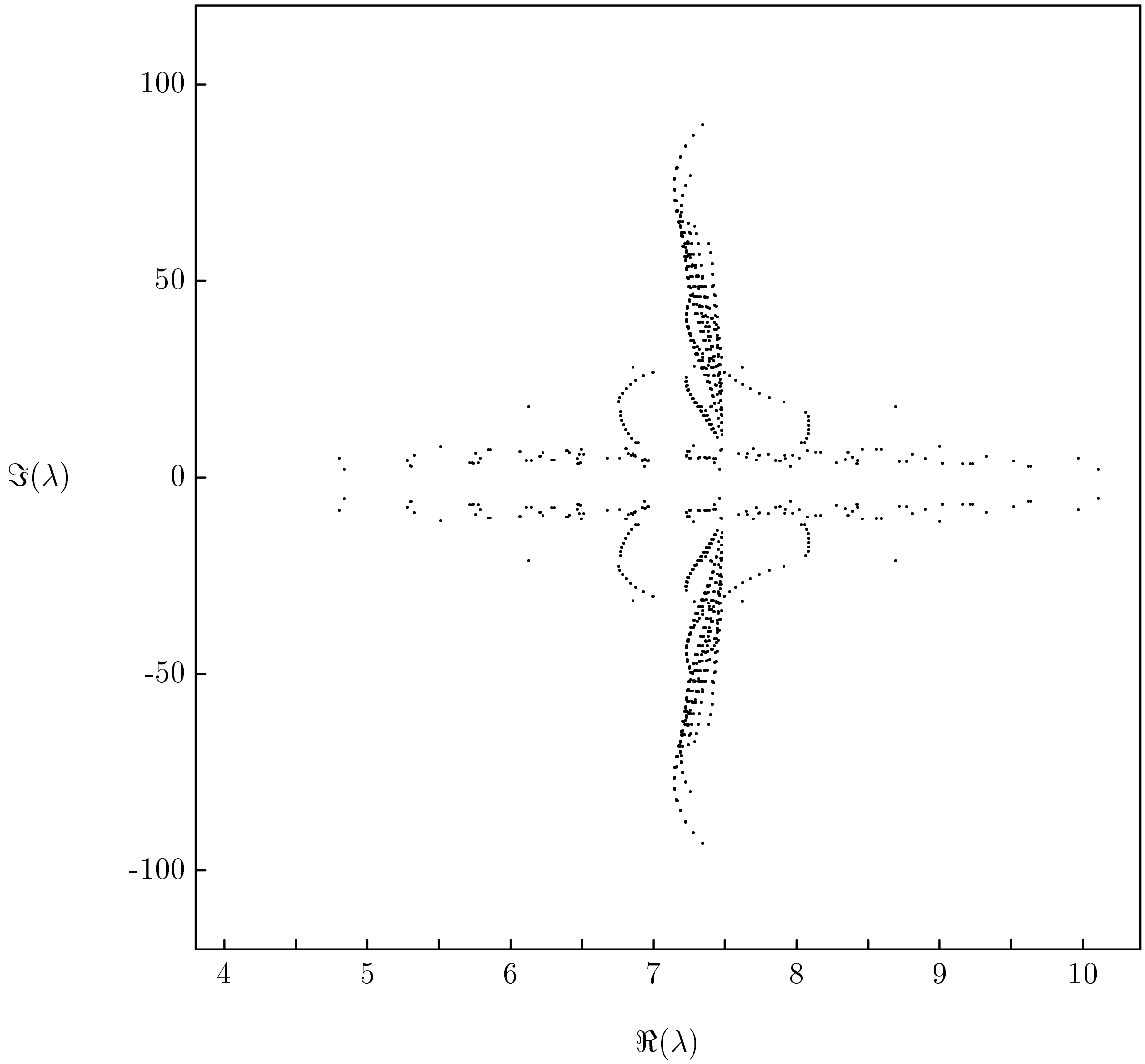


Figure C.4: Spectrum of A for Problem 4 with $N = 1728$

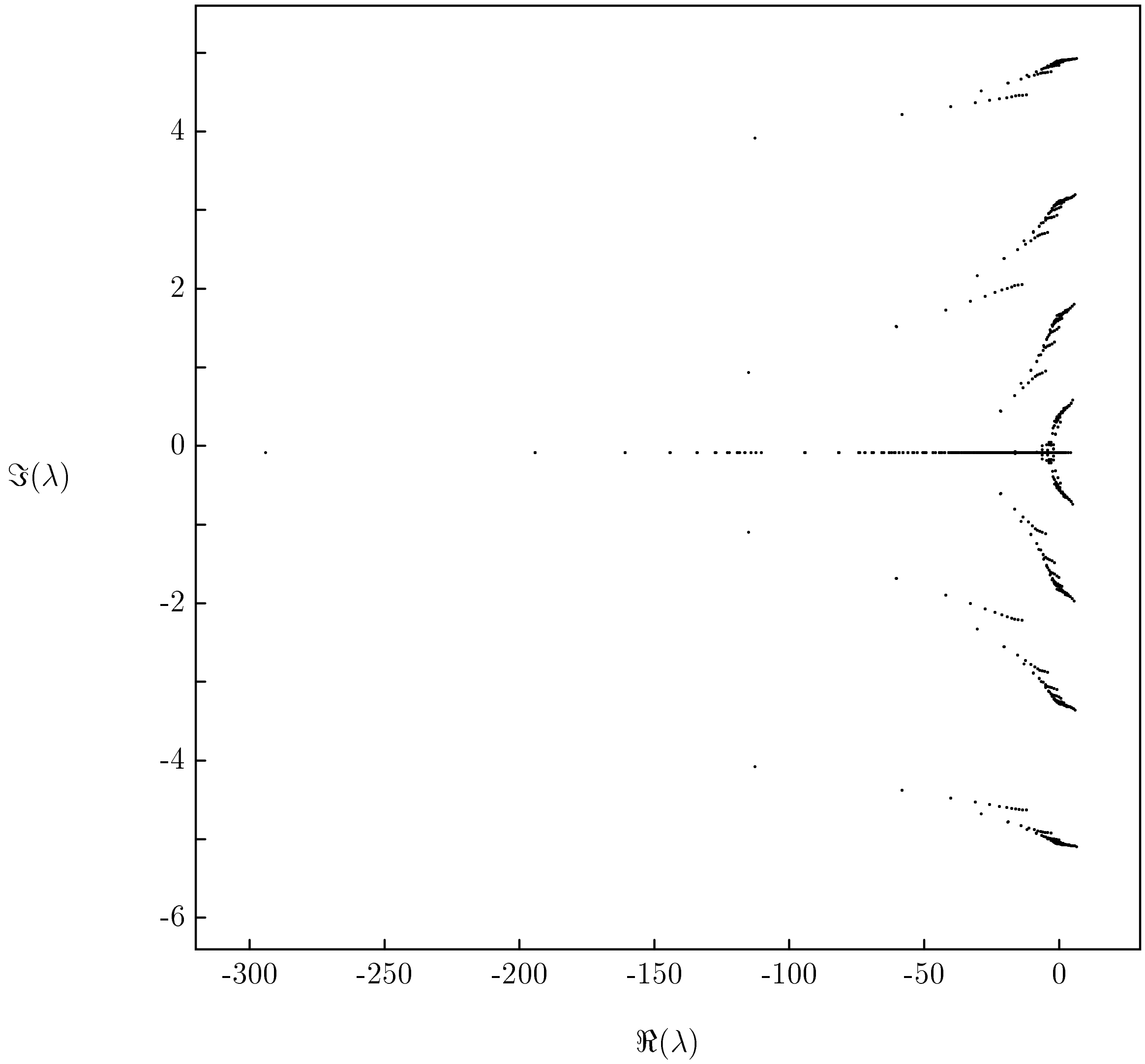


Figure C.5: Spectrum of A for Problem 5 with $N = 1728$

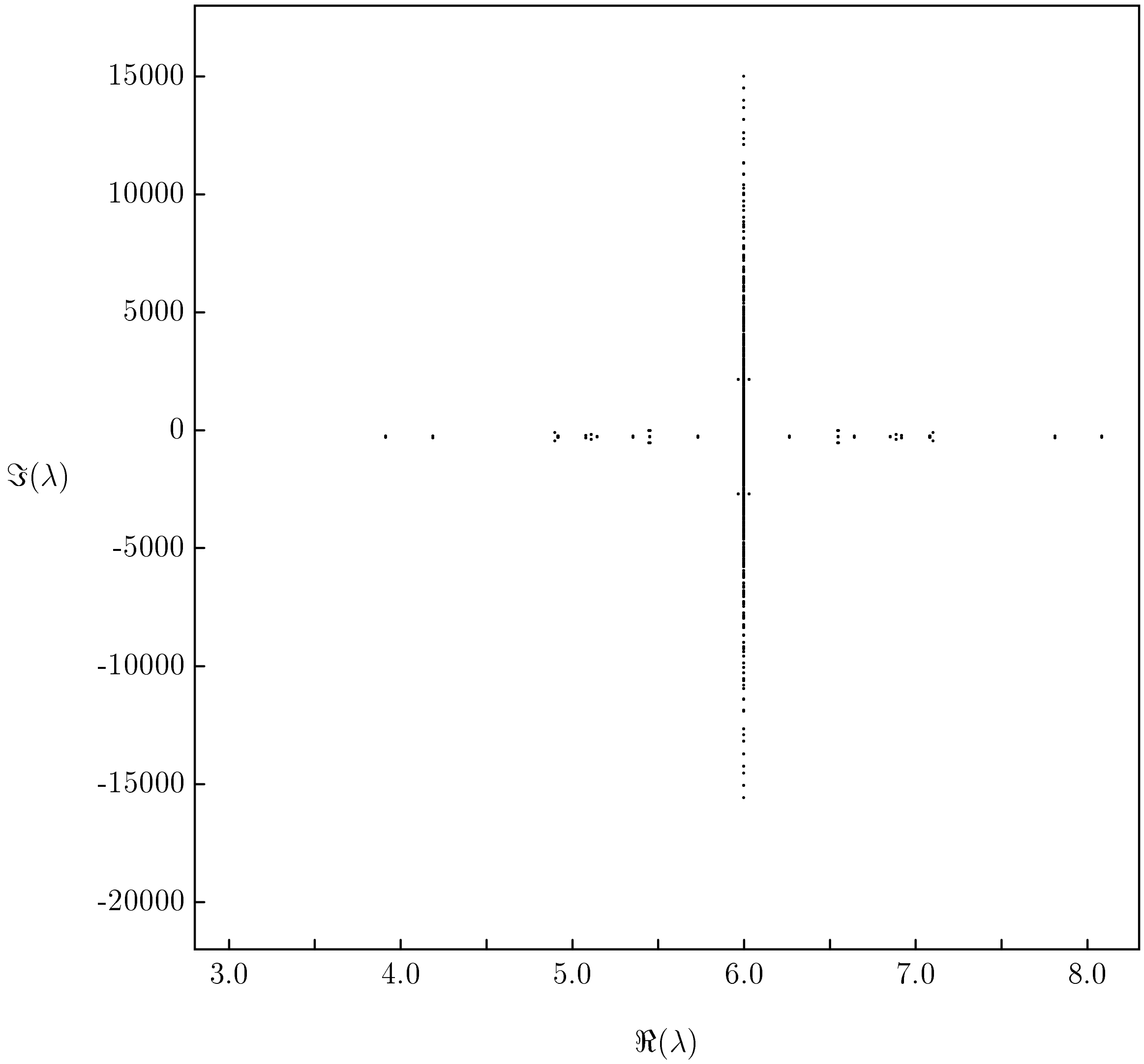


Figure C.6: Spectrum of A for Problem 6 with $N = 1728$

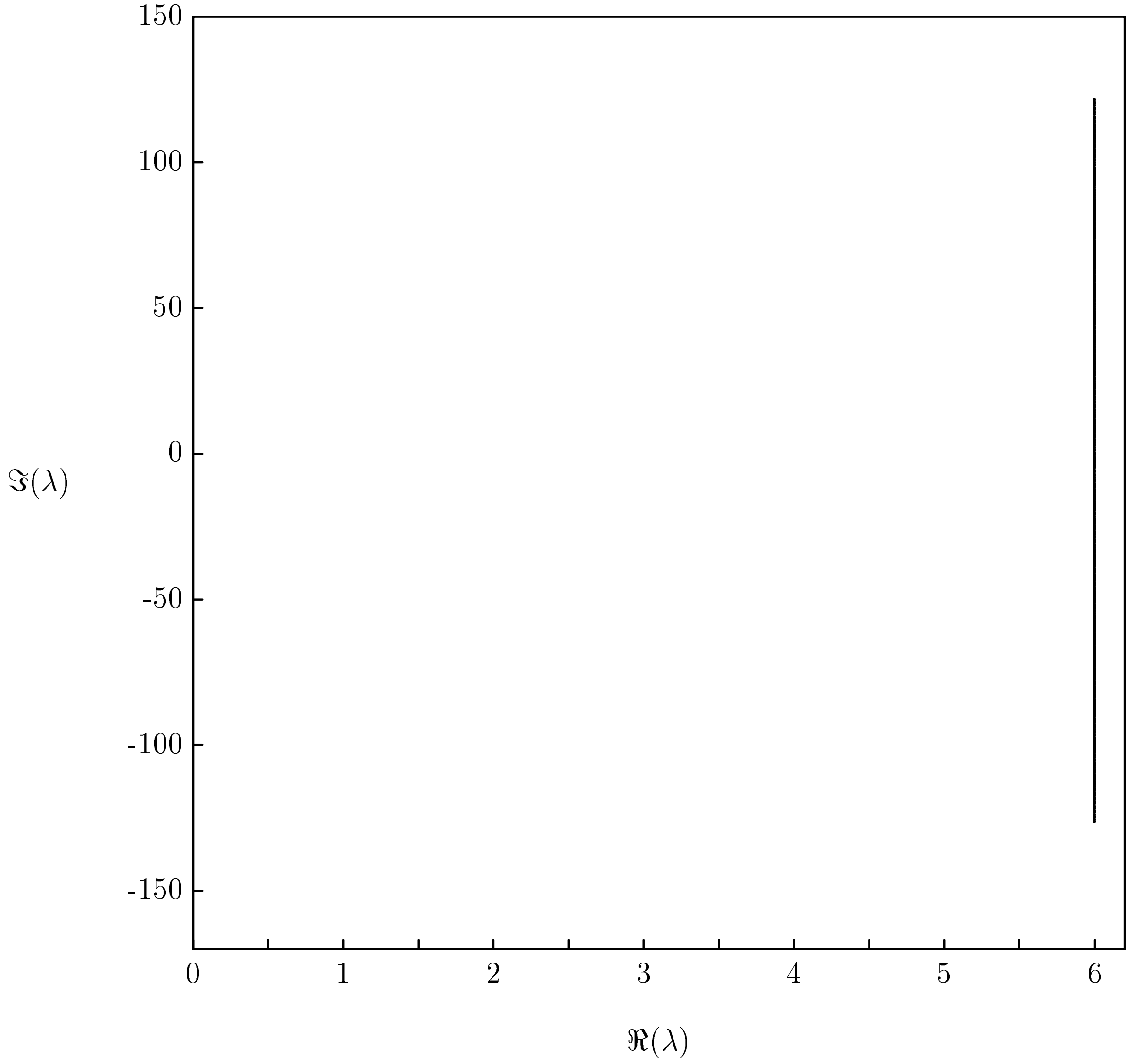


Figure C.7: Spectrum of A for Problem 7 with $N = 1728$

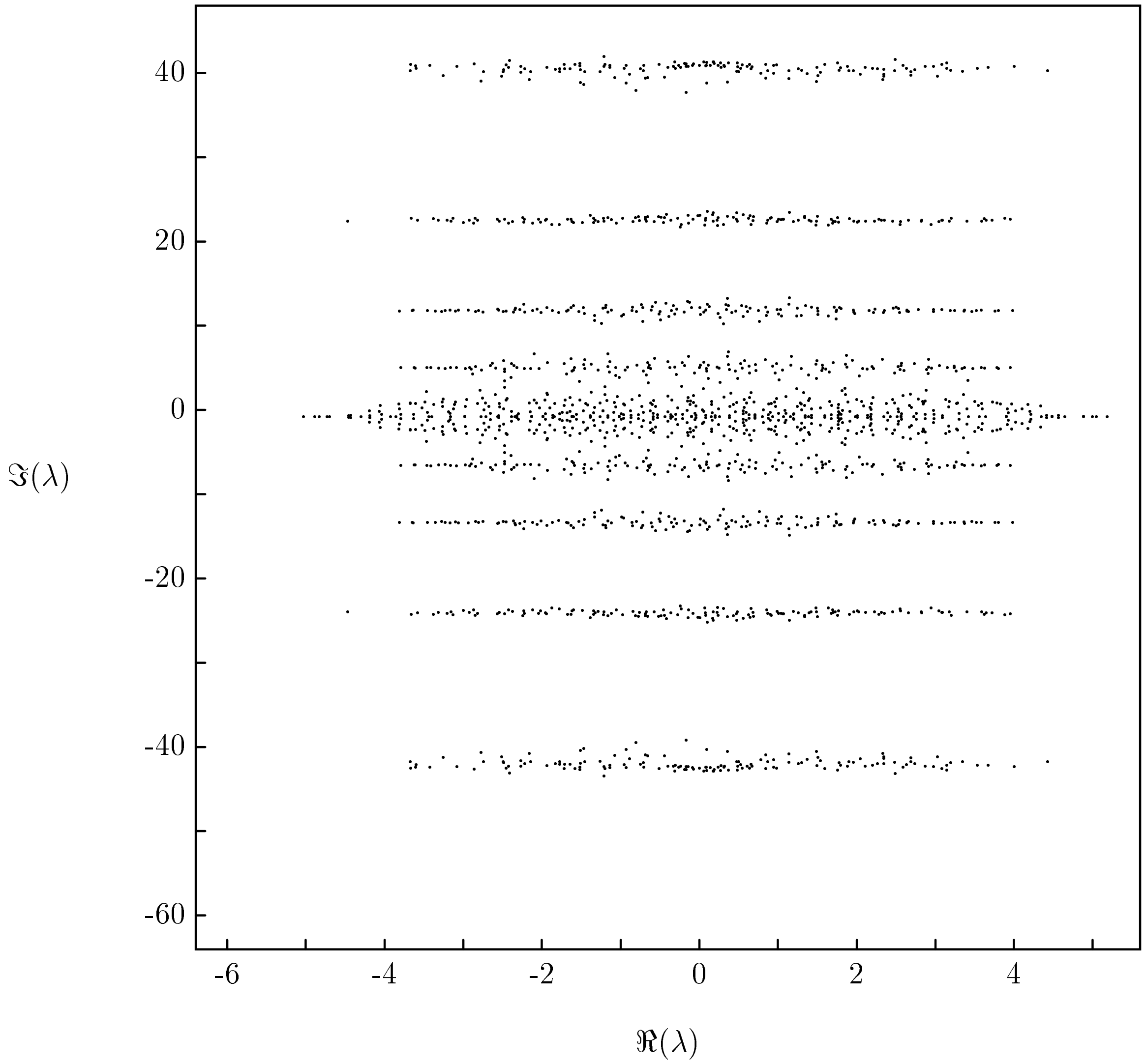


Figure C.8: Spectrum of A for Problem 8 with $N = 1728$

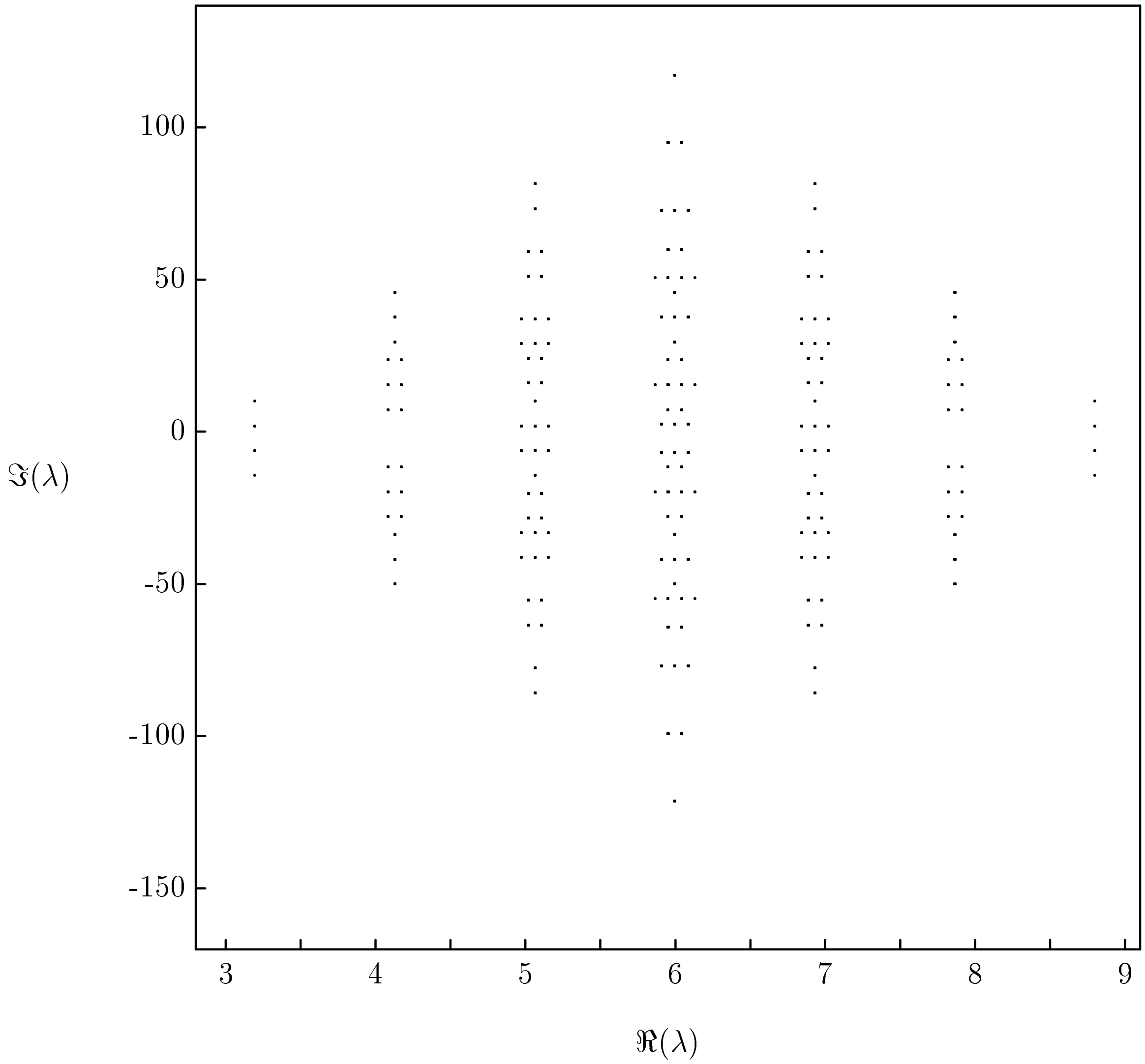


Figure C.9: Spectrum of A for Problem 9 with $N = 1728$

Appendix D

Spectra of $A^T A$ for Test Problems with $N = 1728$

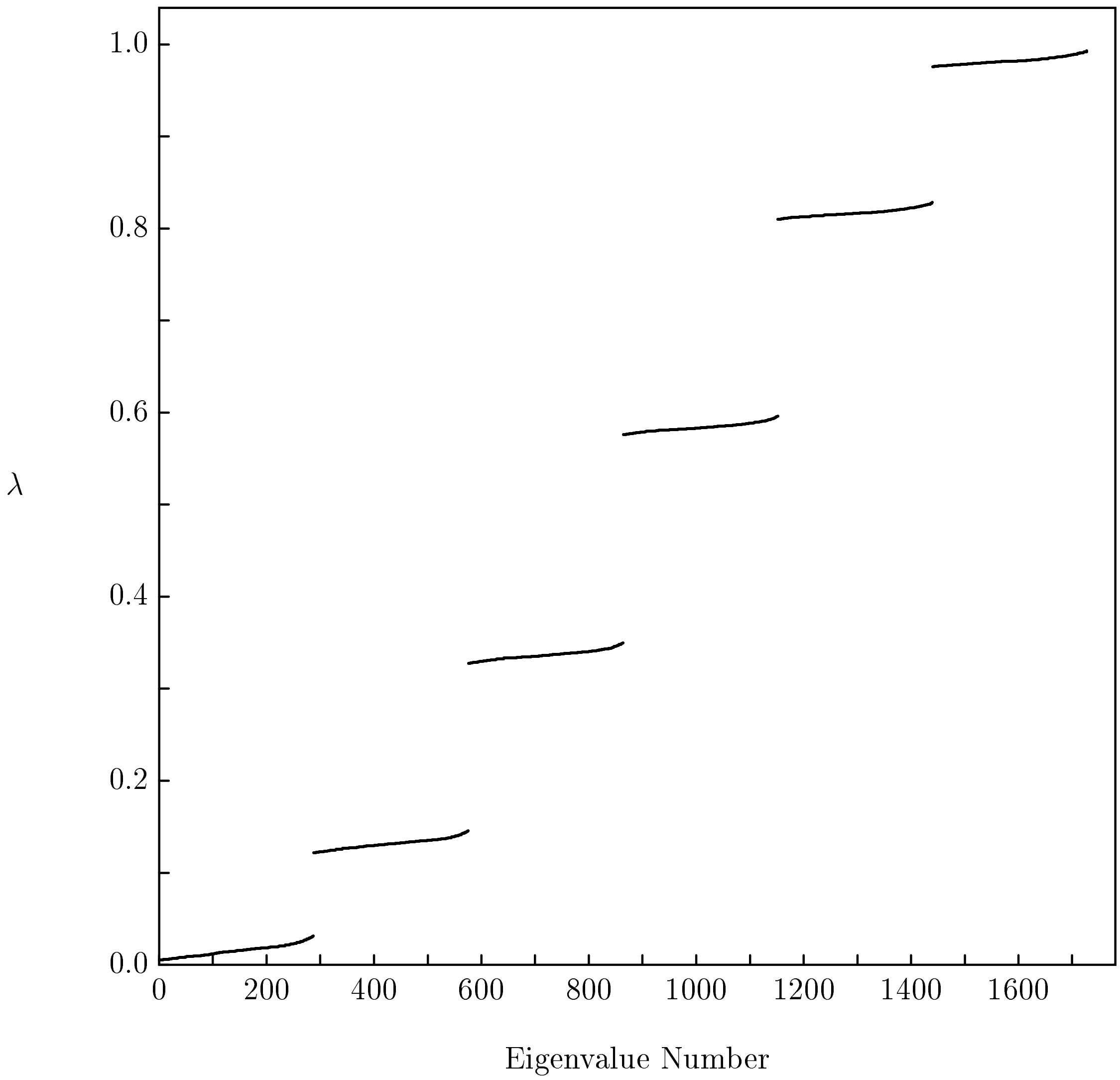


Figure D.1: Spectrum of $A^T A$ for Problem 1 with $N = 1728$

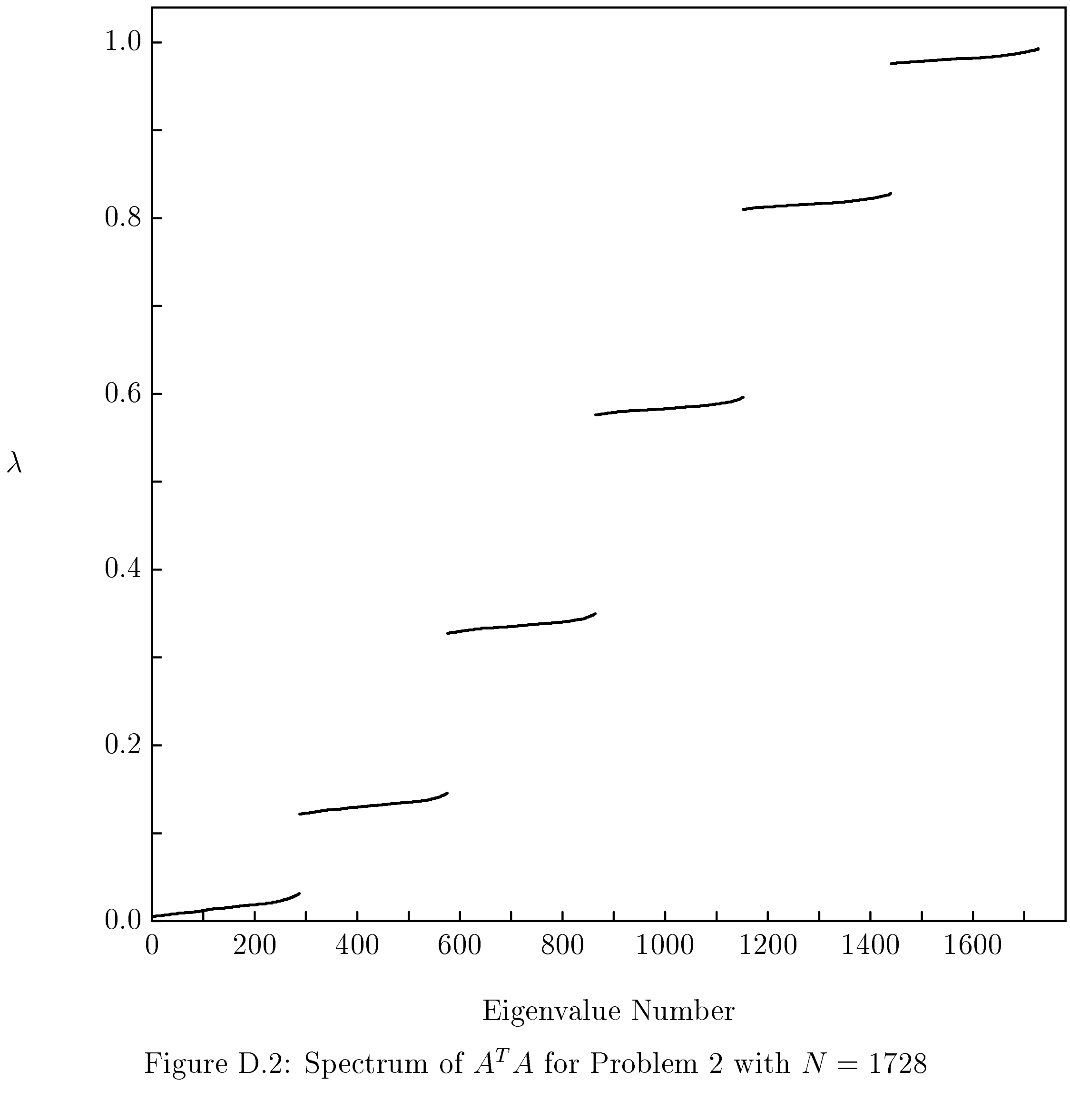


Figure D.2: Spectrum of $A^T A$ for Problem 2 with $N = 1728$

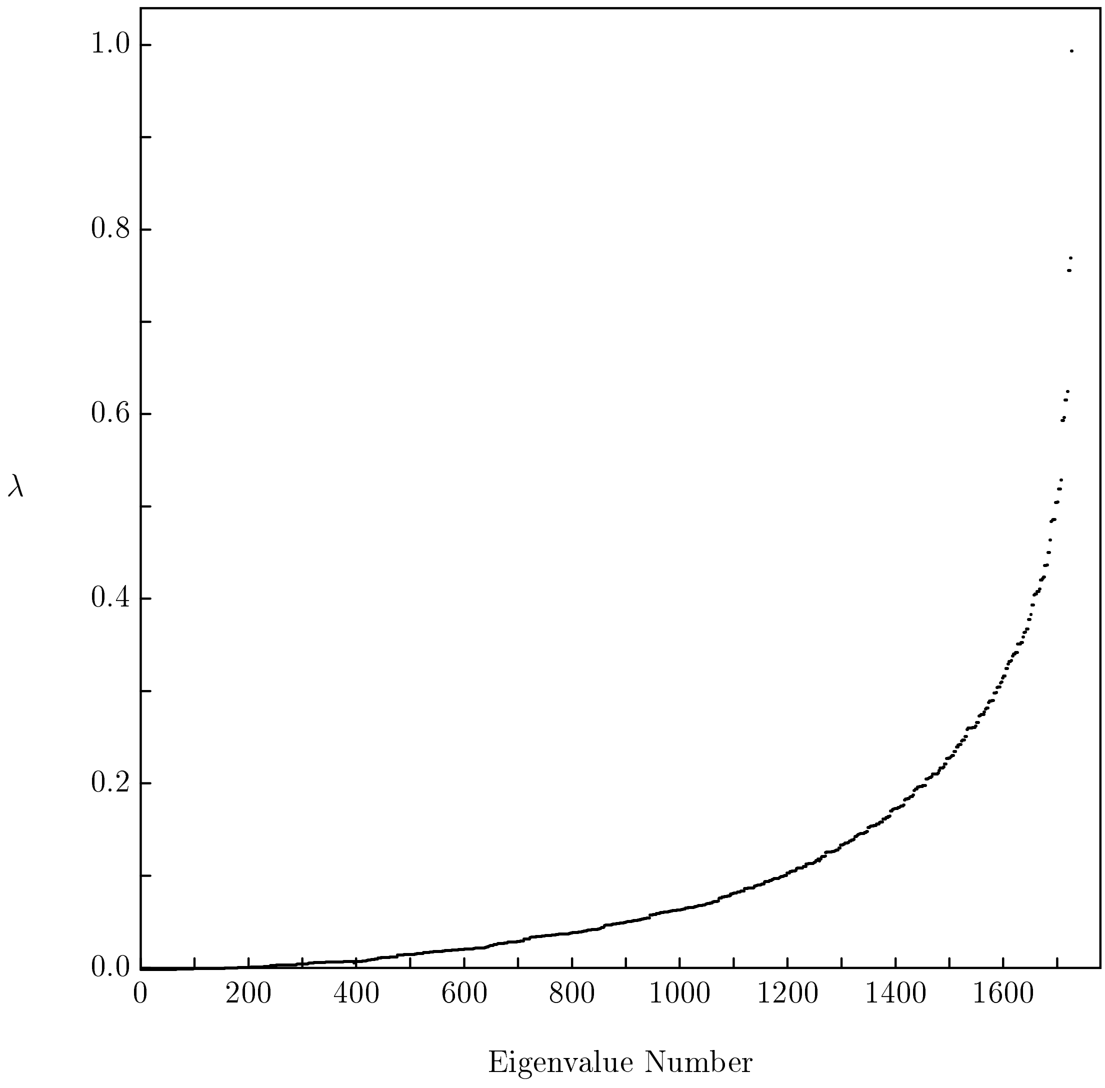


Figure D.3: Spectrum of $A^T A$ for Problem 3 with $N = 1728$

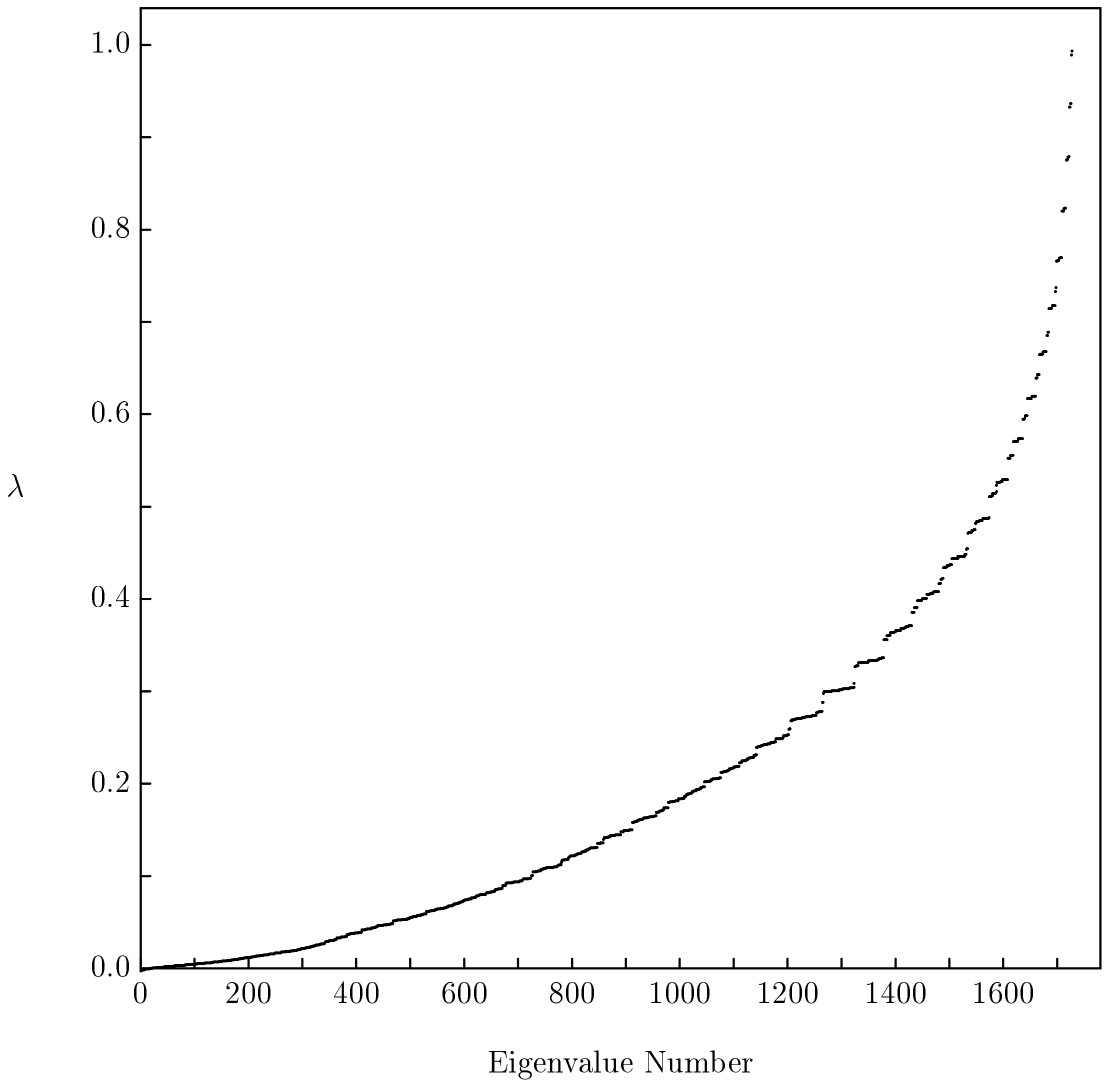


Figure D.4: Spectrum of $A^T A$ for Problem 4 with $N = 1728$

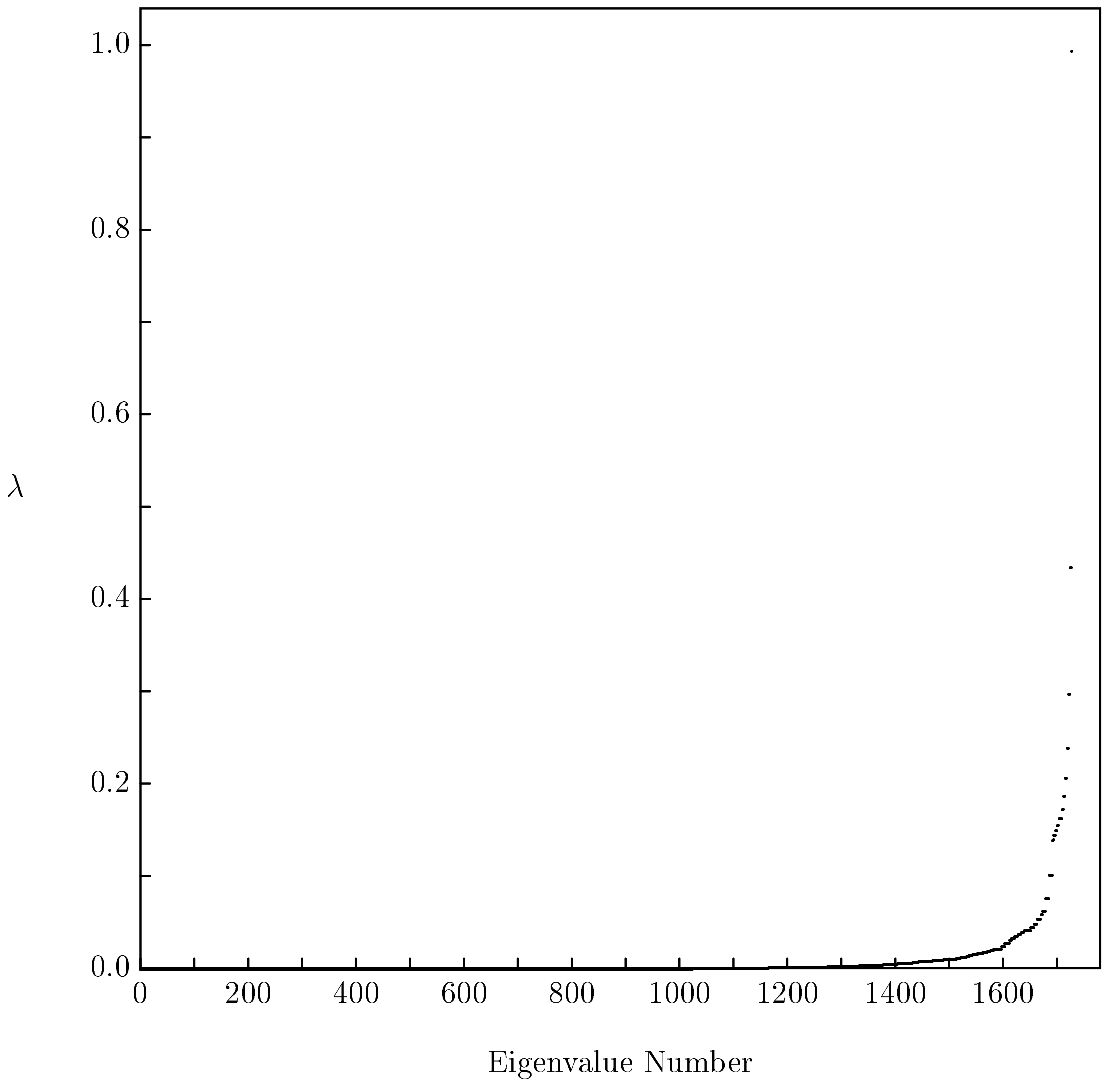


Figure D.5: Spectrum of $A^T A$ for Problem 5 with $N = 1728$

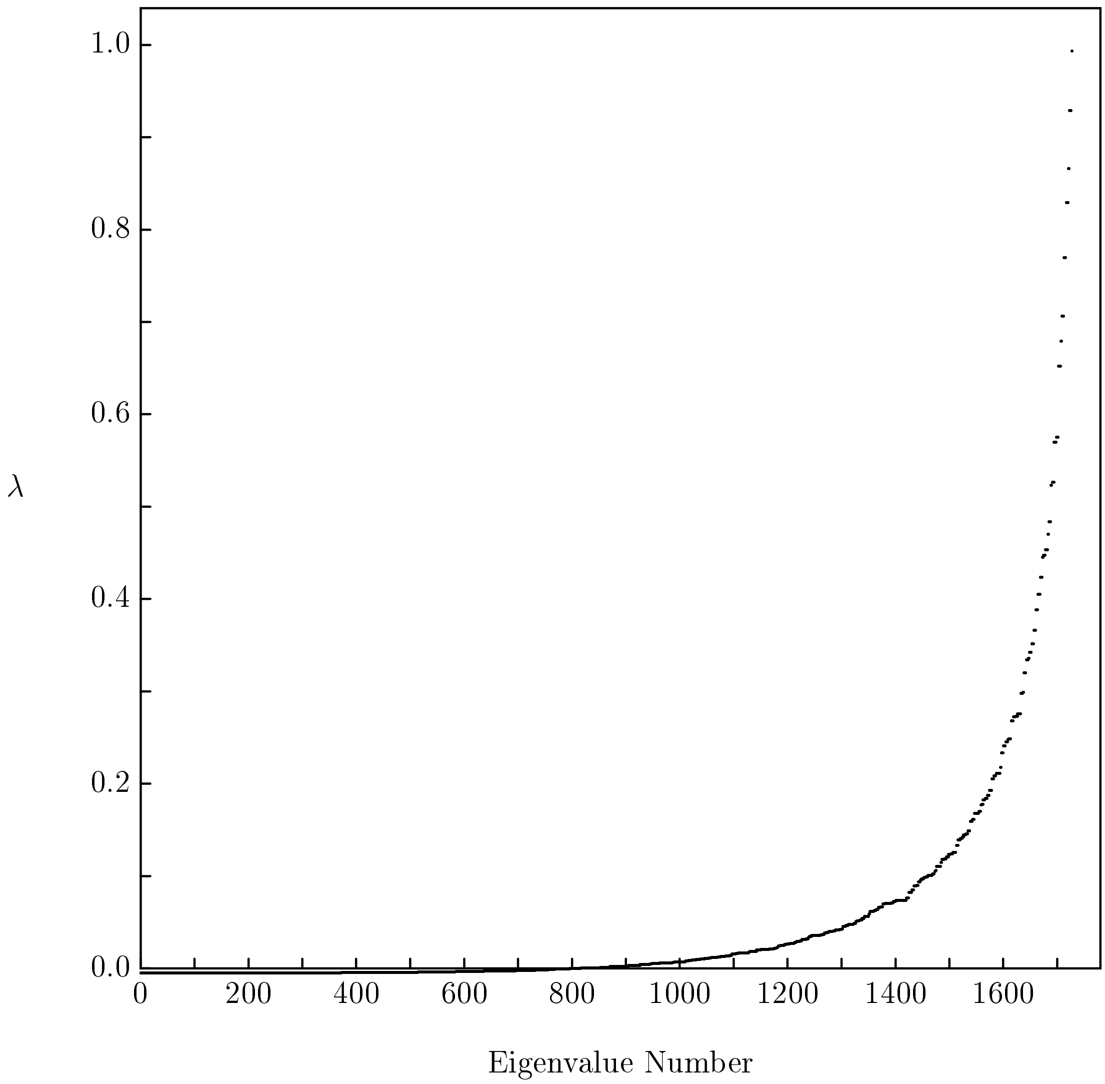


Figure D.6: Spectrum of $A^T A$ for Problem 6 with $N = 1728$

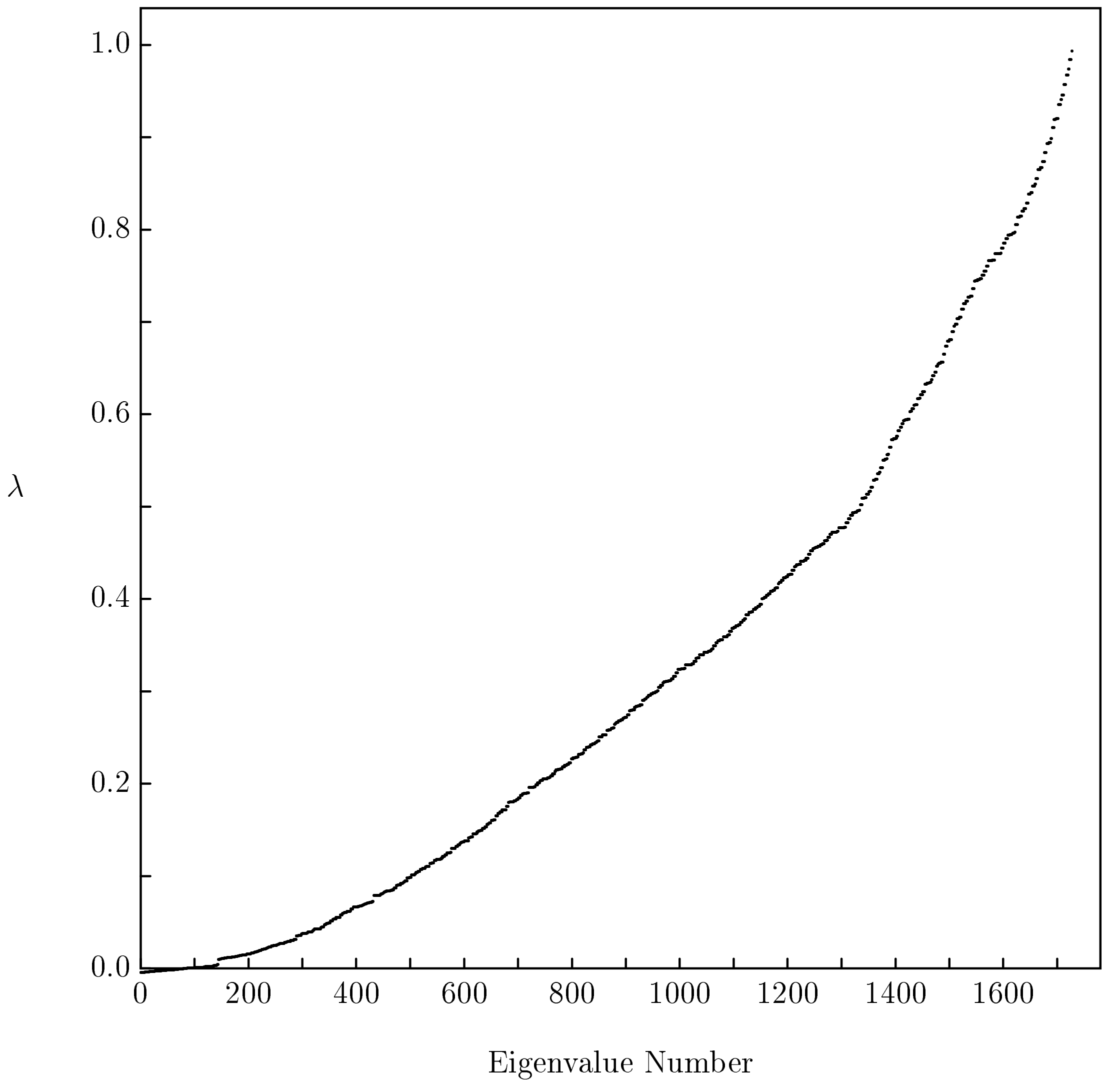


Figure D.7: Spectrum of $A^T A$ for Problem 7 with $N = 1728$

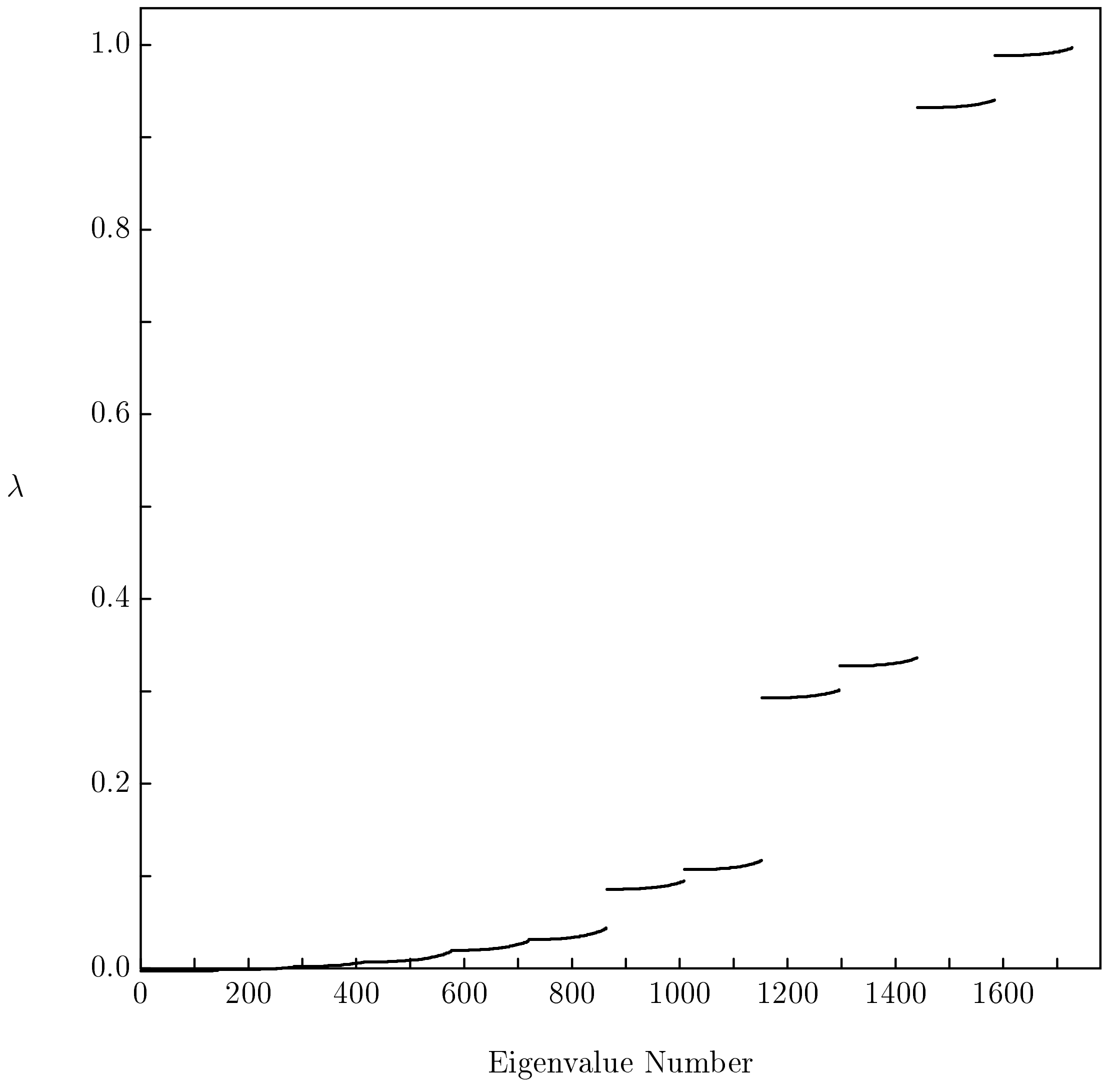


Figure D.8: Spectrum of $A^T A$ for Problem 8 with $N = 1728$

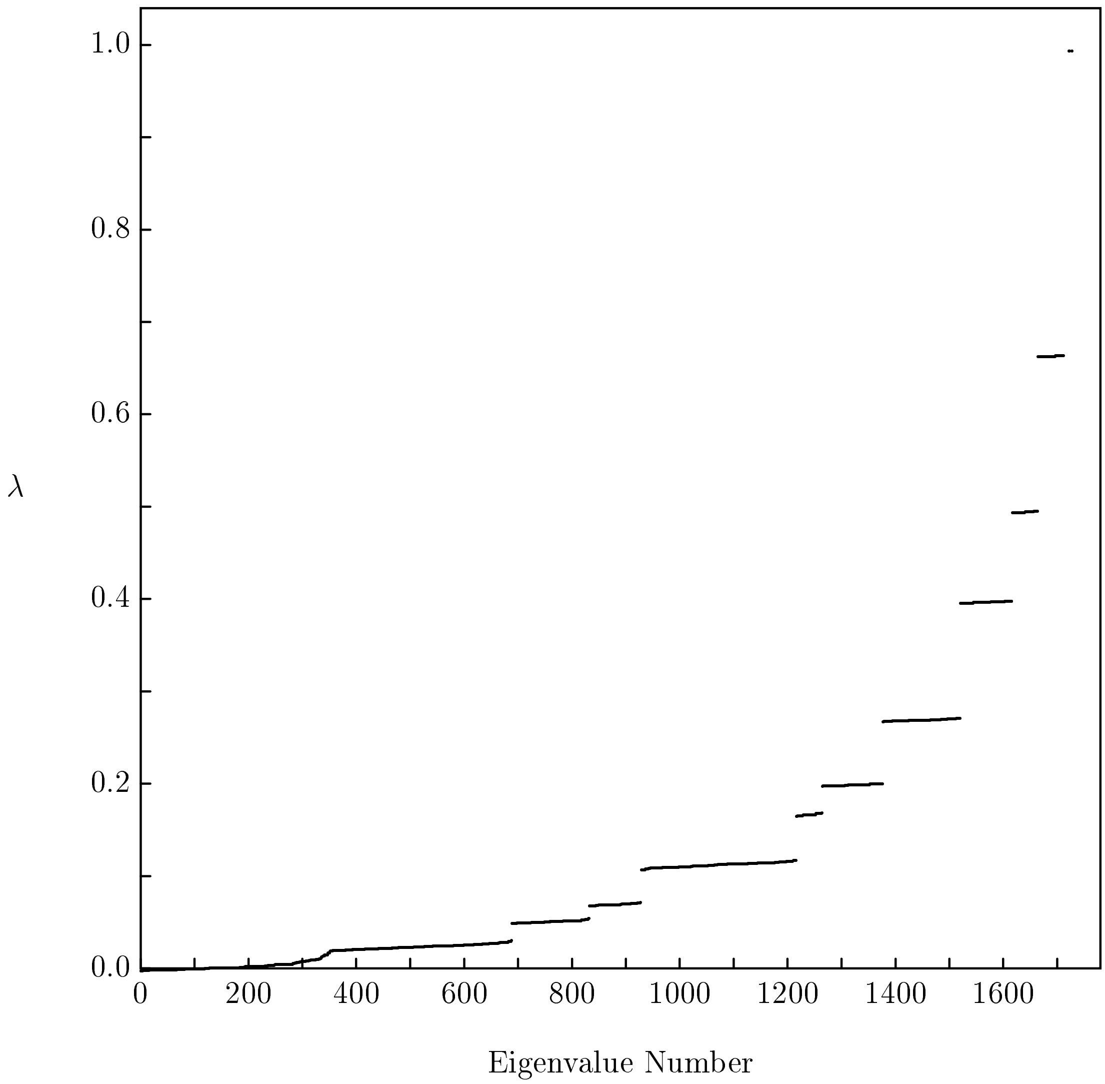


Figure D.9: Spectrum of $A^T A$ for Problem 9 with $N = 1728$

Appendix E

Spectra of the RP-9 System Matrix $I - Q_\pi$ for Test Problems with $N = 1728$

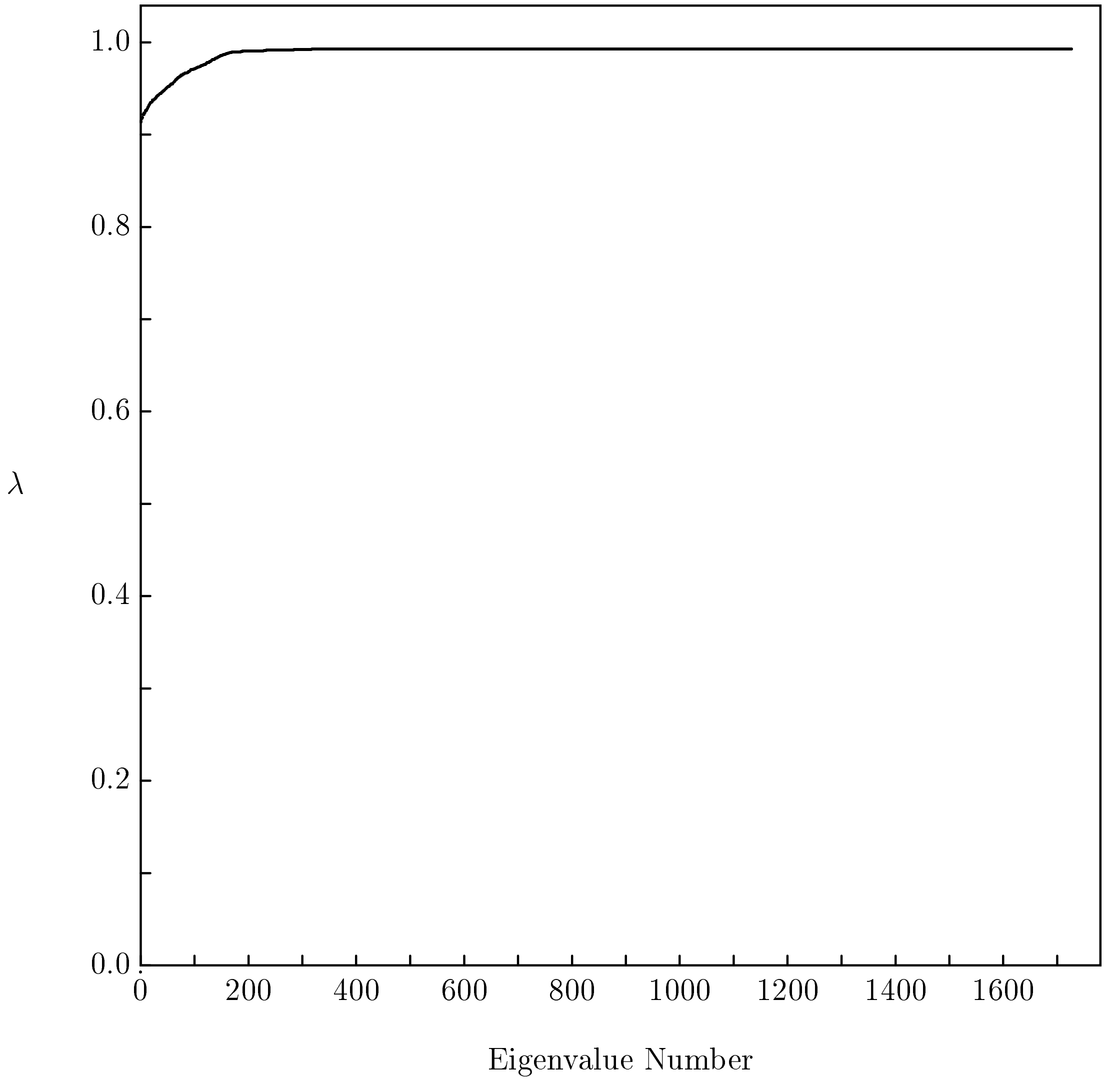


Figure E.1: Spectrum of $I - Q_\pi$ for Problem 1 with $N = 1728$

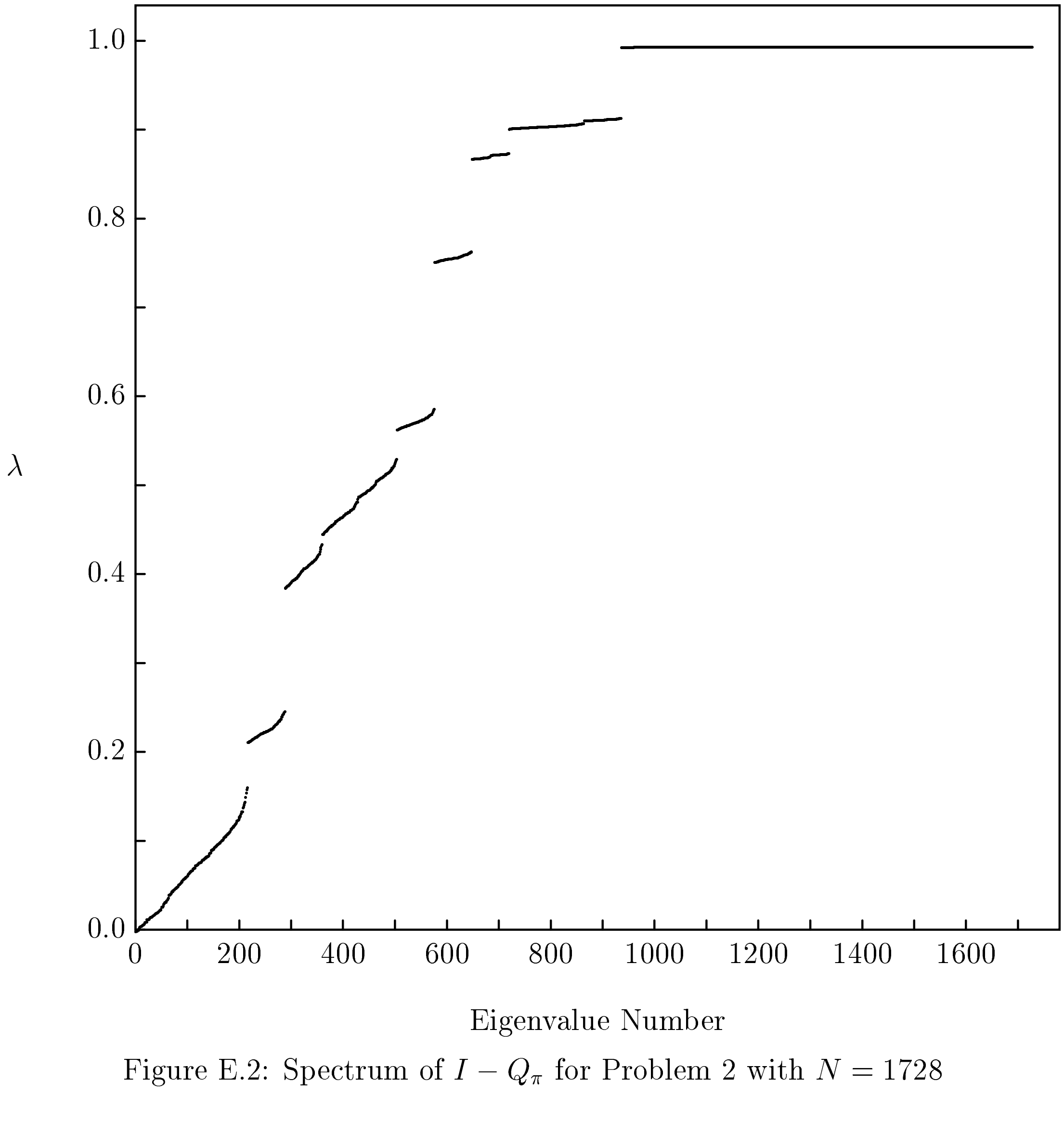


Figure E.2: Spectrum of $I - Q_\pi$ for Problem 2 with $N = 1728$

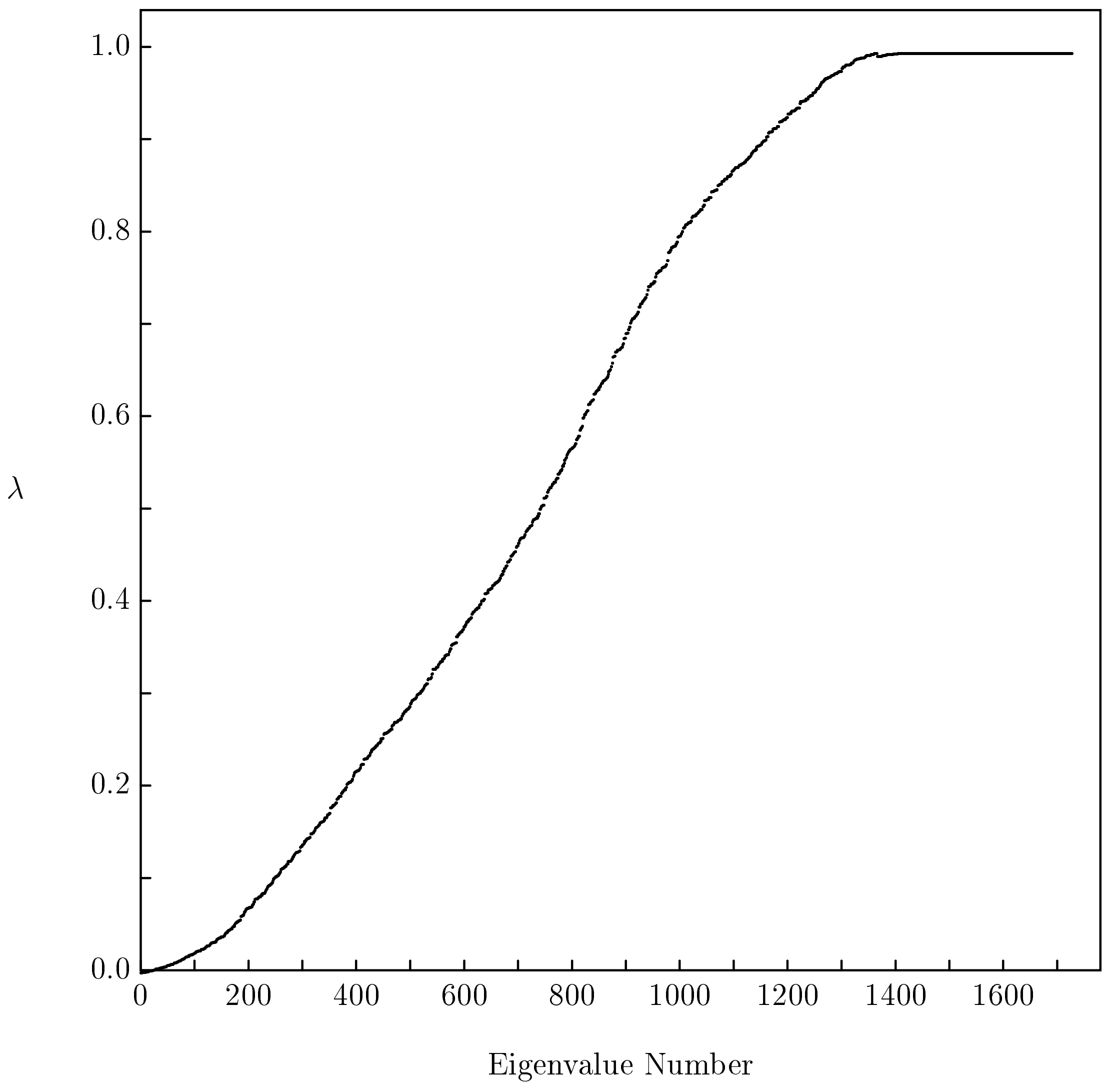


Figure E.3: Spectrum of $I - Q_\pi$ for Problem 3 with $N = 1728$

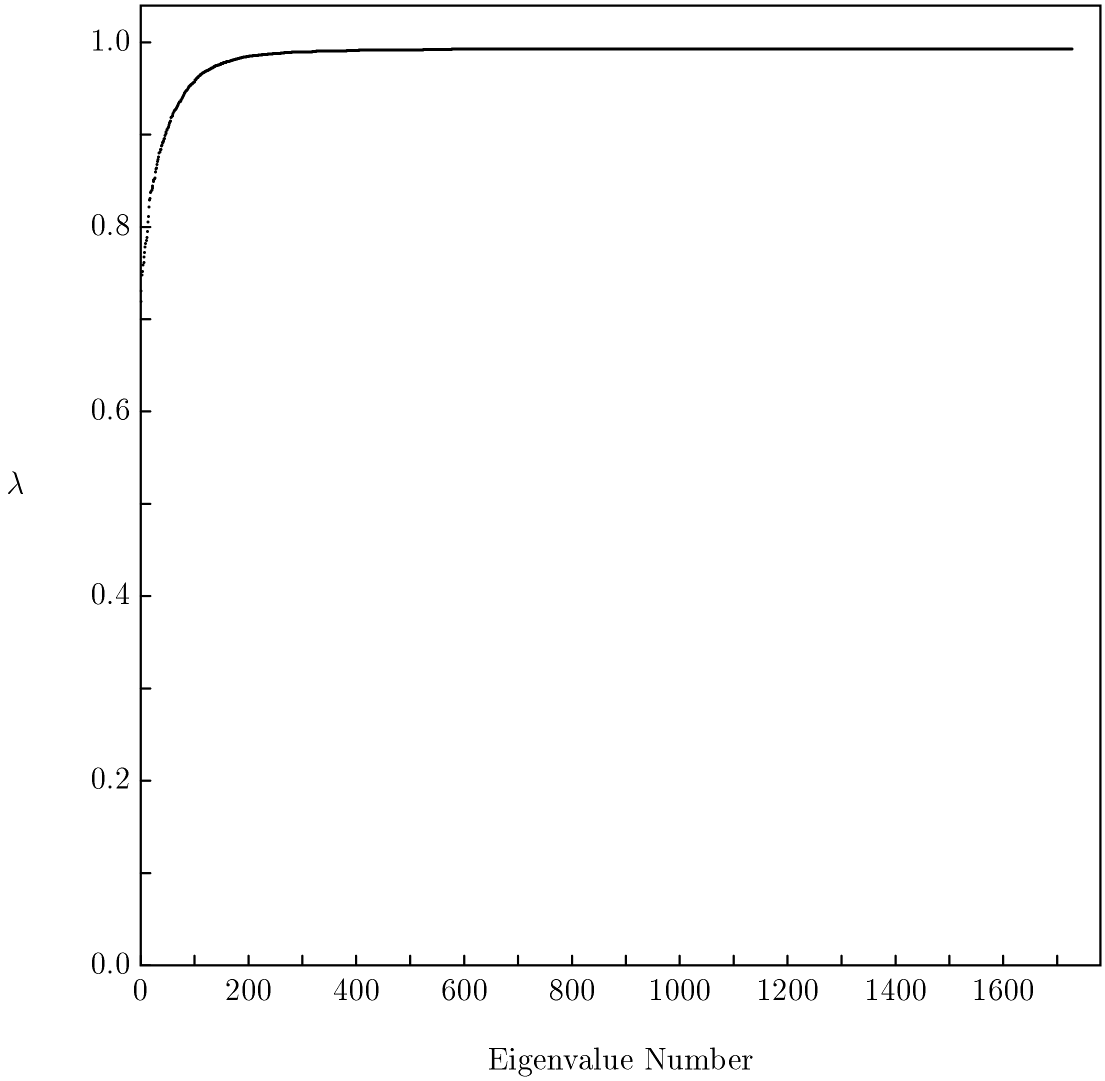


Figure E.4: Spectrum of $I - Q_\pi$ for Problem 4 with $N = 1728$

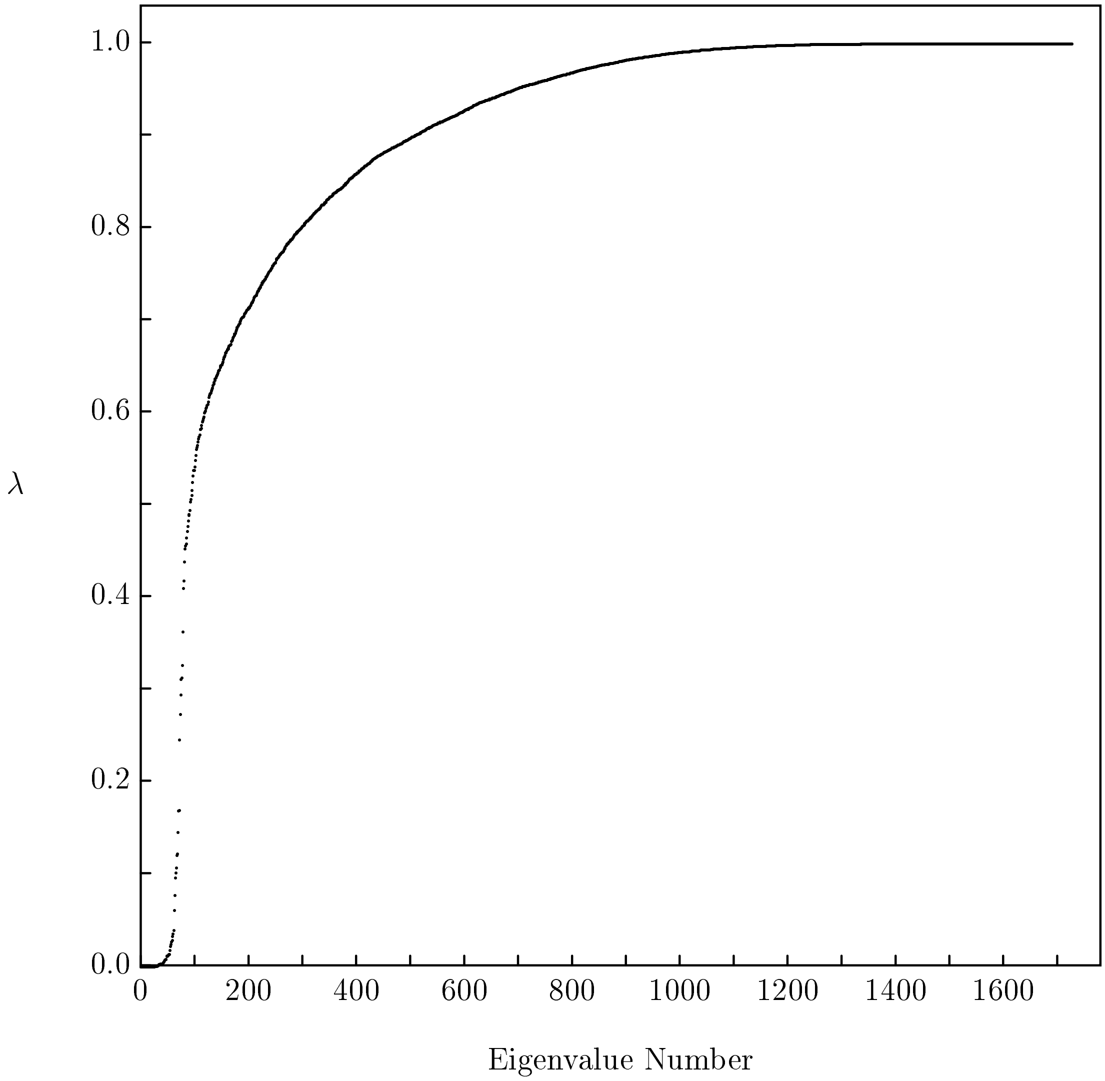


Figure E.5: Spectrum of $I - Q_\pi$ for Problem 5 with $N = 1728$

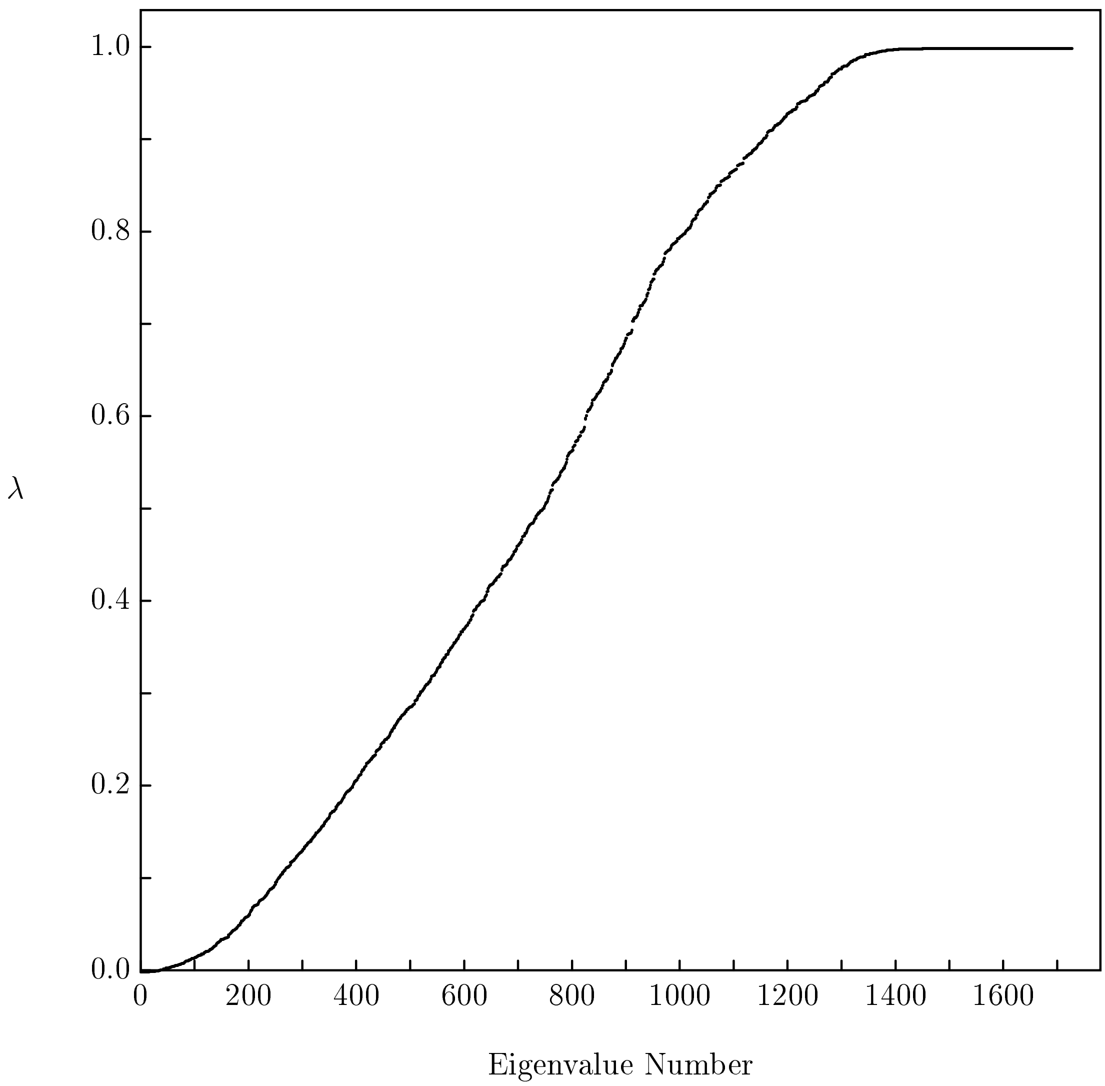


Figure E.6: Spectrum of $I - Q_\pi$ for Problem 6 with $N = 1728$

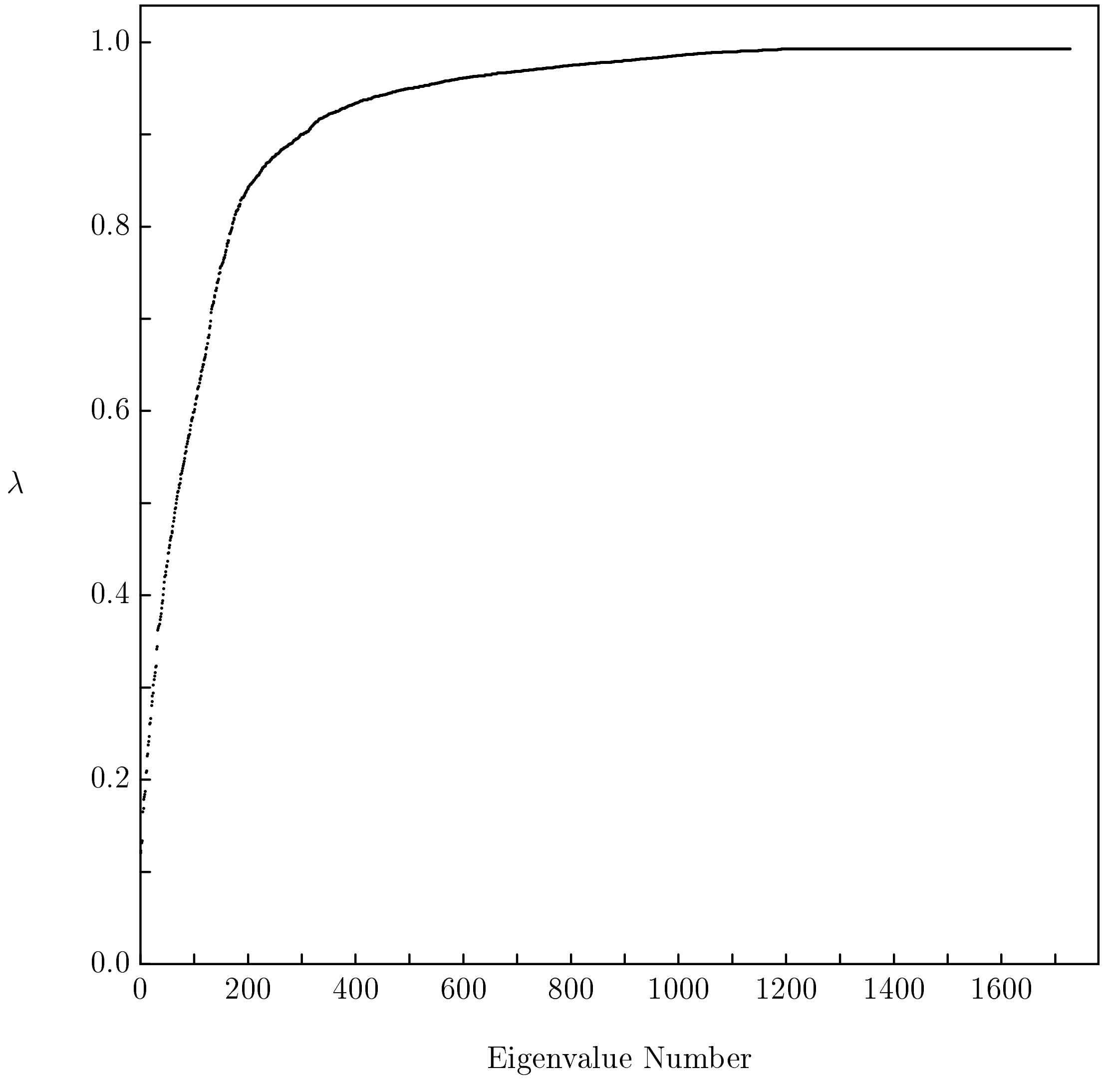


Figure E.7: Spectrum of $I - Q_\pi$ for Problem 7 with $N = 1728$

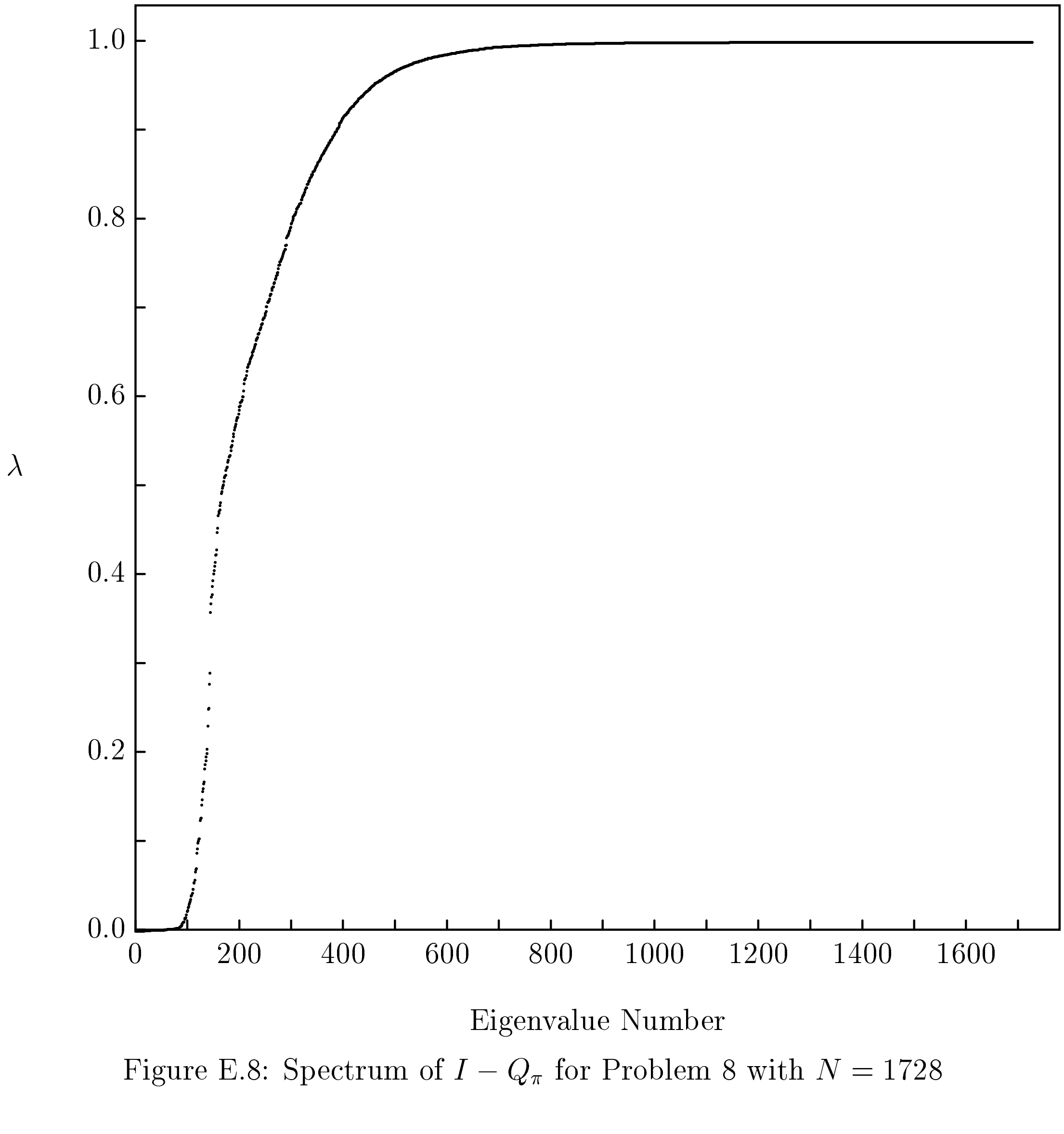


Figure E.8: Spectrum of $I - Q_\pi$ for Problem 8 with $N = 1728$

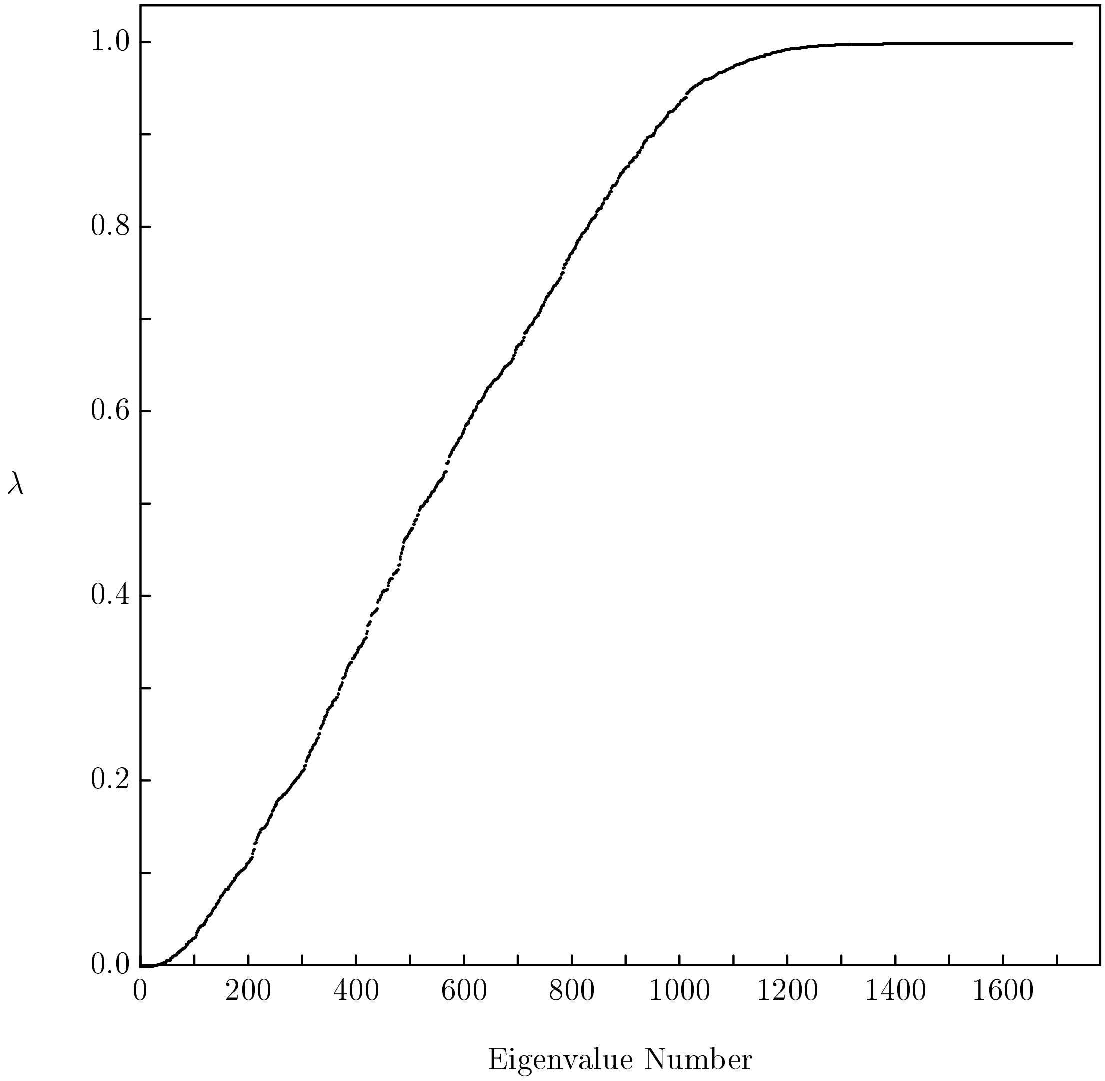


Figure E.9: Spectrum of $I - Q_\pi$ for Problem 9 with $N = 1728$

Appendix F

Spectra of SUM System Matrix $I - Q_\sigma$ for Test Problems with $N = 1728$

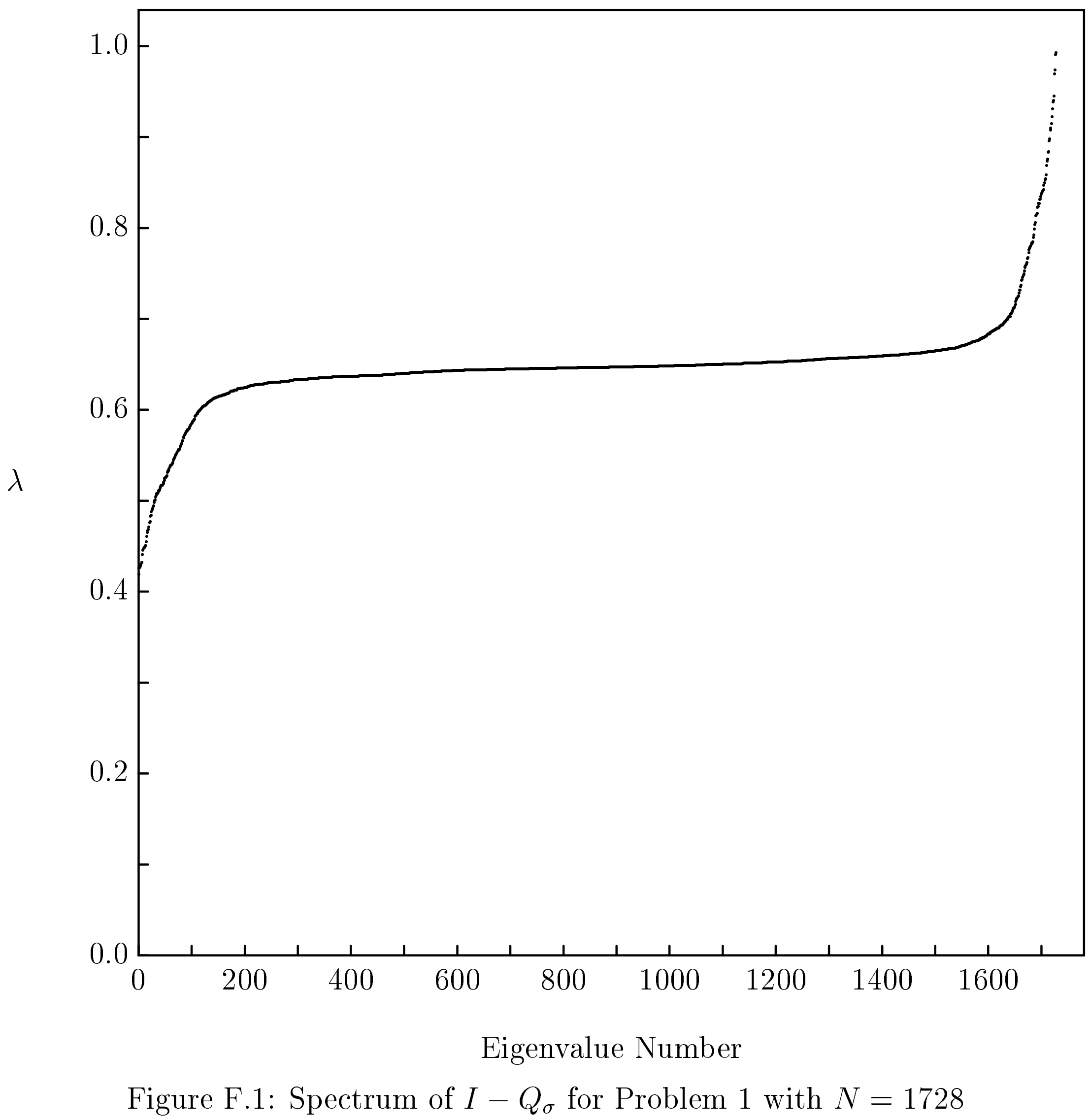


Figure F.1: Spectrum of $I - Q_\sigma$ for Problem 1 with $N = 1728$

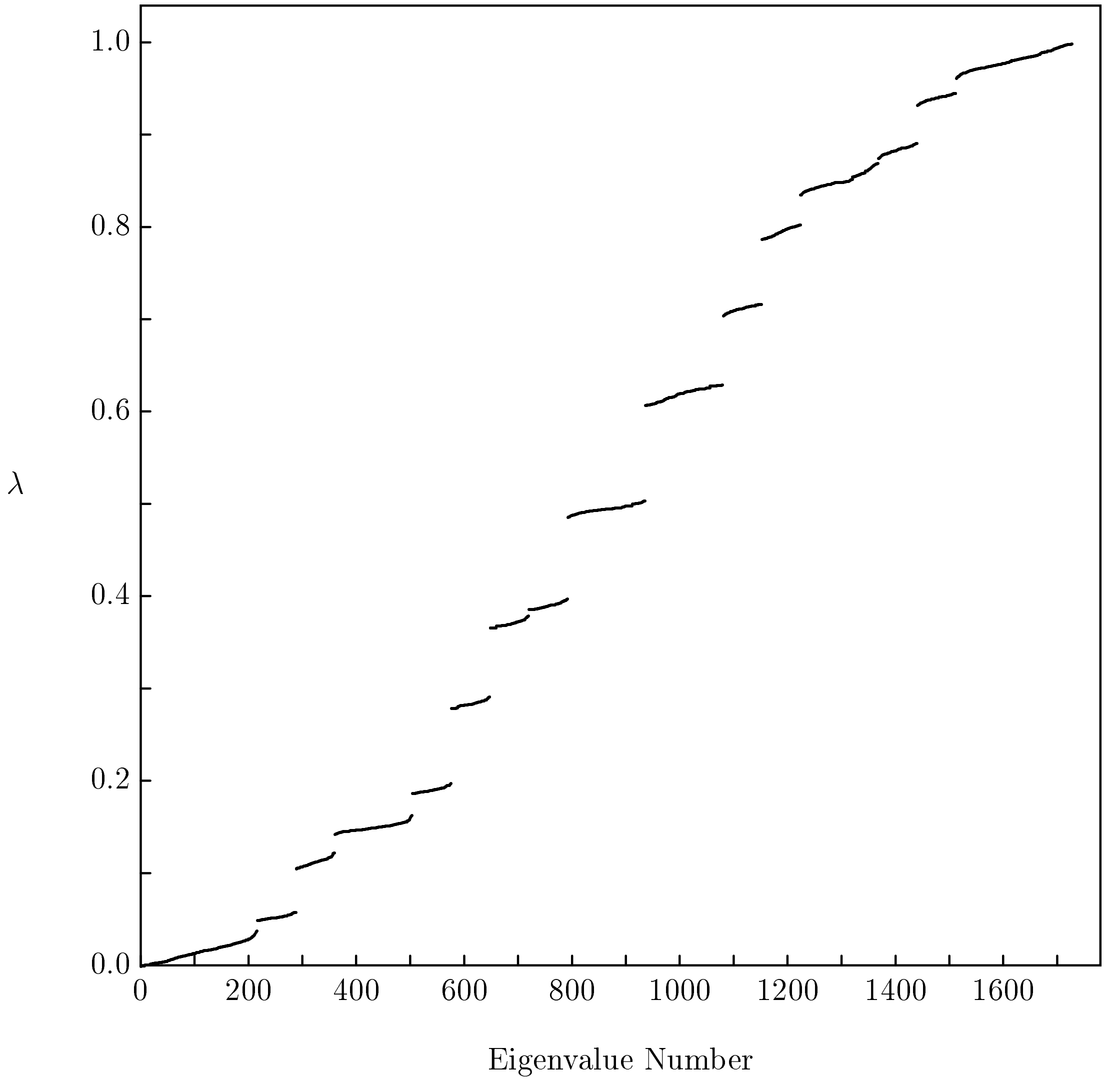


Figure F.2: Spectrum of $I - Q_\sigma$ for Problem 2 with $N = 1728$

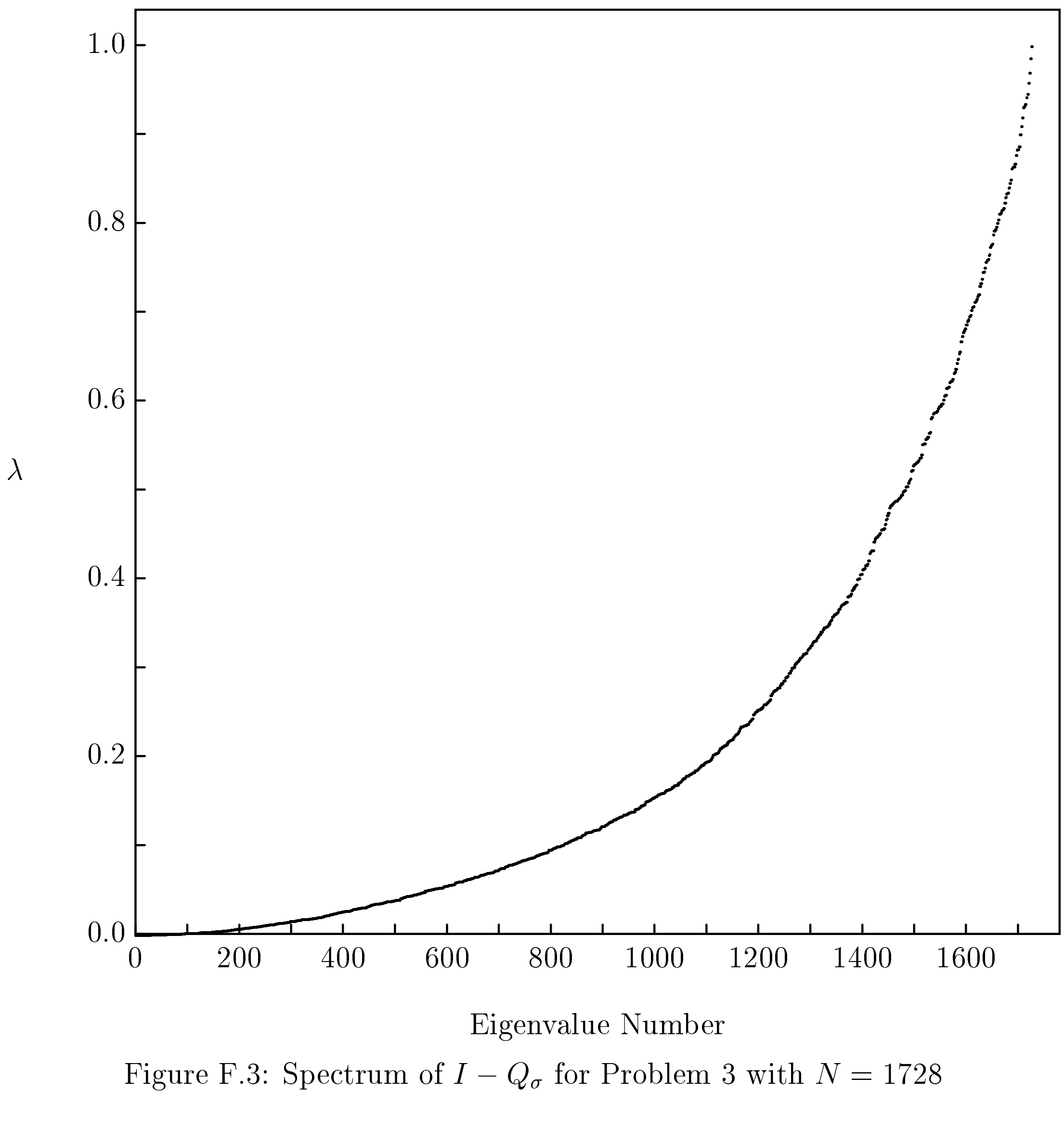


Figure F.3: Spectrum of $I - Q_\sigma$ for Problem 3 with $N = 1728$

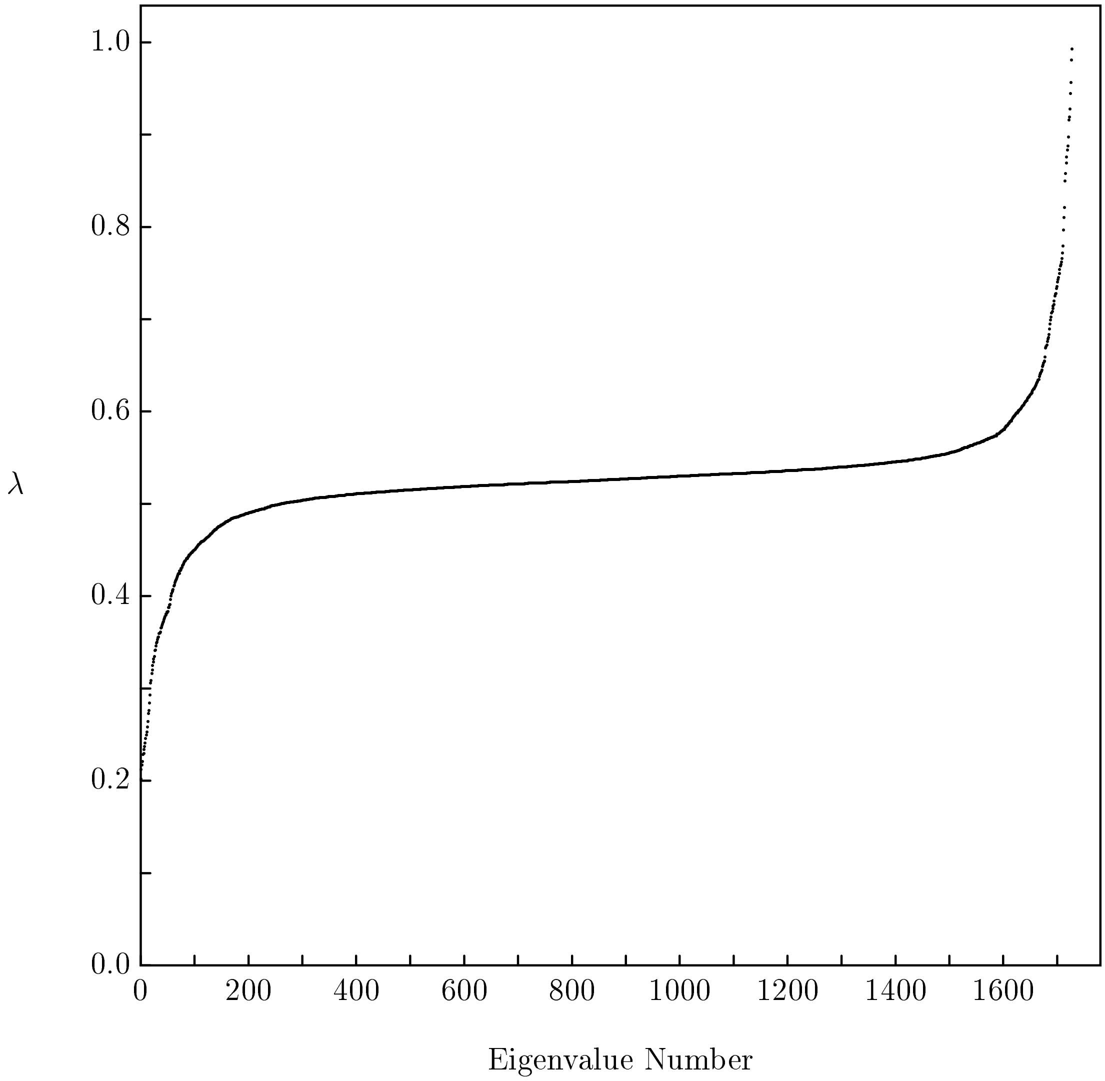


Figure F.4: Spectrum of $I - Q_\sigma$ for Problem 4 with $N = 1728$

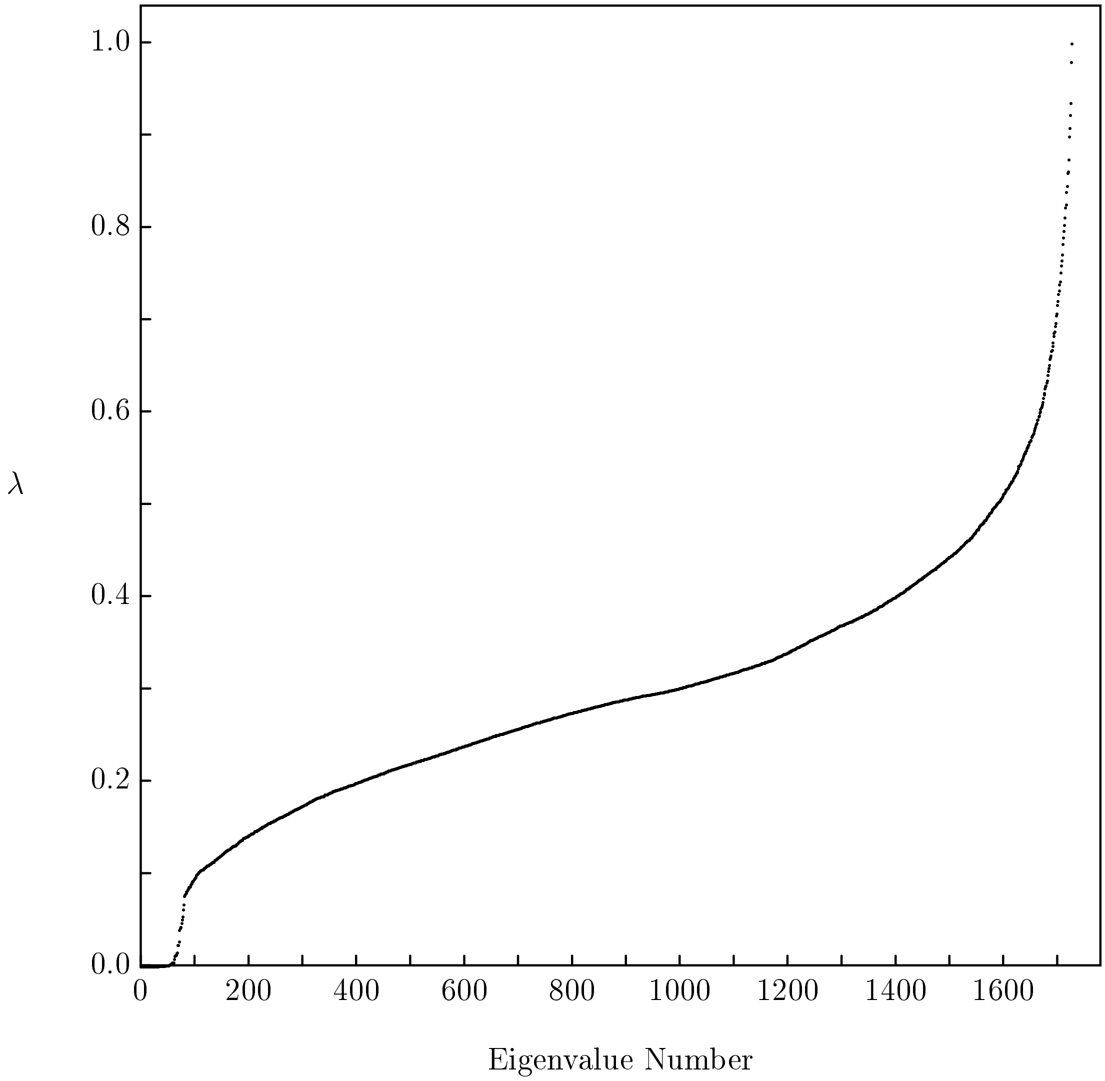


Figure F.5: Spectrum of $I - Q_\sigma$ for Problem 5 with $N = 1728$

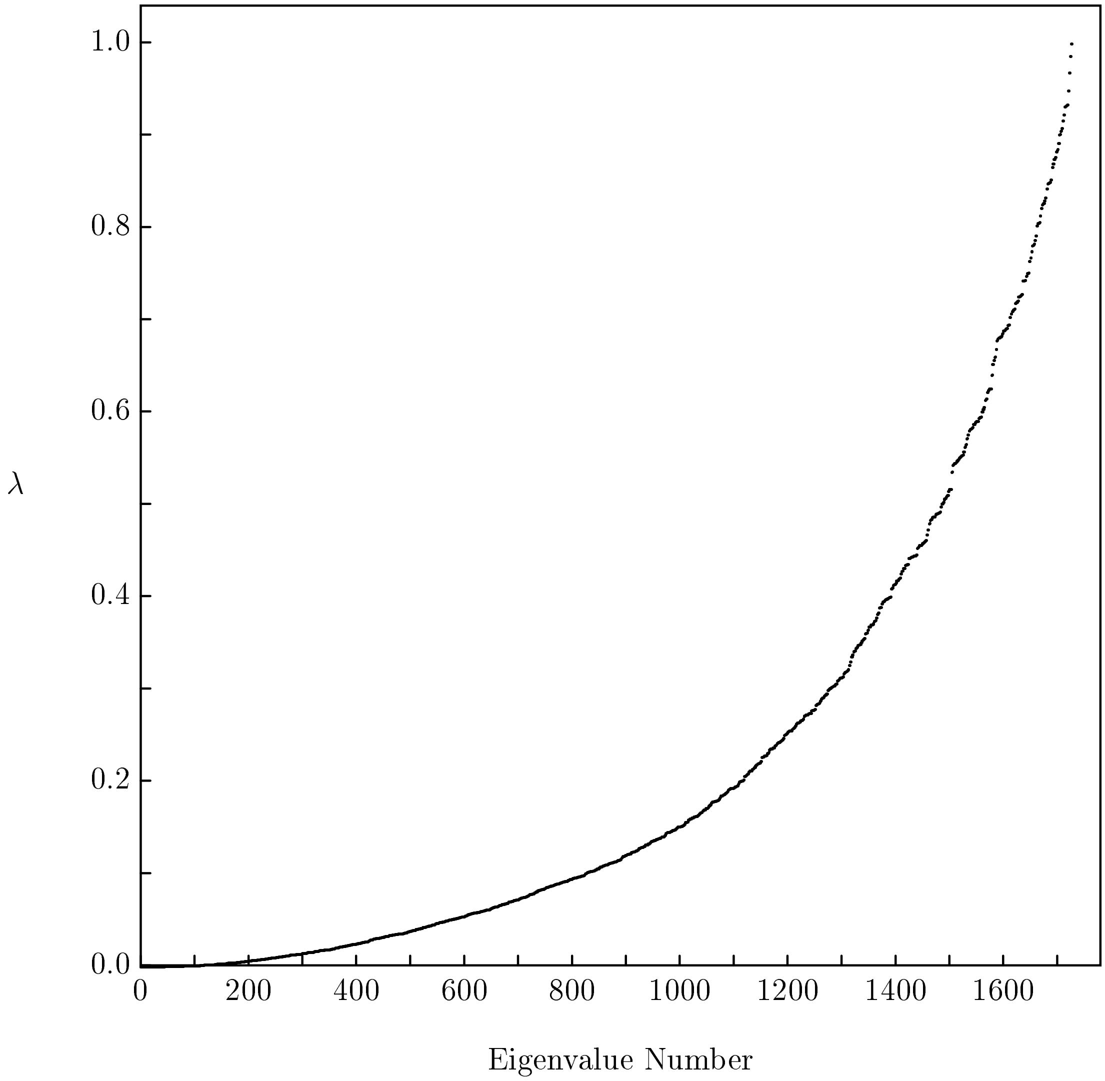


Figure F.6: Spectrum of $I - Q_\sigma$ for Problem 6 with $N = 1728$

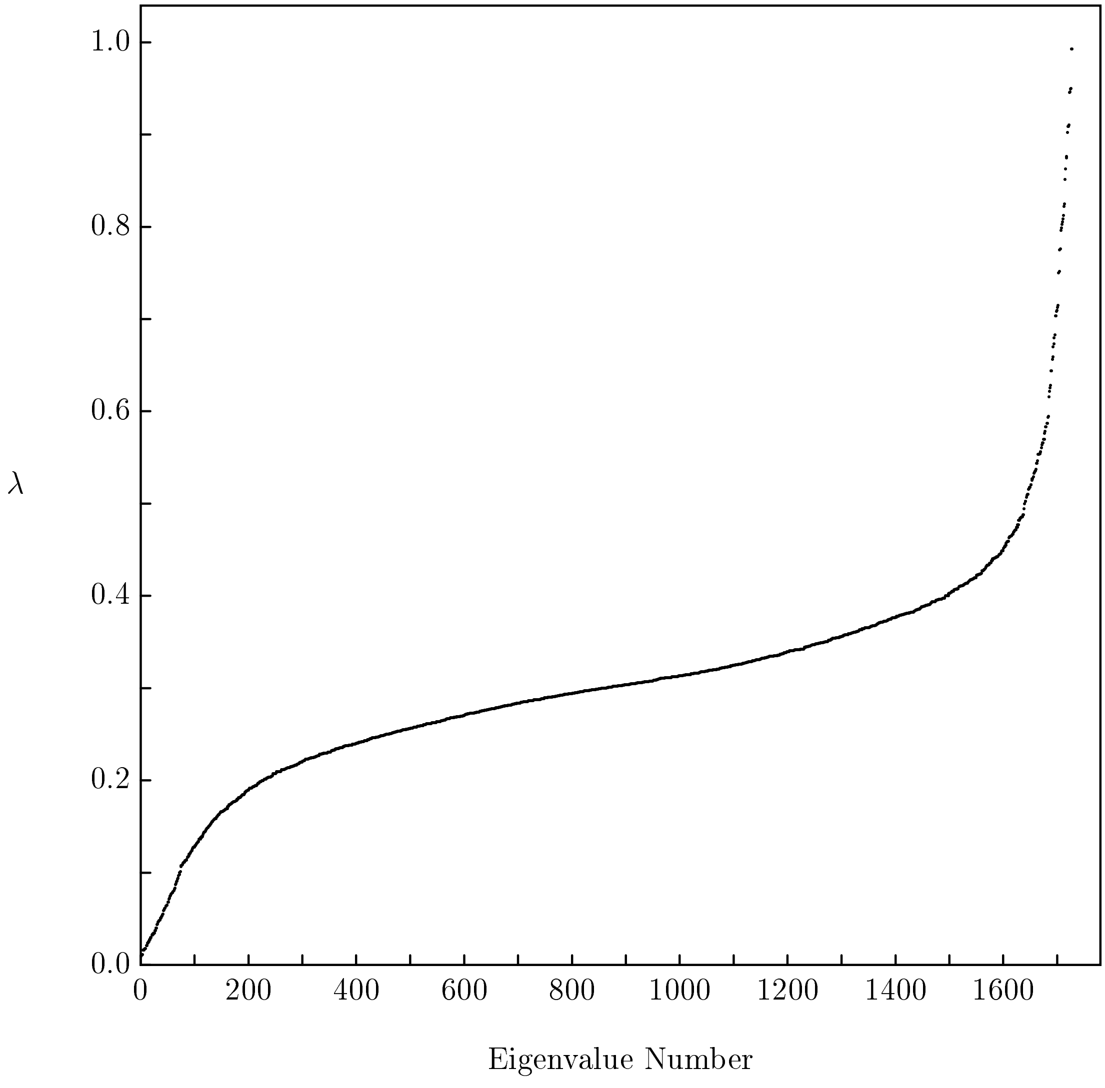


Figure F.7: Spectrum of $I - Q_\sigma$ for Problem 7 with $N = 1728$

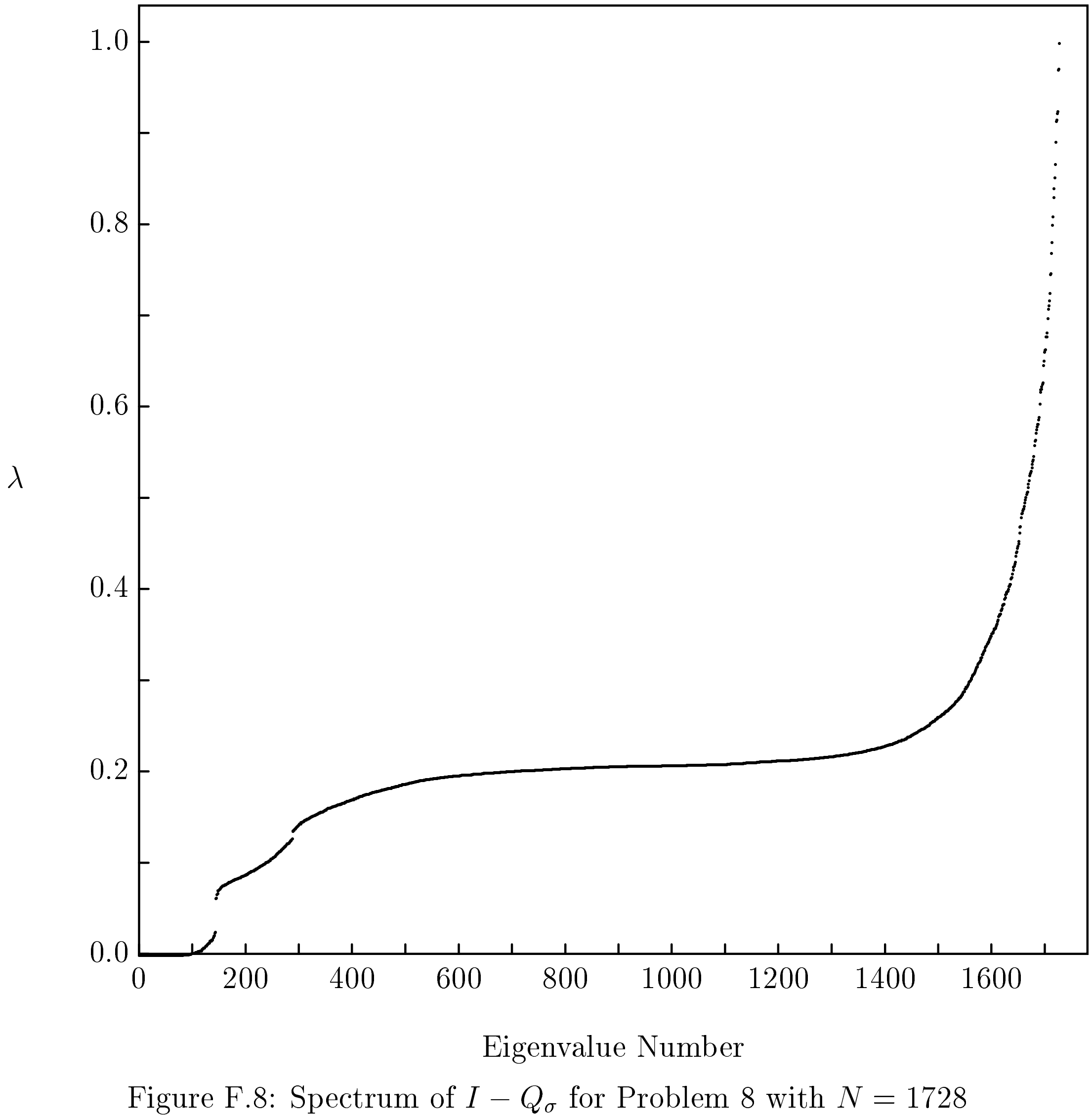


Figure F.8: Spectrum of $I - Q_\sigma$ for Problem 8 with $N = 1728$

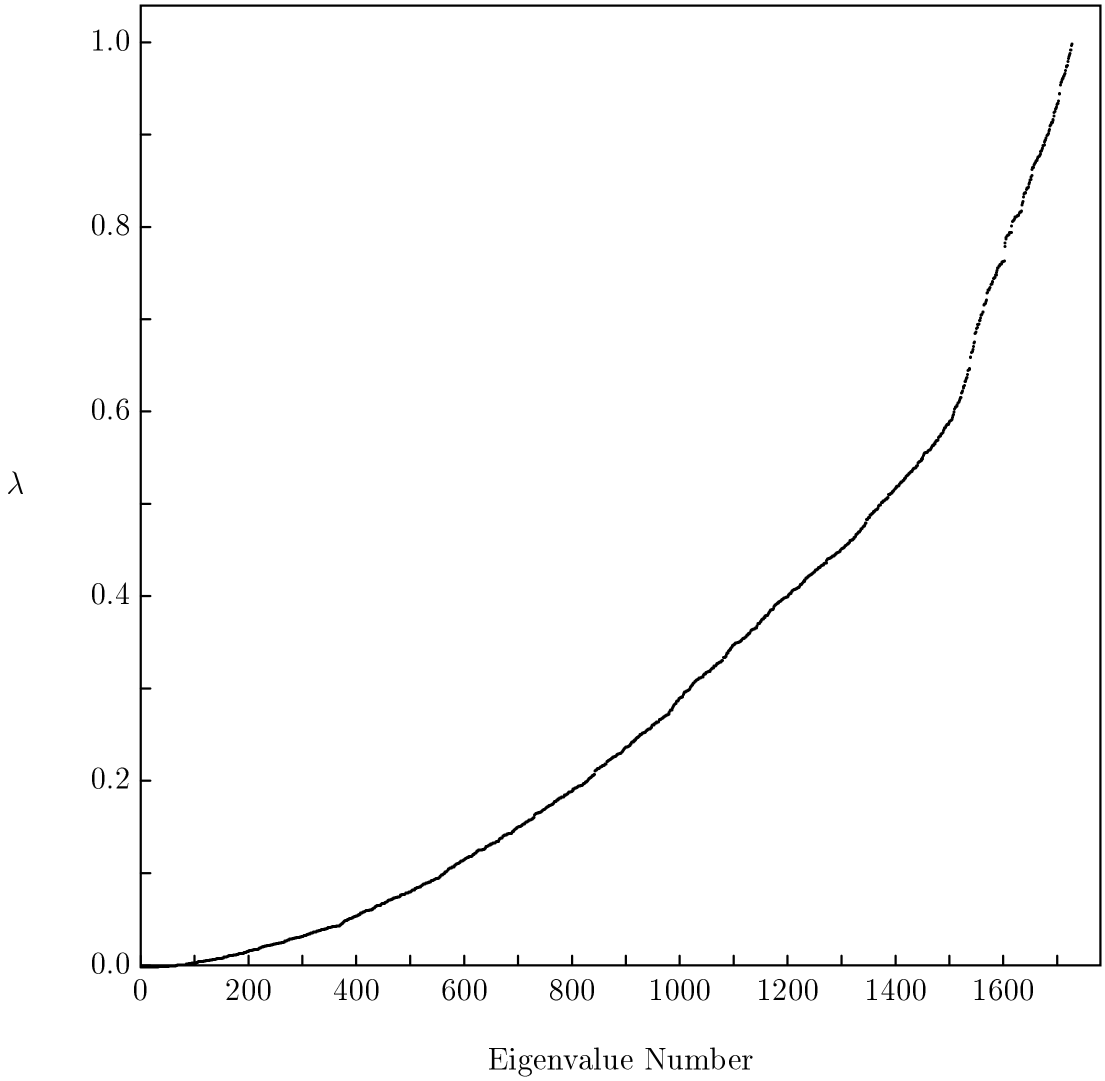


Figure F.9: Spectrum of $I - Q_\sigma$ for Problem 9 with $N = 1728$

Appendix G

Proof of Bound for Block Stiefel Projection Step

This appendix contains the proof to Theorem 5.2. All of the quantities are as defined in Section 5.2.2.

Proof: From the expression for $\bar{z}^{(i)}$ in Lemma 5.26,

$$\left|\frac{\bar{z}^{(i)}}{z_0^{(i)}} - \frac{\bar{z}^{(j)}}{z_0^{(j)}}\right| \leq |p_k(\lambda_i) - p_k(\lambda_j)| + \parallel w \parallel \; \parallel \lambda_i u - \lambda_j y \parallel \tag{G.1}$$

where

$$\begin{aligned} u &= [p_{k-s+1}(\lambda_i), \ldots, p_k(\lambda_i)] \in \Re^{1\times s} \\ y &= [p_{k-s+1}(\lambda_j), \ldots, p_k(\lambda_j)] \in \Re^{1\times s} \end{aligned} \tag{G.2}$$

Similar to the definition $\theta = \delta/\gamma$, define

$$\begin{aligned} \theta_i &= (\delta - \lambda_i)/\gamma = (1 + \nu - 2\lambda_i)/(1-\nu) \\ \theta_j &= (\delta - \lambda_j)/\gamma = (1 + \nu - 2\lambda_j)/(1-\nu) \end{aligned} \tag{G.3}$$

Then

$$\begin{aligned}(\theta_i+\sqrt{\theta_i^2-1})(\theta_j+\sqrt{\theta_j^2-1}) &= (\sqrt{\nu-\lambda_i}+\sqrt{1-\nu})^2(\sqrt{\nu-\lambda_j}+\sqrt{1-\nu})^2/(1-\nu)^2\\ &\le (\sqrt{\nu}+1)^4/(1-\nu)^2\\ &= \left[(1+\sqrt{\nu})/(1-\sqrt{\nu})\right]^2\end{aligned} \tag{G.4}$$

Since $\lambda_i,\ \lambda_j<\nu$, from the definition of p_k and the corresponding Chebyshev polynomial T_k,

$$\begin{aligned}p_k(\lambda_i)-p_k(\lambda_j) &= \left[\cosh(k\cosh^{-1}\theta_i)-\cosh(k\cosh^{-1}\theta_j)\right]/T_k(\theta)\\ &= 2\sinh\tfrac{k}{2}(\cosh^{-1}\theta_i+\cosh^{-1}\theta_j)\sinh\tfrac{k}{2}(\cosh^{-1}\theta_i-\cosh^{-1}\theta_j)\\ &= 2\sinh\left(\ln[(\theta_i+\sqrt{\theta_i^2-1})(\theta_j+\sqrt{\theta_j^2-1})]^{k/2}\right)\\ &\quad *\sinh\left(\ln\left[\tfrac{\theta_i+\sqrt{\theta_i^2-1}}{\theta_j+\sqrt{\theta_j^2-1}}\right]^{k/2}\right)/T_k(\theta)\end{aligned} \tag{G.5}$$

Since $\sinh\xi=(e^{\xi}-e^{-\xi})/2\le e^{\xi}/2$, from (G.4)

$$2\sinh\left(\ln[(\theta_i+\sqrt{\theta_i^2-1})(\theta_j+\sqrt{\theta_j^2-1})]^{k/2}\right)\le\left[(1+\sqrt{\nu})/(1-\sqrt{\nu})\right]^k \tag{G.6}$$

From the definition (G.3) it also follows that

$$\frac{\theta_i+\sqrt{\theta_i^2-1}}{\theta_j+\sqrt{\theta_j^2-1}}=\left[1+\frac{(\sqrt{1-\lambda_i}-\sqrt{1-\lambda_j})+(\sqrt{\nu-\lambda_i}-\sqrt{\nu-\lambda_j})}{(\sqrt{\nu-\lambda_i}+\sqrt{\nu-\lambda_j})}\right]^2 \tag{G.7}$$

Because $\lambda_i\le\lambda_j<\nu$, $\lambda_j-\lambda_i<\epsilon$, and $\sqrt{1-\lambda_i}$, $\sqrt{1-\lambda_j}\ge\sqrt{1-\nu}$,

$$\begin{aligned}\left|\sqrt{1-\lambda_i}-\sqrt{1-\lambda_j}\right| &\le |\lambda_i-\lambda_j|/2\sqrt{1-\nu}\le\epsilon/2\sqrt{1-\nu}\\ \left|\sqrt{\nu-\lambda_i}-\sqrt{\nu-\lambda_j}\right| &\le |\lambda_i-\lambda_j|/2\sqrt{\nu-\lambda_j}\le\epsilon/2\sqrt{\nu-\lambda_j}\end{aligned}\ . \tag{G.8}$$

Hence (G.7) can be bounded by

$$\begin{aligned}\tfrac{\theta_i+\sqrt{\theta_i^2-1}}{\theta_j+\sqrt{\theta_j^2-1}} &\le \left[1+\epsilon/2\tfrac{\sqrt{\nu-\lambda_j}+\sqrt{1-\nu}}{(1-\nu)(\nu-\lambda_j)}\right]^2\\ &= (1+\epsilon\phi)^2\end{aligned}\ . \tag{G.9}$$

Furthermore,

$$
\begin{aligned}
T_k(\theta) &= \cosh(k \cosh^{-1}(\theta)) \\
&= \cosh\left(k \ln\left(\frac{1+\sqrt{\nu}}{1-\sqrt{\nu}}\right)\right) \\
&\geq \left(\frac{1+\sqrt{\nu}}{1-\sqrt{\nu}}\right)^k / 2
\end{aligned}
\tag{G.10}
$$

Putting (G.6), (G.9), and (G.10) into (G.5) gives

$$
\begin{aligned}
|p_k(\lambda_i) - p_k(\lambda_j)| &\leq 2\sinh(\ln(1+\epsilon\phi)^k) \\
&\leq 2\sinh(k\epsilon\phi)
\end{aligned}
\tag{G.11}
$$

where the expansion $\ln(1+\xi) = \xi - \xi^2/2 + \xi^3/3 - \cdots \leq \xi$ has been used on $\xi = \epsilon\phi < 1$. Using Taylor series, it can be shown that for $\xi > 0$

$$
\sinh\xi \leq \xi \cosh\xi. \tag{G.12}
$$

Hence

$$
|p_k(\lambda_i) - p_k(\lambda_j)| \leq 2k\epsilon\phi\cosh(k\epsilon\phi) \tag{G.13}
$$

Applying the same argument that gives (G.13) separately to each component of $u - y$ shows that the l^{th} component of $u - y$ is bounded in absolute value by

$$
2(k-l+1)\epsilon\phi\cosh((k-l+1)\epsilon\phi) \leq 2k\epsilon\phi\cosh(k\epsilon\phi) \tag{G.14}
$$

Since u and y are vectors of length s, the relationship between the infinity norm and the two norm gives

$$
\| u - y \| \leq \sqrt{s} 2k\epsilon\phi\cosh(k\epsilon\phi) \tag{G.15}
$$

and so

$$
\begin{aligned}
\| \lambda_i u - \lambda_j y \| &\leq \| \lambda_i u - \lambda_j u \| + \| \lambda_j u - \lambda_j y \| \\
&\leq \epsilon \| u \| + \lambda_j \sqrt{s} 2k\epsilon\phi\cosh(k\epsilon\phi) \\
&\leq \epsilon\sqrt{s} + \lambda_j \sqrt{s} 2k\epsilon\phi\cosh(k\epsilon\phi)
\end{aligned}
\tag{G.16}
$$

Putting (G.13) and (G.16) into (G.1) gives

$$
\begin{aligned}
\left| \frac{\bar{z}^{(i)}}{z_0^{(i)}} - \frac{\bar{z}^{(j)}}{z_0^{(j)}} \right| &\leq \epsilon \left[2k\phi\cosh(k\epsilon\phi) + \| w \| \sqrt{s}(1 + 2k\phi\cosh(k\epsilon\phi)) \right] \\
&\leq \epsilon(1 + 2k\phi\cosh(k\epsilon\phi))(1 + \sqrt{s} \| w \|)
\end{aligned}
\tag{G.17}
$$

which is precisely the bound in Theorem 5.2. □

Some notes should be made about the bound above:

1. ϕ can be arbitrarily large since $\nu - \lambda_j$ can be small. For the application of interest here, however, $\lambda_j << \nu$.

2. The upper bound is smaller for λ_i and λ_j closer to zero and further away from ν.

3. This suggests that it is better to start with a large initial ν and then decrease it if necessary, rather than start with a small ν and then increase it.

Bibliography

[1] S. Afriat, Orthogonal and oblique projectors and the characteristics of pairs of vector spaces, *Proc. Camb. Phil. Soc.*, 800-816, 53(1957).

[2] R. Ansorge, Connections between the Cimmino-methods and the Kaczmarz-methods for the solution of singular and regular systems of equations, *Computing*, 367-375, 33(1984).

[3] A. Ben-Israel, T. Greville, *Generalized Inverses: Theory and Applications*, Wiley-Interscience, New York, (1974).

[4] A. Berman, R. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, (1979).

[5] G. Birkhhoff, R. Lynch, *Numerical Solution of Elliptic Problems*, SIAM, Philadelphia, (1984).

[6] A. Björck, T. Elfving, Accelerated projection methods for computing pseudo-inverse solutions of systems of linear equations, *BIT*, 145-163, 19(1979).

[7] A. Björck, G. Golub, Numerical methods for computing angles between linear subspaces, *Math. Comp.* 579-594, 27 (1973).

[8] J. Bollen, *Round-off error analysis of descent methods for solving linear equations*, Ph.D Thesis, Technische Hogeschool Eindhoven, (1980).

[9] R. Bramley, A. Sameh, A robust parallel solver for block tridiagonal systems, *CSRD Tech. Rept. 806*, (1988).

[10] Y. Censor, Row-action methods for huge and sparse systems and their applications, *SIAM Rev.*, 444-466 23(1981).

[11] Y. Censor, P. Eggermont, D. Gordon, Strong underrelaxation in Kaczmarz's method for inconsistent systems, *Numer. Math.*, 83-92, 41(1983).

[12] G. Cimmino, Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari, *Ric. Sci. Progr. tecn. econom. naz.*, 326-333, 9(1939).

[13] P. Concus, G. Golub, A generalized conjugate gradient method for nonsymmetric systems of linear equations, *Lecture Notes in Economics and Mathematical Systems: Computing Methods in Applied Sciences and Engineering*, 56-65, Berlin 134(1976).

[14] F. Deutsch, Rate of convergence of method of alternating projections, *Parametric Optimization and Approximations*, ed. Brosowsfi and Deutsch, Birkhäuser Basel: ISNM 72, 96-107 (1985).

[15] J. Dongarra, C. Moler, J. Bunch, G. Stewart, *Linpack User's Guide*, SIAM, Philadelphia, (1979).

[16] P. Eggermont, G. Herman, A. Lent, Iterative algorithms for large partitioned linear systems, with applications to image reconstruction, *Lin. Alg. Appl.*, 37-67, 40(1981).

[17] T. Elfving, Group-iterative methods for consistent and inconsistent linear equations, *Rept. LITH-MAT-R-1977-11*, Dept. Math. Linköping Univ., (1977).

[18] T. Elfving, Block iterative methods for consistent and inconsistent linear equations, *Numer. Math.*, 1-12, 35(1980).

[19] H. Elman, G. Golub, Iterative methods for cyclically reduced non-self-adjoint linear systems, *UMIACS-TR-88-87, CS-TR-2145*, University of Maryland, (1988).

[20] H. Elman, Iterative methods for large, sparse, nonsymmetric systems of linear equations, *Yale Univ. Res. Rept. 229*, (1982).

[21] D. Evans, Parallel algorithms in computational linear algebra, *Parallel Computers and Computations*, 55-79, ed. J. van Leeuwen, J. Lenstra, Centrum voor Wiskunde en Informatica (1985).

[22] R. Fletcher, Conjugate gradient methods for indefinite systems, *Numerical Analysis Dundee 1975*, 73-89, ed. G. Watson, Springer-Verlag, New York, (1976).

[23] J. George, J. Liu, *Computer Solution of Large Sparse Positive Definite Systems* Prentice-hall, Engelwood Cliffs, N.J. (1981).

[24] P. Gilbert, Iterative methods for the three-dimensional reconstruction of an object from projections, *J. Theor. Biol.*, 105-117, 36(1972).

[25] G. Golub, M. Overton, Convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems, *Numer. Math.*, 571-593, 53(1988).

[26] G. Golub, C. Van Loan, *Matrix Computations*, John Hopkins University Press, Baltimore (1983).

[27] R. Gordon, R. Bender, G. Herman, Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and x-ray photography, *J. Theoret. Biol.,* 471-481 29(1970).

[28] I. Gustafsson, Modified incomplete Cholesky methods, *Preconditioning Methods: Theory and Applications*, ed. D. Evans, 265-293, Gordon and Breach, New York, (1983).

[29] I. Halperin, The product of projection operators, *Acta Sci. Math. (Szeged)*, 96-99 23(1962).

[30] M. Hanke, W. Niethammer, On the acceleration of Kaczmarz's method for inconsistent linear systems, *Institut für Praktische Mathematik, Universität Karlsruhe*, (1988).

[31] L. Hageman, D. Young, *Applied Iterative Methods*, Academic Press, New York, (1981).

[32] G. Herman, A. Lent, P. Lutz, Relaxation methods for image reconstruction, *Comm. ACM*, 152-158, 21(2), (1978).

[33] G. Herman, *Image Reconstruction from Projections*, Academic Press, New York (1980).

[34] M. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Research Nat. Bur. Standards*, Paper 2379, #6 49(1952).

[35] D. Jespersen, P. Buning, Accelerating an iterative process by explicit annihilation, *SIAM J. Sci. Stat. Comp.*, 639-651, 6(1985).

[36] C. Johnson, A Gersgorin-type lower bound for the smallest singular value, *Linear Algebra and its Applications*, 1-7, 112 (1989).

[37] S. Kaczmarz, Angenäherte Auflösung von Systemen linearer Gleichungen, *Bull. intern. Acad. polonaise Sci. lettres (Cracouie); Class sci. math. natur.: Seira A. Sci. Math.*, 355-357 (1939).

[38] C. Kamath, Solution of nonsymmetric systems of equations on a multiprocessor, *CSRD Rept. 591*, (1986).

[39] C. Kamath, A. Sameh, A projection method for solving nonsymmetric linear systems on multiprocessors, *Parallel Computing*, 291-312, 9(1988/1989).

[40] D. Kincaid, D. Young, Adapting iterative algorithms developed for symmetric systems to nonsymmetric systems, *Elliptic Problem Solvers*, 353-359, ed. M. Schultz, Academic Press, New York, (1981).

[41] A. Kydes, R. Tewarson, An iterative method for solving partitioned linear equations, *Computing* 357-363, 15(1975).

[42] A. Lakshminarayanan, A. Lent, Methods of least squares and SIRT in reconstruction, *J. Theor. Biol.*, 267-295, 76(1979).

[43] T. Manteuffel, The Tchebychev iteration for nonsymmetric linear systems, Numer. Math., 307-327, 28(1977).

[44] T. Manteuffel, Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration, *Numer. Math.*, 183-208, 31(1978).

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

[46] S. Nelson, M. Neumann, Generalizations of the projection method with applications to SOR theory for Hermitian positive semidefinite linear systems, *Numer. Math.* 123-141, 51 (1987).

[47] W. Peters, Lösung linearer Gleichungssysteme durch Projektion auf Schnitträume von Hyperebenen und Berechnung einer verallgemeinerten Inversen, *Beit. Numer. Math.*, 129-146, 5(1976).

[48] J. Reid, On the use of conjugate gradients for the solution of large sparse systems of linear equations, *Large Sparse Sets of Linear Equations*, ed. J. Reid, 231-254, Academic Press (1971).

[49] Y. Saad, A. Sameh, P. Saylor, Solving elliptic differential equations on a linear array of processors, *SIAM J. Sci. Stat. Comp.*, 1049-1063, 6(1985).

[50] Y. Saad, M. Schultz, Conjugate gradient-like algorithms for solving nonsymmetric linear systems, *Math. Comp*, 417-424, 44(1985).

[51] P. Saylor, Private communication (1988).

[52] Scientific Computing Associates, Inc., *PCGPAK User's Guide*, New Haven, CT (1987).

[53] A. Sherman, An empirical investigation of methods for nonsymmetric linear systems, *Elliptic Problem Solvers*, 429-434, ed. M. Schultz, Academic Press, New York, (1981).

[54] G. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Clarendon Press, Oxford, (1978).

[55] G. Stewart, Introduction to Matrix Computations, Academic Press, New York, (1973).

[56] G. Stewart, On the perturbation of pseudo-inverses, projections, and linear least squares problems, *SIAM Rev.*, 634-662, 19(1977).

[57] K. Tanabe, A projection method for solving a singular system of linear equations, *Numer. Math.*, 203-214 17(1971).

[58] A. Trummer, Reconstructing pictures from projections: on the convergence of the ART algorithm with relaxation, *Computing*, 189-195, 26(1981).

[59] A. Trummer, A note on the ART of relaxation, *Computing*, 349-352, 33(1984).

[60] P. Vinsome, Orthomin, an iterative method for solving sparse sets of simultaneous linear equations, *Proc. Fourth Symp. on Reservoir Simulation*, 149-159, Society of Petr. Eng. of AIME, (1976).

[61] R. Wainwright, R. Keller, Algorithms for projection methods for solving linear systems of equations, *Comp. Math. Appl.*, 235-245, 3(1977).

[62] T. Whitney, R. Meany, Two algorithms related to the method of steepest descent, *SIAM J. Numer. Anal.*, 109-118, 4(1967).

[63] J. Wilkinson, Modern error analysis, *SIAM Rev.*, 548-568, 13(1971).