INCOMPLETE FACTORIZATION PRECONDITIONING FOR LINEAR LEAST SQUARES PROBLEMS

BY

XIAOGE WANG

B.Sc., Tsinghua University, 1982
M.Sc., Tsinghua University, 1984

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1994

Urbana, Illinois

INCOMPLETE FACTORIZATION PRECONDITIONING
FOR LINEAR LEAST SQUARES PROBLEMS

XIAOGE WANG, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1994
K. A. Gallivan , Advisor

A new family of preconditioners for conjugate gradient-like iterative methods applied to large sparse linear least squares problems, $min \parallel Ax - b \parallel_2$, is proposed. The family is based on incomplete Gram-Schmidt (IGS) factorizations of A. Particular attention has been given to the following members of the family: Incomplete Classical Gram-Schmidt (ICGS), Incomplete Modified Gram-Schmidt (IMGS) and Compressed Incomplete Modified Gram-Schmidt (CIMGS) factorizations. The numerical properties of each of these methods have been considered as well as the relationships between the methods concerning the preservation of sparsity, computational efficiency and the quality of the preconditioner. The implementation of these methods has been investigated and all of the important family members have been coded. One of the important topics in this portion of the dissertation is the careful symbolic analysis of the production of the preconditioner and its use during the incomplete factorization phase to avoid excessive unnecessary work.

We have also analyzed some of the implications of viewing the incomplete Cholesky (IC) factorization as a limit case of the IGS family. When certain conditions are imposed on the sparsity pattern of the preconditioning matrix, IC is identical to IMGS. Therefore, a sufficient condition on the sparsity pattern for a stable IC factorization can be derived. Based on this condition, two algorithms for modifying a sparsity pattern for which IC may not succeed have been designed. The altered IC method with the pattern modifications (ICPM) has been studied and its performance evaluated. These methods differ from previously proposed methods to guarantee the existence and improve the stability of the IC factorization since they do not require numerical information. Of course, for a particular matrix we can then use numerical information in addition to the pattern modification techniques to further improve the preconditioner.

Numerical experiments illustrating the capabilities of the preconditioners are also presented. These include matrices from the standard Harwell-Boeing collection and various test matrices from different practical application problems.

TO MY MOTHER

ACKNOWLEDGMENTS

I would like to express my gratitude to all the people who made this dissertation possible.

First, I thank Dr. Ahmed Sameh for inviting me to the CSRD as a visiting scholar and introducing me to the field of scientific computing. I thank him for understanding and support through the several difficult situations. Without his encouragement and help, I would not have finished this work.

My special thanks to Professor Kyle Gallivan, my thesis advisor, for his guidance, encouragement, support, understand and patience throughout my thesis work. Every discussion with him is helpful and enjoyable. I also greatly appreciate his numerous suggestions that improved significantly not only the clarity of my thought but also the quality of this dissertation.

I thank other members of my thesis committee: Randall Bramley, Michael Heath, Paul Saylor and Robert Skeel, for their interest in the topic and helpful comments.

I am grateful to Professor Randall Bramley for his friendship since the beginning of my student life in Urbana. His understanding, patient, considerate, support and also his sense of humor made the life of being a graduate student in a foreign country much less difficult.

My thanks to my friends Meiqing Chen, Wei Li, Rong Chen, Infan Cheong, Hoichi Cheong, Joan Huang, Jimmy Hsia, Bilong Liao, Lixia Liu, Xiaohui Song, and many others for their support and friendship during these years in Urbana. I also thank my officemates for their help and company during my graduate study.

Last but not the least, I thank my family for their love and support that are always there without exception.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 The linear least squares problem and its applications

The target problem of this thesis is the linear least squares problem which can be stated as: Let $A \in \Re^{m\times n}$, $m \geq n$, be a large sparse matrix with full rank and let $b \in \Re^m$ be an arbitrary vector. Find $x \in \Re^n$ which minimizes the value of

$$\| b - Ax \|_2 \ . \tag{1.1}$$

Such problems occur frequently in scientific and engineering applications such as linear programming [17], augmented Lagrangian methods for CFD [51], and the natural factor method in partial differential equations [41] [6].

One example of an application of the linear least squares problems is seismic travel tomography [5]. In this application, the linear least squares problem arises from the solution of nonlinear inverse problems associated with the approximation of acoustic or elastic wavespeed from travel time. The relation between the travel times (data) and the wavespeed (model parameters) is of the form

$$t(r) = \int_{r(s)} s(x,y,z)dl, \tag{1.2}$$

where x, y and z are coordinates, dl is the distance along the ray r and $s(x,y,z) = 1/\mu(x,y,z)$ is the slowness at the point (x,y,z). The ray path depends on the slowness (unknown) and thus 1.2 must be linearized about some initial or reference slowness model. Discretizing the slowness by cell, the resulting overdetermined system of linear equations for the unknown slowness perturbation values is

$$D\Delta s = \Delta t, \tag{1.3}$$

where D is the Jacobian matrix whose (i,j) element is the distance the i-th array travels in the j-th cell (element), the components of Δs are the difference between the initial and updated model, and the components of Δt are the difference between the travel times computed for the model and observed times. The system 1.3 is in general large and sparse.

Another example is the computation involved in geodetic adjustment and surveys [36]. The computations are based on a mathematical model called a geodetic position network. It consists of several mesh points or stations, with unknown positions over a reference surface or 3-dimensional space. The points are connected by lines, each representing the observation involving these two points. These observations then lead to a system of overdetermined nonlinear equations

$$F(p) = q. \tag{1.4}$$

Typically, each equation only involves a small number of unknowns. The Gauss-Newton method is used to solve the nonlinear system. This requires the solution of a sequence of linear least squares problems of the form 1.1, where A is the Jacobian of F at the current vector of unknown coordinates p_0 and $b = q - F(p_0)$. The matrix A here is, in general, very large and sparse and may have a special structure which can be exploited to improve the efficiency of the solution.

1.2 A summary of the thesis

The objective of the thesis is to find, describe, and compare efficient and effective preconditioning methods for conjugate gradient-like (CG-like) iterative methods for solving linear least squares problems. In particular, the thesis generalizes two incomplete orthogonalization preconditioners, proves theorems assuring their existence, provides conditions for the existence of incomplete Cholesky (IC) factors based only on sparsity patterns, develops three new pattern modification algorithms that guarantee the existence of an IC factorization, proposes and analyzes a new incomplete orthogonalization preconditioner, establishes relationships between all of the preconditioners examined, and presents numerical testing results.

Following a review of previous work in the areas, which is presented in Chapter 2, the first part of the main contribution of the thesis is presented in Chapters 3 and 4. Two basic incomplete orthogonal factorization methods, incomplete classical Gram-Schmidt (ICGS) and incomplete modified Gram-Schmidt (IMGS) are introduced. Combining these with a variety of sparsity pattern selection strategies and matrix ordering methods, we create a family of incomplete orthogonalization precondition methods that includes those proposed by Jennings and Ajiz and by Saad.

The numerical properties of these methods are studied and several theoretical results presented in Chapter 3. They include the existence of the IMGS and the ICGS factorization for a full rank matrix, an estimate of the deviation of IMGS from an orthogonal factorization for matrices A whose normal equations coefficient matrix A^TA is an M-matrix, the analysis of the error caused by inexact arithmetic for IMGS and the equivalence of IMGS and the incomplete Cholesky factorization under certain conditions on the sparsity pattern.

In Chapter 4, the results of experiments investigating the performance of IMGS and ICGS as preconditioners for CG applied to the normal equations without explicitly forming A^TA (CGLS) are presented.

It is demonstrated that this family of preconditioners is more robust than IC, since the factorization does not suffer numerical breakdown if the matrix A has full rank. Given the guarantee of existence, we then explore improving the performance by means of sparsity pattern selection and ordering of the columns of A. The results show that the upper triangular matrix R generated by IMGS or ICGS with the various pattern selection strategies and the ordering methods appears to be effective in accelerating the convergence of CGLS. Chapter 4 also contains an explanation of the fact that for a given sparsity pattern, the factor R produced by IMGS typically performs better than the R produced by ICGS in the sense that CGLS needs fewer iterations to converge to a given accuracy.

The study of IMGS and ICGS in the first part of the thesis extends the understanding of the use of incomplete orthogonal factorization preconditioning contained in the literature. Chapters 5 and 6 pursue the implications of two key results from the earlier chapters. The first part of the thesis demonstrates the effectiveness of the preconditioners computed by the IMGS and ICGS and also identifies the major problem with the methods. The cost, in terms of both storage and operation count, is generally much higher for IMGS and ICGS than for other preconditioners such as IC. Chapter 5 addresses this drawback by investigating an improved version of one particular member of the preconditioner family. Properties and implementation considerations of the new algorithm Compressed IMGS (CIMGS) are discussed. The equivalence of the IMGS and the IC preconditioners in exact arithmetic under certain conditions is also demonstrated in the first half of the thesis. These conditions assure the existence of IC based only on the sparsity pattern of R. Chapter 6 pursues the implications of this result with respect to improving the robustness of IC preconditioners.

CIMGS generates the same upper triangular matrix as IMGS. Therefore, in exact arithmetic it preserves the advantages of the IMGS and reduces the computation cost substantially. An error analysis of CIMGS shows that the error introduced by inexact arithmetic is related to the condition of the normal equations and the sparsity pattern of R. The monotone property of CIMGS is also demonstrated in Chapter 5. This property says that if a matrix is a symmetric positive definite M-matrix, two zero position sets P_1 and P_2 satisfy $P_1 \subset P_2$, and R_i is the factor produced by CIMGS using P_i, then $R_1 \leq R_2$ elementwise.

Different sparsity pattern selection strategies have been tested for CIMGS, and in addition, the ordering schemes tested for IMGS have also been tested for CIMGS. The results show that the preconditioner produced by CIMGS is usually as effective as that produced by IMGS and that the difference in the R due to finite precision is usually not large enough to affect the number of iterations required by CGLS preconditioned by the two different R's. Somewhat unexpected, however, is the fact that for a small set of problems, CGLS converges faster when preconditioned by CIMGS than when preconditioned by IMGS. The fact that the computational cost of CIMGS is much less than that of IMGS makes CIMGS a method with great potential.

The quality of the CIMGS (and IMGS) preconditioning strongly depends on the selection of the sparsity patterns. If the sparsity pattern is chosen properly, the performance of CGLS preconditioned by CIMGS is comparable in performance to using the normal equations which is a very efficient direct method for reasonably well conditioned problems. In general, of course, finding an optimal pattern is difficult. However, for applications such that the pattern can be reused for many time steps, the cost of finding an effective pattern is amortized.

It has also been discovered that certain computations in CIMGS with a statically determined pattern are unnecessary and can be identified and eliminated by symbolic analysis. Such a symbolic analysis algorithm has been developed and its effectiveness tested. The results reveal that for many of the test problems, the amount of unnecessary computation is small (a fact that could not be known until the symbolic analysis algorithm existed). Therefore, with the help of matrix reorderings which reduce fill-in for Cholesky factorization, the unnecessary computation can be kept small. Of course, the algorithm will be useful if, in the future, problems with larger amounts of unnecessary computations are encountered.

In Chapter 6 a new way of assuring the existence of the IC factorization is proposed. As mentioned above, it is proven in Chapter 3 that the IC factorization of a symmetric positive definite matrix B with the sparsity pattern P having the certain property (Property C) always exists. This is proven again in Chapter 6 from a different point of view. Three new algorithms, MPADD, MPDROP and MPBBD, that assure the condition for the existence of the IC factorization are introduced. The method MPADD forces the pattern to satisfy the property C by adding elements to the set of the non-zero positions of R, while the method MPDROP eliminates elements from the set of non-zero positions to guarantee existence. It is also proven that if a matrix has a nested bordered block diagonal structure, applying the modification of either MPADD or MPDROP does not destroy the structure. The method MPBBD is designed based on this result. The results of numerical testing of the methods are reported and discussed.

Finally, in Chapter 7, a theoretical comparison is made of IMGS and IC applied using the same sparsity pattern. It is shown that for those problems whose normal equations are M-matrices, IMGS (and hence CIMGS) provides a better approximation of the Cholesky factorization of the normal equations $A^T A$ than IC. The performance of the preconditioning methods, QR factorization, and the normal equations to solve the least squares problem is also compared experimentally in detail in Chapter 7.

CHAPTER 2

Review of the previous work

Methods of solving linear least squares problems are reviewed in this chapter. Excellent surveys of work in this area are [22], [40], and [12].

2.1 Normal equations

The normal equation method is described by the following algorithm:

Algorithm:

begin

Compute the lower triangular portion of $C = A^T A$.

$d = A^T b$.

Compute the Cholesky factorization $C = GG^T$.

Solve $Gy = d$ *and* $G^T x_{LS} = y$.

end

The algorithm requires $(m + n/3)n^2$ flops in the dense case. Details of this method can be found in [37] [45]. This is the most widely used method for solving full rank linear least squares problems [37]. It is easy to derive and understand. It relies on standard algorithms: Cholesky factorization, triangular system solver, matrix-matrix multiplication, and matrix-vector multiplication which are well developed in both dense and sparse cases. When $m \gg n$, the normal equations method becomes more attractive because of the compression of m-by-n data A into the much smaller n-by-n matrix C. The disadvantage of the normal equations method is that the accuracy of the computed normal equations solution depends on the square of the condition number of A.

It is natural for some applications to have a few dense rows in A. In such case, $C = A^T A$ becomes a dense matrix. One way to handle the dense rows is to separate them from A, then use the Sherman-Morrison-Woodbury formula for a low rank correction [17]. Another way is to split dense rows and create a larger sparse matrix by adding properly defined coefficients. The solution to the normal equations is determined by solving an enlarged sparse linear system [19] [38].

Generally, the normal equations will have difficulty for a matrix A where A^TA is too large to use the dense matrix techniques or too dense to apply sparse matrix techniques or most often when A is nearly rank deficient.

2.2 Orthogonal factorization

The 2-norm is preserved by orthogonal transformation, that is

$$\| b - Ax \|_2 = \| Qb - QAx \|_2, \tag{2.1}$$

where $Q^TQ = I$. Householder reflectors and Givens rotations are orthogonal transformations that are typically used to reduce A and b to the form

$$QA = \begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad Qb = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

where R is upper triangular matrix of order n, and $b_1 \in \Re^n$. The solution of least squares is the solution of the triangular system $Rx_{ls} = b_1$ and the residual $b - Ax$ is equal to $\begin{pmatrix} 0 \\ b_2 \end{pmatrix}$.

The Gram-Schmidt orthogonalization procedure (and its modified version) can also be used to solve the least squares problem. The method computes an orthogonal basis of A such that $A = \tilde{Q}R$, where $\tilde{Q} \in \Re^{m \times n}$ and $\tilde{Q}^T\tilde{Q} = I$. The solution of least squares problem is the solution of the triangular system $Rx = \tilde{Q}^Tb$.

Error analyses of these methods show that orthogonal factorizations based on Householder reflectors, Givens rotations, and the modified Gram-Schmidt procedure are stable methods for solving the least squares problem in the sense that the computed solution is the exact solution of a nearby problem.

In the sparse case, orthogonal transformations have a serious drawback. They usually need much more space and time than the normal equations method. Q is usually much denser than A. Although Q can be discarded, large storage may still be needed for the computation. Several sparse orthogonal factorization algorithms have been suggested [31] [46] [30] [32] [34] [53] [18] [64] [29] [49].

If multiple solutions have to be computed for the same A and multiple b, where not all b are available at the same time, and if Q is not stored, the seminormal equations method can be used. It computes the orthogonal factorization of A for R and solves

$$R^TRx = A^Tb. \tag{2.2}$$

It would seem that this method should be more stable than the normal equations method because it use an orthogonal factorization to compute R. However, a stability analysis of seminormal equations method by Björck shows that the error of the solution of the seminormal equations method is of the same order as the solution of the normal equations [10]. Under mild conditions, a correction step can be added to yield a solution as accurate as the QR method.

As with the normal equations, if there are a few dense rows in A, severe fill–in occurs in R. Usually, such rows are treated separately by using technique of updating a QR factorization [37][35][29].

2.3 An elimination method

A method based on an LU factorization can be described as follows:

Algorithm:

begin

Compute $P_1 A P_2 \ = \ LU$.

Solve $L^T L y \ = \ L^T P_1 b$.

Solve $U P_2^T x_{LS} \ = \ y$.

end

where $P_1 A P_2 = LU$ is an LU factorization with complete pivoting. P_1, P_2 are permutation matrices.

The method was first proposed by Peters and Wilkinson [57]. Usually, by using an appropriate pivoting strategy, it is possible to control the condition of L, so $L^T L$ is better conditioned than $A^T A$. The numerical stability is not as satisfactory as with the orthogonal factorization methods. From the sparsity point of view, this method is usually better in preserving sparsity than orthogonal factorization methods (it is impossible to do a theoretical comparison because examples can be shown to say that one method is better than the other)[22].

A. Björck and I. Duff later proposed a modification on this method [14] which is adaptable to large sparse problems. Let

$$L = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix}, \qquad P_1 b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

be a partition such that $L_1 \in \Re^{n \times n}$ and $b_1 \in \Re^n$. Let

$$L_1 c = b_1, \qquad d = b_2 - L_2 c,$$

If A is nearly consistent, that is $\| d \|_2$ is very small, then the solution of $L^T L y = L^T P_1 b$ is set to c. If $\| d \|_2$ is not small enough, then solve $L^T L z = L^T \begin{pmatrix} 0 \\ d \end{pmatrix}$, $U P_2^T x1 = c$, $U P_2^T x_2 = z$, and $x = x_1 + x_2$.

The modified method has some advantages. First, it detects the consistent case and treats it effectively. Second, it preserves the sparsity using threshold pivoting strategy. An updating technique is used if there are a few dense rows that could cause severe fill–in in $L^T L$.

2.4 Augmented system method

Let $r = b - Ax$ be the residual of the least squares problem. The solution of 1.1 can be obtained from the solution of the augmented system:

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \quad or \quad \begin{pmatrix} \alpha I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \alpha^{-1} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \tag{2.3}$$

which is the scaled form.

This method was proposed by Bartels et al., and later considered for the sparse case by Hachtel. Björck used it in a study of iterative refinement for least square solutions [7]. Numerical experiments on this method and the comparison with other methods have been done [1] [20].

If the pivots are chosen from the diagonal when Cholesky factorization is applied to the augmented system, then after m steps, the reduced system is exactly the normal equations. Expressing the least squares problems in augmented system form with scaling allows more flexibility in pivoting and avoids the loss of information in forming $A^T A$. The system is symmetric indefinite, so a symmetric pivoting strategy proposed by Bunch and Kaufman [15] could be applied to preserve the symmetry and maintain stability as well. It determines 2×2 block pivots as well as single element pivots. It is shown that this pivoting strategy leads to a stable method in the sense that the element growth is bounded by a factor 2.75^{m+n-1}, which can be compared to the growth factor 2^{m+n-1} of Gaussian elimination with partial pivoting. Bunch and Kaufman's method alone is not in general satisfactory enough in solving least squares problem. A proper scaling shall be also applied. If α is set sufficiently large, then the Bunch-Kaufman method will result in the normal equations after m steps. Using smaller α will introduce 2-by-2 pivots of the form $\begin{pmatrix} \alpha & a_{ir} \\ a_{ir} & 0 \end{pmatrix}$. The choice of α will effect the choice of pivots and the stability. The error analysis by Björck [13] shows that

$$\min_{\alpha} \kappa_2(M_\alpha) = \frac{1}{2} + \sqrt{\frac{1}{2} + 2(\frac{\sigma_1}{\sigma_n})^2} \approx \sqrt{2}\kappa_2(A)$$

is attained for $\tilde{\alpha} = \frac{1}{\sqrt{2}}\sigma_n(A)$, where $\sigma_1 \geq \ldots \geq \sigma_n$ are the singular values of A. Another formula for the optimal value of α is derived by minimizing an upper bound for the round-off error in the solution:

$$\alpha = \begin{cases} \tilde{\alpha}, & \sigma_n \leq \tilde{\alpha} \leq \sigma_1 \\ \sigma_1, & \sigma_1 < \tilde{\alpha} \\ \sigma_n, & \tilde{\alpha} < \sigma_n \end{cases}$$

where $\tilde{\alpha} = (\frac{\sigma_n \|r\|_2}{\|x\|_2})^{1/2}$. The parameter α is usually unknown before the computation. If α is not small enough, the result may not be acceptable. On the other hand, if α is too small, the amount of fill–in will be too large to be accepted. Experimental results are presented in [1].

2.5 Iterative methods

As in solving large sparse linear systems, an iterative method is a very attractive alternative to direct methods for solving large sparse linear least squares problems. There are basically two classes of methods. The first class contains those iterative methods for solving symmetric positive definite systems and include CG. The methods are applied to the normal equations to solve least squares problems. To avoid the explicit form of A^TA, the standard iterative methods use the normal equations in the form of $A^T(b-Ax)=0$, while CG type methods use intermediate vectors for the computation [54] [55] [56] [11]. Although these methods do not explicitly form the normal equations, the rate of convergence still relies on the spectrum of A^TA. Preconditioning techniques that accelerate the convergence of these methods have received extensive attention in the literature. Some examples of the literature of preconditioning symmetric positive definite linear systems and least squares problems are [25] [2] [39] [59] [21] [4] [61] [3] [16].

They can be summarized as follows:

- Column scaling. $C = diag(d_i)$, where d_i are norms of columns of A.
- SSOR preconditioning [9]. $C = I + \omega L^T$, where A has been normalized so that $A^TA = L + I + L^T$ with L strictly lower triangular, and ω is a scalar parameter.
- Incomplete Cholesky factorization. $C = \hat{R}$, where $\hat{R}$ is from approximation of Cholesky factorization of A^TA, and $\hat{R}$ is more sparse than the normal Cholesky factor.
- Approximate orthogonal factorization [58] [43] [27] [66] [42]. $C = \tilde{R}$, where $A = \tilde{Q}\tilde{R} + E$ is an approximation of a QR factorization, $\tilde{Q}$ is an orthogonal matrix and $\tilde{R}$ is upper triangular matrix.

Most preconditioners are adapted from preconditioners for general symmetric positive definite matrices except for the approximation of QR factorizations. The work presented in this thesis will focus on a class of incomplete QR factorization preconditioners.

Another class of methods, such as 2-block SOR and 3-block SOR, are based on splitting the augmented system. The conditions for convergence are discussed in [52] [48].

2.6 Incomplete orthogonal factorization preconditioning

Since our study will focus on a class of approximate orthogonal factorization preconditioning, we give a more detailed review of the previous work on this subject.

2.6.1 Methods of Jennings and Ajizi

Jennings and Ajiz discussed three incomplete factorization methods [43]. Two are incomplete orthogonal factorizations. The incomplete Gram-Schmidt method proposed in this paper is the modified version of Gram-Schmidt orthogonalization with a *rejection test* applied right after the formation of the off-diagonal elements of the factor R. For a given rejection parameter $0 \leq \psi \leq 1$, the rejection test is:

$$\text{if } r_{ij} < \psi / \parallel a_i^{(0)} \parallel_F \;\; \text{then } r_{ij} = 0$$

where $A = [a_1^{(0)}, \ldots, a_i^{(0)}]$. The rejection of small elements in R has the effect of reducing the fill–in in R and therefore in Q. This decomposition can not break down if A is of full rank. This incomplete orthogonalization is discussed in the next chapter as a special case of a generalized incomplete orthogonal factorization algorithm.

The incomplete Givens method proposed in their paper has two drawbacks although the method does not suffer from the possibility of breakdown. The method rejects an element if its magnitude is smaller than a given rejection parameter. By rejecting the element the method shifts the element into a new row – the element is not dropped or ignored. It must be eliminated by a Givens rotation later. In this way, the total computation may not be reduced. The other drawback is that, as the authors indicated, the splitting may not improve the condition of the matrix. It is therefore not clear if this approach can be used as a basis for an effective preconditioning strategy.

2.6.2 ILQ method of Saad

The work in [58] was motivated by the need to solve indefinite nonsymmetric linear systems. The ILQ preconditioner is combined with the normal equations to solve a nonsymmetric linear system. Obviously, this approach can be applied to solve linear least squares problems. ILQ can be described as follows:

Algorithm (ILQ)

begin

for $i = 1, 2, \ldots n$

step 1: **if** $i = 1$ **then** $\hat{q}_1 = a_1$, *goto step 5* **endif**

step 2: Compute all nonzero inner products $l_{ij} = q_j a_i^T, j = 1, 2, \ldots, i-1$

step 3: Determine the P_L *largest* l_{ij}*'s* $j = 1, 2, \ldots, i-1$ *and assign a zero value to the others*

step 4: Compute $\hat{q}_i = a_i - \sum_{j=1,\, l_{ij} \neq 0}^{j=i-1} l_{ij} q_j$

step 5: Determine the P_Q *largest elements of* $\hat{q}_i$ *and assign a zero value to the other elements*

step 6: Compute $l_{ii} = \parallel \hat{q}_i \parallel_2$ *and* $q_i = \hat{q}_1 / i_{ii}$

endfor

end

Note that the algorithm is equal to the classical Gram-Schmidt orthogonalization procedure if steps 3 and 5 are removed. It is these steps that introduce the incompleteness into the factorization. At steps 3 and 5, the algorithm assigns zero values to some elements of Q and L. This makes the matrix L and Q sparse and reduces the amount of computation to less than that of the classical Gram-Schmidt procedure. Because of the zero value assignment to the elements of Q, there is a possibility of breakdown of this incomplete orthogonalization procedure due to loss of full rank.

The performance of this approach of solving the linear systems that arise from partial differential equations is evaluated and reported by Saad. The performance for the two test problems shows the superiority of this approach compared to four other standard approaches. This method is also a member of the incomplete orthogonalization family analyzed in this thesis.

2.6.3 Method of Zlatev and Nielson

Zlatev and Nielson [66] proposed a method for solving least squares problems by preconditioned CG. Their algorithm first computes the incomplete orthogonal factorization $A = QDR + E$ using the Gentleman version [28] of Givens rotation with a numerical dropping technique which has been apply in the Y12M sparse linear system solver [65]. DR is used as a preconditioner for CG applied to the normal equations.

In this incomplete orthogonalization, E is from the contribution of the dropped elements whose magnitude are smaller than a given positive number ϵ called the dropping tolerance. From the test results reported in the paper, if ϵ is small, the perturbation E will be small so a few steps of iterative refinement will be better than applying CG. The computation cost is then very close to direct methods (both the space and time). When ϵ is larger, the space and number of computations for the factorization are reduced considerably and preconditioned CG will perform better. If ϵ is too large, then the algorithm may breakdown due to a loss of full rank. Or the preconditioned CG iteration may converge too slowly due a poor preconditioner. In this case, the algorithm adjusts the value of ϵ and restarts the computation.

For the test problems used in the paper, if pure CG can converge fast, then it is the best compared to the preconditioned CG and ORTHOMIN without a preconditioner. For most problems pure CG method does not converge. In this case, either preconditioned CG or ORTHOMIN will be best. For the problems that preconditioned CG loses to ORTHOMIN, the reasons for the loss are either because the starting value of dropping tolerance is not small enough so that PCG failed to converge, and it has to start over using an adjusted dropping tolerance, or because PCG takes too many iterations to converge which is also related to an improper choice of the drop tolerance. It is pointed out specifically by the authors that ' The best result, obtained with an optimal value of the drop-tolerance will nearly always be better than the results obtained with the "pure" orthogonalization method. However, it may sometimes be difficult to find such an optimal value'. The difficulty of finding the optimal value of ϵ also appears in that the number of elements that will be kept in R are not know a priori for a given ϵ.

2.6.4 Method of James

James also discusses incomplete QR factorization [42]. The method is an incomplete QR factorization using Givens rotation with positional dropping. The set of non-zero positions kept in R is predetermined using the pattern of normal equations. The pivot row is kept sparse by dropping the elements in the zero positions. All the fill-in in target rows are kept. The following statements are made in [42] to summarize the reported experiments:

- Incomplete QR factorization can breakdown even if $A^T A$ is diagonally dominant or an M-matrix.
- Row permutation may or may not prevent breakdown.
- For the test problems from H/B collection, the quality of the IQR preconditioner was marginal at best.

Based on these comments, extensions of this approach will not be pursued in this thesis.

CHAPTER 3

Incomplete Gram-Schmidt Orthogonalization

There are two basic organizations of the Gram-Schmidt orthogonalization process: classical Gram-Schmidt (CGS) and modified Gram-Schmidt (MGS). Both methods compute a factorization $A = QR$ where $A \in \Re^{m \times n}$, $R \in \Re^{n \times n}$ is an upper triangular matrix and $Q \in \Re^{m \times n}$ is an orthogonal matrix. The methods differ in their numerical properties with, in general, MGS preferred in practice.

In this chapter we define the corresponding incomplete forms of these methods: incomplete classical Gram-Schmidt (ICGS) and incomplete modified Gram-Schmidt (IMGS). Both incomplete methods compute a factorization $A = QR$ where R is the upper triangular matrix. The matrix Q, however, is not necessarily orthogonal due to selective computation of elements in R, i.e., the process of dropping elements is applied explicitly to R only. Basic properties of these methods are explored including existence and backward numerical error properties. The relationship of the factorizations computed by the two methods is examined theoretically under the assumption that $A^T A$ is an M-matrix and empirically when this assumption does not hold. No assumptions are made in this chapter concerning the method of choosing the elements that are dropped from R during the computation of the incomplete factorization. The effects of varying the dropping strategy are discussed in detail in Chapter 4.

3.1 The ICGS and IMGS algorithms

Let $P_n = \{(i,j) | i \neq j, \quad 1 \leq i, j \leq n\}$ and assume that the matrix $A \in \Re^{m \times n}$ has full column rank. Suppose we are given a set of index pairs P such that that $P \subset P_n$ and $(i,j) \in P$ implies that $1 \leq i < j \leq n$. The set P determines which elements of the matrix R will not be retained in the approximate factorization, i.e., P is the drop set. Given these assumptions, the ICGS factorization algorithm can be easily derived from CGS to yield the following algorithm:

Algorithm [Q, R] = ICGS [A, P]

begin

for $j = 1, 2, \ldots, n,$

$z = a_j$

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

$$r_{ij} = \begin{cases} 0 & (i,j) \in P \\ q_i^T a_j & (i,j) \notin P \end{cases}$$

$z = z - r_{ij} q_i$

endfor

$r_{jj} = \| z \|_2$

$q_j = z / r_{jj}$

endfor

end

Note that ICGS is identical to CGS with the exception that there is an incompleteness decision imposed on the assignment of the computed value of r_{ij}. This organization of the algorithm computes Q and R one column at a time. This implies that columns of A are read-only and therefore amenable to the use of a static data structure until each is converted into the associated column of Q whose storage requires a very simple form of dynamic data structure. The final values of the nonzero elements of R in the positions given by the set $P_n - P$ are also produced in their final form via the evaluation of a single dotproduct.

A standard implementation of MGS can be altered in a similar fashion to produce the following version of IMGS:

Algorithm [Q, R] = IMGS [A, P]

begin

for $k = 1, 2, \ldots, n,$

$r_{kk} = \| a_k \|_2$

$q_k = a_k / r_{kk}$

for $j = k+1, k+2, \ldots, n$

$$r_{kj} = \begin{cases} 0 & (k,j) \in P \\ q_k^T a_j & (k,j) \notin P \end{cases}$$

$a_j = a_j - q_k r_{kj}$

endfor

endfor

end

This form of MGS and the induced version of IMGS produce the nonzero elements of R in their final form using a row major ordering. Notice that the columns of Q are produced in an incremental fashion from the associated columns of A. Therefore, a more aggressive use of a dynamic data structure is required to store the columns of A as they are updated and increase in density. This can be avoided by rearranging the computations in MGS and IMGS. Note that the loop with index j in the version of IMGS above is a rank-1 update of the matrix formed by the previously updated $k+1$-st to n-th columns of A. The update is determined by q_k and the k-th row of R which is computed by a set of independent dotproducts of q_k with the columns of A to be updated. This can be replaced by the computationally equivalent loop with index i shown in the following:

Algorithm [Q, R] = IMGS [A, P]

begin

for $j = 1, 2, \ldots, n,$

$z = a_j$

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

$$r_{ij} = \begin{cases} 0 & (i,j) \in P \\ q_i^T z & (i,j) \notin P \end{cases}$$

$z = z - r_{ij} q_i$

endfor

$r_{jj} = \| z \|_2$

$q_j = z / r_{jj}$

endfor

end

The new form of the inner loop consists of a series of dependent dotproducts and triads. Note, however, that each column of A can be read once into a work vector z which is then updated by a series of vector projections to yield the associated final column of Q. As a result, the dynamic data structure issues are restored to the simple form encountered in the ICGS algorithm. The new version generates the same R and Q as the first version but in a column major ordering. All references to IMGS in the remainder of this thesis that involve performance data will imply the use of the column-oriented form. In some instances, the equivalent row-oriented form is more convenient for demonstrating the properties of IMGS.

The algorithmic descriptions above do not require any particular method of selecting the set P. The only condition imposed is that a diagonal element cannot be chosen as a zero element. Specific methods of zero element selection and their effects are discussed in Chapter 4.

3.2 Basic properties of ICGS and IMGS

In this section, the basic properties of the ICGS and IMGS algorithms are discussed. The issues considered include the existence of the factorization, the numerical stability of the computations, and the characteristics of the matrix Q^TQ which indicate the quality of the preconditioner.

3.2.1 Existence of the incomplete factorizations

One of the disadvantages of incomplete factorization is that often their existence cannot be guaranteed. For example, for least squares problems the lack of this guarantee causes difficulties with the use of an incomplete Cholesky factorization of the normal equations to precondition an iterative method like CGLS. Fortunately, ICGS and IMGS do not suffer from this problem. The proof of Theorem 1 demonstrates this relatively straightforward fact for IMGS.

Theorem 1 *If $A \in R^{m\times n}$, where $m \geq n$, has full rank then IMGS applied with a drop set $P \subset P_n$ completes and produces a factorization $A = QR$, where R is an upper triangular matrix with positive diagonal elements and Q is a nonsingular matrix.*

Proof: Let $a_i^{(j)}$ denote the i-th column of A after j steps of IMGS (see the row-oriented version of IMGS). From the algorithm we can see that

$$\begin{aligned}
q_1 &= a_1^{(0)}/r_{11}, \\
q_2 &= a_2^{(1)}/r_{22}, \\
&= (a_2^{(0)} - q_1 r_{12})/r_{22}, \\
q_3 &= a_3^{(2)}/r_{33}, \\
&= (a_3^{(1)} - q_2 r_{23})/r_{33}, \\
&= (a_3^{(0)} - q_1 r_{13} - q_2 r_{23})/r_{33}, \\
&\vdots
\end{aligned}$$

The algorithm cannot complete if at some step k, $r_{kk} = \| a_k^{(k-1)} \|_2 = 0$. This means that we have

$$\begin{aligned}
a_k^{(k-1)} &= a_k^{(k-2)} - q_{k-1} r_{k-1\,k} \\
&= a_k^{(k-3)} - q_{k-2} r_{k-2\,k} - q_{k-1} r_{k-1\,k} \\
&\vdots \\
&= a_k^{(0)} - q_1 r_{1k} - \ldots - q_{k-1} r_{k-1\,k} \\
&= 0
\end{aligned}$$

or equivalently

$$a_k^{(0)} = \begin{pmatrix} q_1 & q_2 & q_3 & \ldots & q_{k-1} \end{pmatrix} \begin{pmatrix} r_{1k} \\ r_{2k} \\ \vdots \\ r_{k-1\,k} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Therefore,

$$\begin{pmatrix} q_1 & q_2 & q_3 & \ldots & q_{k-1} & a_k^{(0)} \end{pmatrix}$$

is rank deficient. But, $\begin{pmatrix} q_1 & q_2 & q_3 & \ldots & q_{k-1} \end{pmatrix}$ is a basis of *span* $\begin{pmatrix} a_1^{(0)} & a_2^{(0)} & a_3^{(0)} & \ldots & a_{k-1}^{(0)} \end{pmatrix}$ and $\begin{pmatrix} a_1^{(0)} & a_2^{(0)} & a_3^{(0)} & \ldots & a_k^{(0)} \end{pmatrix}$ is assumed to be of full rank. Therefore,

$$\begin{pmatrix} q_1 & q_2 & q_3 & \ldots & q_{k-1} & a_k^{(0)} \end{pmatrix}$$

is of full rank which is a contradiction. So if A is full rank, $r_{kk} \neq 0$ for $k = 1, 2, \ldots, n$ and the factorization must exist. □

A similar existence proof is possible for ICGS and is stated as Theorem 2 without proof.

Theorem 2 *If $A \in R^{m \times n}$, where $m \geq n$, has full rank then ICGS applied with a drop set $P \subset P_n$ completes and produces a factorization $A = QR$, where R is an upper triangular matrix with positive diagonal elements and Q is a nonsingular matrix.*

Of course, the assumptions above also guarantee that CGS and MGS complete when using exact arithmetic. Furthermore, CGS and MGS generate identical Q and R matrices although MGS tends to be preferred when finite precision is used. This property does not carry over to the incomplete factorizations and IMGS and ICGS can produce different Q and R matrices. This can be seen from the following simple example. Let

$$A = \begin{pmatrix} 0 & 0 & 1 & -2 \\ 1 & 0 & 1 & -1 \\ -1 & 1 & 0 & 1 \\ 0 & 1 & -1 & -1 \end{pmatrix}.$$

If we apply IMGS with the drop set P taken as the complement of the nonzero position set $\hat{P} = \{(1,1), (1,4), (2,2), (3,3), (3,4), (4,4)\}$, then the upper triangular factor produced is

$$R_{IMGS} = \begin{pmatrix} \sqrt{2} & 0 & 0 & -\sqrt{2} \\ & \sqrt{2} & 0 & 0 \\ & & \sqrt{3} & -1/\sqrt{3} \\ & & & \sqrt{42}/3 \end{pmatrix}.$$

Applying ICGS with the same drop set yields

$$R_{ICGS} = \begin{pmatrix} \sqrt{2} & 0 & 0 & -\sqrt{2} \\ & \sqrt{2} & 0 & 0 \\ & & \sqrt{3} & -2/\sqrt{3} \\ & & & \sqrt{5} \end{pmatrix}.$$

3.2.2 Error analysis of IMGS

The theorems above guarantee the existence of IMGS and ICGS when exact arithmetic is used. The use of finite precision will, of course, affect the algorithms in practice. For example, it may cause the algorithm to 'break down'. The next set of results investigate the effect of rounding errors on IMGS. This is accomplished by generalizing the standard result of Björck's error analysis of MGS [8].

Theorem 3 *Suppose $A \in R^{m\times n}$ has full rank and P is the drop set which specifies the forced zero positions of the matrix R produced by IMGS. If $\bar{Q}\bar{R} = A + E$, where $\bar{Q}$ and $\bar{R}$ are the computed factors produced by IMGS then E satisfies:*

$$\| E \|_F \leq 1.54 \cdot 2^{-t} \| AD \|_F,$$

where, t is the number of bits in the mantissa,

$$D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}$$

and d_i, $1 \leq i \leq n$ is the number of nonzero elements in the column i of R.

Proof: Since we will use some results due to Björck [8], we use his notation. Computed quantities are distinguished by bars. At the k-th step of MGS,

$$\bar{a}_j^{(k+1)} = fl(\bar{a}_j^{(k)} - \bar{r}_{kj}\bar{q}_k),$$

for $j = k+1, \ldots, n$ where $fl(x\text{`op'}y) = (x\text{`op'}y)(1+\epsilon)$, $|\epsilon| \leq 2^{-t}$ and t is the number of digits in the precision of the floating point number. Define the related errors $\delta_j^{(k)}$ by

$$\bar{a}_j^{(k+1)} = \bar{a}_j^{(k)} - \bar{r}_{kj}\bar{q}_k - \delta_j^{(k)}. \tag{3.1}$$

Björck shows that:

$$\| \delta_j^{(k)} \|_2 \leq 1.45 \cdot 2^{-t} \| \bar{a}_j^{(k)} \|_2 . \tag{3.2}$$

Applying this result to the computation of the k-th update of the j-th column in IMGS, we have

$$\bar{a}_j^{(k+1)} = \begin{cases} \bar{a}_j^{(k)} - \bar{r}_{kj}\bar{q}_k - \delta_j^{(k)} & (k,j) \notin P \\ \bar{a}_j^{(k)} & (k,j) \in P \end{cases} \tag{3.3}$$

and

$$\| \delta_j^{(k)} \|_2 \begin{cases} \leq 1.45 \cdot 2^{-t} \| \bar{a}_j^{(k)} \|_2 & (k,j) \notin P \\ = 0 & (k,j) \in P \end{cases} \tag{3.4}$$

The final computation for the j-th column is its normalization which yields

$$\bar{q}_j = \frac{\bar{a}_j^{(j)}}{\bar{r}_{jj}}(1+\epsilon), \quad |\epsilon| \leq 2^{-t}. \tag{3.5}$$

Now if we define

$$\delta_j^{(j)} = \bar{a}_j^{(j)} - \bar{r}_{jj}\bar{q}_j \tag{3.6}$$

it follows that the error due to normalization is

$$\| \delta_j^{(j)} \|_2 \leq 2^{-t} \| \bar{a}_j^{(j)} \|_2 \, . \tag{3.7}$$

Let δ_j be the j-th column of $A - \bar{Q}\bar{R}$. It follows that

$$\delta_j = a_j - \sum_{(k,j)\notin P} \bar{r}_{kj}\bar{q}_k. \tag{3.8}$$

By repeatedly applying (3.3) to (3.8) we get

$$\delta_j = \sum_{(k,j)\notin P, k\neq j} \delta_j^{(k)} + a_j^{(j)} - \bar{r}_{jj}\bar{q}_j. \tag{3.9}$$

Applying (3.6) to (3.9) yields

$$\delta_j = \sum_{(k,j)\notin P} \delta_j^{(k)}.$$

Applying the bounds of (3.4) and (3.7) yields

$$\| \delta_j \|_2 \leq 1.45 \cdot 2^{-t} \sum_{(k,j)\in P} \| \bar{a}_j^{(k)} \|_2 \, .$$

In order to simplify the inequalities, we make the same assumptions as Björck:

$$\begin{array}{l} m \geq 2 \\ 2n(m+1)2^{-t} < 0.01 \end{array} .$$

Based on these assumptions, Björck derives

$$\| \bar{a}_j^{(k+1)} \|_2 < (1 + 1.05 \cdot 2^{-t}) \| \bar{a}_j^{(k)} \|_2,$$

By applying it to the incomplete case, we then get

$$\begin{aligned}
\| \delta_j \|_2 &\leq 1.45 \cdot 2^{-t} \sum_{(k,j)\notin P} \| \bar{a}_j^{(k)} \|_2 \\
&= 1.45 \cdot 2^{-t} \| a_j \|_2 \sum_{k=1}^{d_j} (1 + 1.05 \cdot 2^{-t})^k \\
&= 1.45 \cdot 2^{-t} \| a_j \|_2 \, 1.06 \cdot (1 + 1.05 \cdot 2^{-t}) \cdot d_j
\end{aligned}$$

where d_j is the number of nonzero elements in j^{th} column of R.

Thus,

$$\| \delta_j \|_2 \leq 1.54 \cdot 2^{-t} \cdot d_j \cdot \| a_j \|_2$$

Finally, if we let $\bar{Q}\bar{R} = A + E$ then it follows that

$$\begin{aligned}
\| E \|_F &= (\sum_{j=1}^{n} \| \delta_j \|_2^2)^{1/2} \\
&\leq (\sum_{j=1}^{n} (1.54 \cdot 2^{-t}) \| d_j a_j \|_2)^{1/2} \\
&= 1.54 \cdot 2^{-t} (\sum_{j=1}^{n} \| d_j a_j \|_2)^{1/2} \\
&= 1.54 \cdot 2^{-t} \| AD \|_F
\end{aligned}$$

where

$$D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}$$

and d_i is defined above. □

Theorem 3 shows that the computed errors are related to the norm of matrix A and the drop set P. In the extreme case, when P is chosen to be the set of all the off diagonal matrix elements, then R is a diagonal matrix and IMGS corresponds to column scaling. In this case, D is the identity matrix and we have

$$\| E \|_F \leq 1.54 \cdot 2^{-t} \| A \|_F \, .$$

When P is chosen to be the empty set, IMGS is the same as MGS. The resulting bound,

$$\| E \|_F \leq 1.54 \cdot 2^{-t} n \| A \|_F,$$

is the same as the bound Björck derives for MGS [8]. Note that this proof has only exploited the sparsity in R implied by the drop set P, i.e., we have modified a theorem that was derived for a dense A. Björck's result and Theorem 3 could both be modified to take into account the sparsity of A and Q in order to reduce the coefficients in the bounds used in the proof.

From the error bounds we can see that, as expected, since IMGS essentially drops computations from MGS its computed error is smaller than that of MGS assuming that there is no significant cancellation of rounding error. This suggests that the computation of IMGS will be less likely than MGS to break down due to possible numerical rank deficiency of A.

3.2.3 Quality of IMGS and ICGS preconditioners

Since the goal of both IMGS and ICGS is to produce a good preconditioner for CGLS we need an incomplete factorization that approximates the complete orthogonal factorization to some extent. Specifically, we want $R^{-T}A^TAR^{-1} = Q^TQ$ to approximate the identity matrix I. Q^TQ is the coefficient matrix of the normal equations of the preconditioned least squares problem; therefore, its spectrum governs the convergence of CGLS. As with most preconditioners, bounding the difference between Q^TQ and I requires imposing further restrictions on A. A typical restriction when preconditioning symmetric and nonsymmetric linear systems requires that the coefficient matrix is an M-matrix.

In Theorems 7 and 8, we show the consequences of a similar assumption applied to the least squares problem. In particular, we consider the Q^TQ matrices that arise from applying IMGS and ICGS to a problem for which A^TA is an M-matrix and derive a bound on $\| I - Q^TQ \|_2$. This assumption implies bounds on the angles between pairs of columns of A. Since the off-diagonal elements of A^TA are all negative or zero we have $\pi/2 \leq \theta_{i,j} \leq 3\pi/2$ where $\theta_{i,j}$ is the angle between columns i and j of A. The full column rank of A implies an additional condition of a hole in the middle of this bounding interval, i.e., $\theta_{i,j} \neq \pi$. Intuitively, these bounds mean that the columns are reasonably well spread out in $\Re^m$ for well-conditioned problems. There cannot be a subset which forms a narrow cones of vectors all pointing in the same general direction nor can there be a subset which forms a bundle of vectors consisting of two narrow cones placed end to end.

Before the presentation of these results, we state three theorems and four lemmas which are needed to derive the bounds. Theorem 4 states that one step of Gaussian elimination preserves the properties of an M-matrix. Theorem 5 states a condition on a range of perturbations to an M-matrix that yields another M-matrix.

Theorem 4 *(due to Fan [50]). Let $A = (a_{ij})$ be a nonsingular matrix and let $A^{(1)} = (a_{ij}^{(1)})$ be the matrix that arises by zeroing out the off-diagonal elements in the first column of A via one step of Gaussian elimination. If A is an M-matrix then $A^{(1)}$ exists and is also an M-matrix.*

Theorem 5 *(due to Meijerink and van der Vorst [50]). Let $A = (a_{ij})$ be an $n \times n$ M-matrix. If the matrix $B = (b_{ij})$ is such that*

$$a_{ij} \leq b_{ij} \leq 0 \quad for \quad i \neq j$$

and $0 < a_{ii} \leq b_{ii}$ then B is also an M-matrix.

We also need the well-known fact that MGS applied to A produces a upper triangular factor R that is identical to the matrix L^T produced by the Cholesky factorization of $A^T A = LL^T$. This is summarized without proof in Lemma 1.

Lemma 1 *Let $A^{(1)}$ and $R^{(1)}$ be the matrices that arise from one step of modified Gram-Schmidt (MGS) applied to A, and let $B^{(1)}$ and $L^{(1)}$ be the matrices that arise from one step of Cholesky factorization of $B = A^T A$. Then $B^{(1)} = {A^{(1)}}^T A^{(1)}$ and ${L^{(1)}}^T = R^{(1)}$.*

Given these results we can state the following about the effect of MGS on the properties of an M-matrix.

Lemma 2 *Let $A \in R^{m \times n}$ be a matrix of full rank and let $A^{(1)}$ be the matrix that results after applying one step MGS. If $A^T A$ is an M-matrix, then ${A^{(1)}}^T A^{(1)}$ is also an M-matrix.*

Proof: This results directly from Theorem 4 and Lemma 1. □

The results above can be generalized to IC and IMGS. The generalization to IC appears in the literature and can be stated as Lemma 3.

Lemma 3 *(due to Meijerink and van der Vorst [50]). Let $A = (a_{ij})$ be a symmetric positive definite matrix and let $A^{(1)} = (a_{ij}^{(1)})$ be the matrix that arises by from one step of IC. If A is an M-matrix then $A^{(1)}$ exists and is also an M-matrix.*

A similar property can be derived for IMGS.

Lemma 4 *For a given $A \in R^{m \times n}$ and a set $P \subset P_n$, Let $\tilde{A}$ be the matrix arises from one step of IMGS. If $A^T A$ is an M-matrix, then $\tilde{A}^T \tilde{A}$ is also an M-matrix.*

Proof: Let $A^T A = (a_{ij})$, ${A^{(1)}}^T A^{(1)} = (a_{ij}^{(1)})$ and $\tilde{A}^T \tilde{A} = (\tilde{a}_{ij})$, where $A^{(1)}$ is the matrix that arises from one step of complete MGS on A. If we can show that $a_{ij}^{(1)} \leq \tilde{a}_{ij} \leq 0 \quad for \quad i \neq j$ and $0 < a_{ii}^{(1)} \leq \tilde{a}_{ii}$, then the desired result follows Theorem 5 and Lemma 2.

First, we see from the proof of Lemma 2 that $a_{ij} = a_i^T a_j$ and

$$a_{ij}^{(1)} = a_i^T (I - qq^T) a_j = a_i^T a_j - \frac{(a_i^T a_1)(a_1^T a_j)}{a_1^T a_1}$$

Because $A^T A$ is an M-matrix, we then have $a_i^T a_1 \leq 0$ for $2 \leq i \leq n$, so that $a_{ij}^{(1)} \leq a_{ij}$.

Consider $\tilde{a}_{ij}$ for $i \neq j$. There are three cases:

Case 1: $(1,i) \in P$ and $(1,j) \in P$. In this case, there is no modification on a_i and a_j. We have

$$\tilde{a}_{ij} = a_i^T a_j = a_{ij} \geq a_{ij}^{(1)}$$

Case 2: $(1,i) \in P$ and $(1,j) \notin P$, in this case, a_j is updated by IMGS to $(I - qq^T)a_j$, so we get

$$\tilde{a}_{ij} = a_i^T (I - qq^T) a_j = a_{ij}^{(1)}.$$

Case 3: $(1,i) \notin P$ and $(1,j) \notin P$, In this case, we have both a_i and a_j being updated,

$$\tilde{a}_{ij} = a_i^T (I - qq^T)(I - qq^T) a_j = a_i^T (I - qq^T) a_j = a_{ij}^{(1)}$$

Combining all the arguments above, we get that for all $i \neq j$ we have

$$a_{ij}^{(1)} \leq \tilde{a}_{ij} \leq 0 \quad for \quad i \neq j$$

For the diagonal elements, we have either $\tilde{a}_{ii} = a_i^T (I - qq^T) a_i = a_{ii}^{(1)}$ or $\tilde{a}_{ii} = a_i^T a_i = a_{ii} \geq a_{ii}^{(1)}$. In both cases we have $0 < a_{ii}^{(1)} \leq \tilde{a}_{ii}$. It follows by Lemma 2 that $A^{(1)^T} A^{(1)}$ is an M-matrix. From the relations shown above and Theorem 5 it follows that $\tilde{A}^T \tilde{A}$ is also an M-matrix. □

We need one final result from the literature before stating our main result in Theorem 7.

Theorem 6 *(due to Young [63]). Let $A = M - N$ be a regular splitting of the nonsingular matrix A. Then $A^{-1} \geq 0$ if and only if $\rho(M^{-1}N) < 1$. Moreover, if $A^{-1} \geq 0$, then*

$$\rho(M^{-1}N) = \rho(A^{-1}N)/(1 + \rho(A^{-1}N)).$$

We can now state a theorem which characterizes the difference $I - Q^T Q$ that results from IMGS under the assumption that $A^T A$ is an M-matrix.

Theorem 7 *Let $A \in R^{m \times n}$ be a matrix of full rank. If $A^T A$ is an M-matrix and $Q \in R^{m \times n}$, $R \in R^{n \times n}$ are the matrices that are produced by applying IMGS with a given $P \subset P_n$, then*

$$Q^T Q = R^{-T} A^T A R^{-1} = I - E$$

is a regular splitting with $E \geq 0$, all of the diagonal elements of E equal to 0, and $\rho(E) < 1$, where $\rho(E)$ is the spectral radius of E.

Proof: Let $A = (a_1^{(0)}, \ldots, a_n^{(0)})$. From the description of IMGS we have on the k-th step

$$\begin{pmatrix} a_k^{(k-1)} & a_{k-1}^{(k-1)} & a_{k-2}^{(k-1)} & \ldots & a_n^{(k-1)} \end{pmatrix} =$$

$$\begin{pmatrix} a_k^{(k)} & a_{k-1}^{(k)} & a_{k-2}^{(k)} & \ldots & a_n^{(k)} \end{pmatrix} \begin{pmatrix} r_{kk} & r_{k\,k+1} \ldots r_{kn} \\ & \\ 0 & I \\ & \end{pmatrix}$$

where $r_{kk} = \| a_k^{(k-1)} \|_2$, $a_k^{(k)} = a_k^{(k-1)}/r_{kk}$, and for $j = k+1, \ldots, n$

$$r_{kj} = \begin{cases} a_k^{(k-1)^T} a_j^{(k-1)}/r_{kk} & (k,j) \notin P \\ 0 & (k,j) \in P \end{cases}$$

and

$$a_j^k = \begin{cases} a_j^{(k-1)} - (r_{kj}/r_{kk}) a_k^{(k-1)} & (k,j) \notin P \\ a_j^{(k-1)} & (k,j) \in P \end{cases} .$$

Denote for $j = k+1, \ldots, n$

$$e_{kj} = \begin{cases} 0 & (k,j) \notin P \\ a_k^{(k-1)^T} a_j^{(k-1)}/r_{kk} & (k,j) \in P \end{cases} .$$

Let R_i be an $n \times n$ upper triangular matrix which is equal to identity matrix except for the i-th row which is equal to $(0 \ldots 0 r_{ii} \ldots r_{in})$ We can write R as

$$R = R_n R_{n-1} \ldots R_1 .$$

Now we start to construct the splitting,

$$R_1^{-T} A^T A R_1^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & \begin{pmatrix} a_2^{(1)^T} \\ \vdots \\ a_n^{(1)^T} \end{pmatrix} \begin{pmatrix} a_2^{(1)} & \ldots & a_n^{(1)} \end{pmatrix} \end{pmatrix} - \underbrace{\begin{pmatrix} 0 & -a_1^{(1)^T} \begin{pmatrix} a_2^{(1)} & \ldots & a_n^{(1)} \end{pmatrix} \\ -\begin{pmatrix} a_2^{(1)^T} \\ \vdots \\ a_n^{(1)^T} \end{pmatrix} a_1^{(1)} & 0 \end{pmatrix}}_{E_1}$$

Note that by definition of IMGS,

$$e_{1j} = a_1^{(1)^T} a_j^{(1)} = \begin{cases} 0 & (1,j) \notin P \\ a_1^{(0)^T} a_j^{(0)}/r_{11} & (1,j) \in P \end{cases} .$$

Since $A^T A$ is an M-matrix, we have $a_1^{(0)^T} a_j^{(0)} \leq 0$. Therefore, $e_{1j} \leq 0$, $r_{11} > 0$, and $r_{1j} \leq 0$ for $j = 2, \ldots, n$. It follows that $E_1 \geq 0$ and $R_1^{-1} \geq 0$. By Lemma 4, we know that

$$\begin{pmatrix} a_2^{(1)^T} \\ \vdots \\ a_n^{(1)^T} \end{pmatrix} \begin{pmatrix} a_2^{(1)} & \ldots & a_n^{(1)} \end{pmatrix}$$

is an M-matrix. Repeating the process we have,

$$R_2^{-T}R_1^{-T}A^TAR_1^{-1}R_2^{-1} = \left\{ \left(\begin{array}{cc} I_2 & 0 \\ 0 & \begin{pmatrix} a_3^{(2)^T} \\ \vdots \\ a_n^{(2)^T} \end{pmatrix} \begin{pmatrix} a_3^{(2)} & \dots & a_n^{(2)} \end{pmatrix} \end{array} \right) - \right.$$

$$\left. \underbrace{\begin{pmatrix} 0 & 0 & 0\dots0 \\ 0 & 0 & -a_2^{(2)^T}\begin{pmatrix} a_3^{(2)} & \dots & a_n^{(2)} \end{pmatrix} \\ 0 & -\begin{pmatrix} a_3^{(2)^T} \\ \vdots \\ a_n^{(2)^T} \end{pmatrix} a_2^{(2)} & 0 \end{pmatrix}}_{E_2} \right\} - R_2^{-T}E_1R_2^{-1}$$

$$= \left(\begin{array}{cc} I_2 & 0 \\ 0 & \begin{pmatrix} a_3^{(2)^T} \\ \vdots \\ a_n^{(2)^T} \end{pmatrix} \begin{pmatrix} a_3^{(2)} & \dots & a_n^{(2)} \end{pmatrix} \end{array} \right) - E_2 - R_2^{-T}E_1R_2^{-1}$$

As in the first step we have $E_2 \geq 0$ and $R_2^{-1} \geq 0$. Combining these two steps we have $E_2 + R_2^{-T}E_1R_2^{-1} \geq 0$.

Similarly, define the matrix E_i as being equal to zero except at i-th column and i-th row. The i-th row is equal to

$$\left(\underbrace{0\dots0}_{i} a_i^{(i)^T}(a_{i+1}^{(i)}\dots a_n^{(i)}) \right).$$

It can be seen that $E_i \geq 0$.

If we continue the construction in the same manner as above after n steps, we have

$$\begin{aligned} R_n^{-T}\dots R_1^{-T}A^TAR_1^{-1}\dots R_n^{-1} &= \\ & I - \underbrace{\sum_{i=1}^{n} R_n^{-T}\dots R_{i+1}^{-T}E_iR_{i+1}^{-1}\dots R_n^{-1}}_{E} \\ &= R^{-T}A^TAR^{-1} \end{aligned}$$

Repeatedly applying Lemma 4 shows that $R^{-T}A^TAR^{-1}$ is an M-matrix. Also, since $R_i^{-1} \geq 0$ and $E_i \geq 0$, for $i = 1, \dots, n$ we have $E \geq 0$ and the splitting is regular. Finally, since we also have $(R^{-T}A^TAR^{-1})^{-1} > 0$, Theorem 6 implies that $\rho(E) < 1$. □

A similar result can be proven for ICGS. However, in order to prove some results in Chapter 7 we only need the following simpler version.

Theorem 8 *Let $A \in R^{m\times n}$. If A^TA is an M-matrix and $Q \in R^{m\times n}$, $R \in R^{n\times n}$ are the matrices that arise from ICGS with a given $P \subset P_n$, then there is a splitting*

$$Q^TQ = R^{-T}A^TAR^{-1} = I - E$$

with $E \geq 0$ and all the diagonal elements of E are equal to 0.

Proof: By induction on n, the number of columns of the matrix. It is trivial for n=1. Assume that for problems $n-1$ or fewer columns, the statement is true. Now for a problem with n columns, we prove the statement.

Let $A = (A_1, a_n)$, $Q = (Q_1, q_n)$, $R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & r_{22} \end{pmatrix}$ and $E = \begin{pmatrix} E_{11} & E_{12} \\ E_{12}^T & e_{22} \end{pmatrix}$ be partitioned, such that both A_1 and Q_1 have $n-1$ columns, and R_{11} and E_{11} are $(n-1)$-by-$(n-1)$ matrices.

By the induction hypothesis, after $n-1$ steps of ICGS, we have $A_1 = Q_1R_{11}$, $Q_1^TQ_1 = I - E_{11}$ and $E_{11} \geq 0$. Let $x = Q_1^Ta_n = R_{11}^{-T}A_1^Ta_n$. Because A^TA is an M-matrix, the vector $A_1^Ta_n \leq 0$, so that $x \leq 0$. Notice that at positions $(i,n) \notin P$, we have $x_i = r_{i,n}$, so we have $R_{12} \leq 0$. Let $d = Q_1^Ta_n - R_{12}$ be the dropped vector in the n-th step; then $d \leq 0$.

Note that

$$\begin{aligned} Q^TQ &= \begin{pmatrix} Q_1^TQ_1 & Q_1^Tq_n \\ q_n^TQ_1 & q_n^Tq_n \end{pmatrix} \\ &= I - \begin{pmatrix} E_{11} & -Q_1^Tq_n \\ -q_n^TQ_1 & 0 \end{pmatrix} \\ &= I - \begin{pmatrix} E_{11} & E_{12} \\ E_{12}^T & e_{22} \end{pmatrix} \end{aligned}$$

and

$$\begin{aligned} E_{12} &= -Q_1^Tq_n \\ &= -Q_1^T(a_n - Q_1R_{12})/r_{22} \\ &= -(Q_1^Ta_n - R_{12} + E_{11}R_{12})/r_{22} \\ &= -(d + E_{11}R_{12})/r_{22} \end{aligned}$$

Because $d \leq 0$, $E_{11} \geq 0$ and $R_{12} \leq 0$, we conclude that $E_{12} = Q_1^Tq_n \geq 0$ and $e_{22} = 0$. Together, we have $E \geq 0$. □

If A^TA is an M-matrix, Theorem 7 bounds the distance of Q^TQ and an orthogonal matrix since it implies that $\rho(Q^TQ) \leq 2$. Unfortunately, it does not guarantee an improvement in the condition of Q^TQ relative to A^TA in general since $\lambda_{min}(A^TA)$ is not necessarily a lower bound on $\lambda_{min}(Q^TQ)$. In practice, however, we have found that one step of IMGS tends to behave like one step of MGS in that one of the eigenvalues is brought close to 1 and the remaining ones tend to stay in an interval whose lower and upper bounds do not change significantly for the worse. (For MGS one of the eigenvalues is made exactly 1 and the remaining ones are in an interval bounded by the minimum and maximum eigenvalues of the normal equations of the matrix before the step is performed.)

A similar result is contained in [50] and states that $A^TA = LL^T - \hat{E}$ is a regular splitting where L results from applying IC to A^TA which is an M-matrix and $\hat{E} \geq 0$ and $\rho(\hat{E}) < 1$. It is straightforward to transform this result into one similar to Theorem 7. Specifically, if L is used to precondition the least squares problem then it can be shown that $\tilde{Q}^T\tilde{Q} = L^{-1}A^TAL^{-T} = I - \tilde{E}$ is a regular splitting satisfying conditions on $\tilde{E}$ identical to those on E of Theorem 7. However, as is shown in the next section, IMGS and IC do not necessarily produce the same triangular preconditioning matrix for a given set P. In fact, it is shown in Chapter 7 that IMGS will produce a preconditioner superior to IC for a given P when A^TA is an M-matrix.

3.3 Relationship of IMGS on A to IC on A^TA

Since CGS and MGS produce the same R as Cholesky factorization applied to A^TA, it is reasonable to ask if the incomplete Gram-Schmidt methods can shed any light on the application of incomplete Cholesky preconditioning of CG to positive definite systems. Before addressing this question, we describe the incomplete Cholesky factorization algorithm in a fashion similar to our IMGS description.

Given a symmetric positive definite matrix $A \in R^{n\times n}$, and a set $P \subset P_n$, the incomplete Cholesky factorization algorithm (IC) generates an upper triangular matrix $U \in R^{n\times n}$ as follows:

Algorithm [U]=IC[A,P]

begin

for $k = 1, 2, \ldots, n,$

  if $a_{kk} > 0$ **then**

    $u_{kk} = \sqrt{a_{kk}}$

    for $j = k+1, k+2, \ldots, n$

$$u_{kj} = \begin{cases} 0 & (k,j) \in P \\ a_{kj}/\sqrt{a_{ii}} & (k,j) \notin P \end{cases}$$

    endfor

    for $j = k+1, k+2, \ldots, n$

      for $i = k+1, k+2, \ldots, n$

        $a_{ij} = a_{ij} - u_{ki}u_{kj}$

      endfor

    end

else

quit (incomplete Cholesky factorization can not complete)

endif

endfor

end

It was noted earlier that, in general, applying IMGS and ICGS with the same set P does not produce the same R. Occasionally, however, they do produce the same R. This is the case for the following matrix:

$$A = \begin{pmatrix} 0 & 0 & 1.0000 & -1.0001 \\ 0 & 0 & 1.0000 & -1.0000 \\ -1.0000 & 1.0000 & 0 & 1.0000 \\ 0 & 1.0000 & -1.0000 & -1.0000 \end{pmatrix},$$

If we normalize the columns we get,

$$B = \begin{pmatrix} 0 & 0 & 0.57735 & -0.50004 \\ 0 & 0 & 0.57735 & -0.49999 \\ -1.0000 & 0.70711 & 0 & 0.49999 \\ 0 & 0.70711 & -0.57735 & -0.49999 \end{pmatrix}$$

and the corresponding normal equations are

$$C = B^T B = \begin{pmatrix} 1.00000 & -0.70711 & 0 & -0.49999 \\ -0.70711 & 1.0000 & -0.40825 & 0 \\ 0 & -0.40825 & 1.00000 & -0.28870 \\ -0.49999 & 0 & -0.28870 & 1.00000 \end{pmatrix}.$$

Let the set of non-zeros positions kept in R be the set of positions of non-zero elements in C. The factor produced by IMGS and ICGS is

$$R_{IMGS} = \begin{pmatrix} 1.00000 & -0.70711 & 0 & -0.49999 \\ & 0.70711 & -0.57735 & 0 \\ & & 0.81650 & -0.70712 \\ & & & 0.49999 \end{pmatrix}.$$

Even though IMGS and ICGS produce the same factorization, this example shows that this is not enough to also guarantee the equivalence of the factor, R_{IC}, produced by IC. For the problem above we have

$$R_{IC} = \begin{pmatrix} 1.0000 & -0.70711 & 0 & -0.49999 \\ & 0.70711 & -0.57735 & 0 \\ & & 0.81650 & -0.35358 \\ & & & 0.79057 \end{pmatrix}.$$

So a reasonable first question to consider is the conditions under which IMGS and IC produce the same factorization.

It is well-known that the results of an IC factorization are independent of the elements of the symmetric positive definite matrix that are in positions contained in the drop set P. In other words, if $[U_1] = IC[A, P]$ and $[U_2] = IC[\hat{A}, P]$ where $\hat{A}$ is a matrix derived by setting to 0 all of the elements of A that are in positions in P, then $U_1 = U_2$. Lemma 5 is a more useful form of this fact that relates two different matrices that have identical non-zero elements in positions not in P.

Lemma 5 . *Suppose $A \in R^{n\times n}$ and $B \in R^{n\times n}$ are symmetric positive definite matrices for which IC completes using a drop set $P \subset P_n$. Let the matrices U and T be the upper triangular matrices that arise from IC on A and B respectively, that is*

$$[U] = IC[A, P] \quad \textit{and} \quad [T] = IC[B, P].$$

If A and B are such that $a_{ij} = b_{ij}$ for all $(i, j) \notin P$ then $T = U$.

Proof: Follows immediately the fact that the algorithm is equivalent to dropping all elements from the input matrix that are in P and then performing the factorization with the subsequent dropping of fill-in elements. $\Box$

Lemma 6 relates the IMGS factorization of A and the IC factorization of $B = A^T A$. Let $A^{(1)}$ be the matrix that arises from one step of IMGS on A and $B^{(1)}$ be the matrix that arises from one step of IC on $B = A^T A$. Then certain elements of ${A^{(1)}}^T A^{(1)}$ are equal to the corresponding elements of $B^{(1)}$.

Lemma 6 *Given $A \in R^{m\times n}$, $P \subset P_n$, Let $B = A^T A$, and suppose $R^{(1)}$ and $A^{(1)}$ are the matrices that arise from one step of IMGS on A such that $A = A^{(1)} R^{(1)}$. Suppose $B^{(1)}$ and $U^{(1)}$ are the matrices that arise from one step of IC on B such that $B = {U^{(1)}}^T B^{(1)} U^{(1)}$. Denote $A^{(1)} = (a_1^{(1)}, a_2^{(1)}, \ldots a_n^{(1)})$ and $a_{ij}^{(1)} = {a_i^{(1)}}^T a_j^{(1)}$, Then $U^{(1)} = R^{(1)}$ and $a_{ij}^{(1)} = b_{ij}^{(1)}$ for those $1 < i, j \leq n$, such that either both $(1, i)$ and $(1, j)$ are in P, or both $(1, i)$ and $(1, j)$ are not in P.*

Proof: By definition we have

$$U^{(1)} = \begin{pmatrix} u_{11} & u_{12} \ldots u_{1n} \\ & \\ 0 & I \\ & \end{pmatrix}$$

where $u_{11} = \sqrt{b_{11}^{(0)}}$ and

$$u_{1j} = \begin{cases} 0 & (1, j) \in P \\ b_{1j}^{(0)} / \sqrt{b_{11}^{(0)}} & (1, j) \notin P \end{cases} .$$

We also have

$$R^{(1)} = \begin{pmatrix} r_{11} & r_{12} \dots r_{1n} \\ & \\ 0 & I \\ & \end{pmatrix}$$

where $r_{11} = \| a_1^{(0)} \|_2 = \sqrt{a_1^{(0)^T} a_1^{(0)}} = \sqrt{b_{11}^{(0)}}$ and

$$r_{1j} = \begin{cases} 0 & (1,j) \in P \\ a_1^{(0)^T} a_j^{(0)} / \sqrt{a_1^{(0)^T} a_1^{(0)}} = b_{1j} / \sqrt{b_{11}^{(0)}} & (1,j) \notin P \end{cases}.$$

We can see then that $R^{(1)} = U^{(1)}$.

Assume that $(1,i)$ and $(1,j)$ are both in P. From the description of the algorithms, we can see that

$$a_{ij}^{(1)} = a_i^{(1)^T} a_j^{(1)} = a_i^{(0)^T} a_j^{(0)} = b_{ij}^{(0)}$$

and $b_{ij}^{(1)} = b_{ij}^{(0)}$. So that $a_{ij}^{(1)} = b_{ij}^{(1)}$. If both $(1,i)$ and $(1,j)$ are not in P, then we have

$$\begin{aligned} a_{ij}^{(1)} &= a_i^{(1)^T} a_j^{(1)} \\ &= a_i^{(0)^T} \left(I - \frac{a_1^{(0)} a_1^{(0)^T}}{a_1^{(0)^T} a_1^{(0)}}\right) a_j^{(0)} \\ &= a_i^{(0)^T} a_j^{(0)} - \frac{(a_i^{(0)^T} a_1^{(0)})(a_1^{(0)^T} a_j^{(0)})}{a_1^{(0)^T} a_1^{(0)}} \\ &= b_{ij}^{(0)} - \left(\frac{b_{1i}^{(0)}}{\sqrt{b_{11}^{(0)}}}\right) \left(\frac{b_{1j}^{(0)}}{\sqrt{b_{11}^{(0)}}}\right) \end{aligned}$$

and

$$\begin{aligned} b_{ij}^{(1)} &= b_{ij}^{(0)} - u_{1i} u_{1j} \\ &= b_{ij}^{(0)} - \left(\frac{b_{1i}^{(0)}}{\sqrt{b_{11}^{(0)}}}\right) \left(\frac{b_{1j}^{(0)}}{\sqrt{b_{11}^{(0)}}}\right). \end{aligned}$$

Together, we have that $a_{ij}^{(1)} = b_{ij}^{(1)}$ for both cases. $\square$

The next theorem shows that for some special $P \subset P_n$, the upper triangular matrix generated by applying IMGS to A is equal to that from applying IC to $A^T A$.

Theorem 9 *Suppose $A \in R^{m \times n}$, $B = A^T A$, and the set $P \subset P_n$ has the property that for any $1 \leq i, j, k \leq n$, with $i < j$ and $i < k$, the condition $(i,j) \in P$ and $(i,k) \notin P$ implies that $(j,k) \in P$. If R and U are the upper triangular matrices that arise from the IMGS on A and IC on B using the set P respectively, then $R = U$.*

Proof: By induction on n. When $k = 2$, it is trivial to show that the assertion is true. Assume that for $k < n$ the assertion is true. For $k = n$, applying one step of IMGS to A, we get $A = A^{(1)}R^{(1)}$ such that $A^{(1)} = \left(a_1^{(1)},\ a_2^{(1)},\ \ldots,\ a_n^{(1)}\right)$ and

$$R^{(1)} = \begin{pmatrix} r_{11} & r_{12} \ldots r_{1n} \\ & \\ 0 & I \\ & \end{pmatrix}.$$

The rest of the steps of IMGS can be viewed as applying IMGS on the reduced matrix $\tilde{A} = \left(a_2^{(1)}\ \ldots\ a_n^{(1)}\right)$. Let R_2 be the $(n-1) \times (n-1)$ upper triangular matrix that arises from IMGS on the reduced matrix, then

$$R = \begin{pmatrix} r_{11} & r_{12} \ldots r_{1n} \\ & \\ 0 & R_2 \\ & \end{pmatrix}.$$

On the other hand, by applying one step of IC on $B = A^T A$, we get $B = {U^{(1)}}^T B^{(1)} U^{(1)}$, where

$$U^{(1)} = \begin{pmatrix} u_{11} & u_{12} \ldots u_{1n} \\ & \\ 0 & I \\ & \end{pmatrix}.$$

The rest of the steps of IC could be viewed as applying IC on the reduced matrix

$$\tilde{B} = \begin{pmatrix} b_{22}^{(1)} & b_{23}^{(1)} & \ldots & b_{2n}^{(1)} \\ b_{32}^{(1)} & b_{33}^{(1)} & \ldots & b_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n2}^{(1)} & b_{n3}^{(1)} & \ldots & b_{nn}^{(1)} \end{pmatrix}.$$

Let U_2 be the $(n-1) \times (n-1)$ upper triangular matrix that arises from IC on reduced matrix $\tilde{B}$, then

$$U = \begin{pmatrix} u_{11} & u_{12} \ldots u_{1n} \\ & \\ 0 & U_2 \\ & \end{pmatrix}.$$

Lemma 6 implies $R^{(1)} = U^{(1)}$. We now show that $R_2 = U_2$. Compare the elements of $\tilde{A}^T \tilde{A} = (a_{ij}^{(1)})$ and $\tilde{B} = (b_{ij}^{(1)})$ in position $(i, j) \notin P$. If either both $(1, i)$ and $(1, j)$ are in P or both $(1, i)$ and $(1, j)$ are not in P, we have $a_{ij}^{(1)} = b_{ij}^{(1)}$ due to the Lemma 6. In other words, $a_{ij}^{(1)} \neq b_{ij}^{(1)}$ is possible only in positions (i,j) such that either $(1,i) \in P$ and $(1,j) \notin P$ or $(1,i) \notin P$ and $(1,j) \in P$. However, due to the assumed property of P, we have $(i,j) \in P$. Therefore, $a_{ij}^{(1)} \neq b_{ij}^{(1)}$ can only be true if $(i,j) \in P$. This is equivalent to saying that $\tilde{A}^T\tilde{A}$ and $\tilde{B}$ satisfy the condition of Lemma 5 and it follows that $U_2 = \tilde{U}_2$ where $\tilde{U}_2$ and U_2 are $(n-1) \times (n-1)$ upper triangular matrices that arise from IC applied to $\tilde{A}^T\tilde{A}$ and $\tilde{B}$ respectively. We also have, according to the inductive assumption, $\tilde{U}_2 = R_2$, where R_2 arises from applying IMGS to $\tilde{A}$. These two conditions taken together imply $U_2 = R_2$ completing the desired result for the $k = n$ case. By induction, the assertion is true for all n. □

Theorem 9 establishes a connection between IC and IMGS. From this connection we can derive the following result regarding IC applied to an arbitrary symmetric positive definite matrix.

Theorem 10 *Let $B \in R^{n\times n}$ be a symmetric positive definite matrix. If $P \subset P_n$ has the property that $(i,j) \in P$ and $(i,k) \notin P$ implies $(j,k) \in P$ for all $1 \leq i \leq n$, then the IC factorization algorithm completes successfully.*

Proof: Let $B \in R^{n\times n}$ be a symmetric positive definite matrix. There exists a matrix $A \in R^{m\times n}$, $m \geq n$ with full rank such that $B = A^TA$. According to Theorem 1, IMGS applied to A completes successfully. So we have an upper triangular matrix R with positive diagonal elements such that $A = QR$. By Theorem 9, we get that for such a set P that satisfies the assumed property, IC applied to $B = A^TA$ generates the same upper triangular matrix as IMGS So IC completes successfully and generates an upper triangular matrix L^T which is equal to R. □

This result allows us, under certain restrictions, to view IC as a member of the class of incomplete Gram-Schmidt factorization. Chapter 6 discusses in more detail this property of the set P which relates IMGS to IC. Based on this property, three algorithms for the modifying the set P to insure the existence of an IC factorization.

3.4 Summary

In this chapter we have introduced a family of preconditioners for the least squares problem based on the notion of incomplete orthogonalization. The family includes the complete factorizations of modified and classical Gram-Schmidt as well as the Cholesky factorization of the corresponding normal equations. Also included in the family are the incomplete Cholesky factorization, column scaling and the two canonical methods incomplete modified and incomplete classical Gram-Schmidt. We have also derived some basic properties of the family of methods that are consistent with the results in the literature for the limiting members of the family Cholesky, IC, MGS and CGS. These results address the existence and numerical stability of the incomplete factorizations as well as the quality of the preconditioner under certain assumptions. The most important property of IMGS and ICGS is that they do not have the 'break down' problem when A has full rank and exact arithmetic is used. For finite precision, the error analysis shows that IMGS is less likely than MGS to break down due to the numerical rank deficiency. This result along with the guaranteed existence of IMGS and the lack of such a guarantee for IC, indicates the superior robustness of IMGS.

The other major problem addressed in this chapter is the relationship between IMGS and IC. When certain conditions are imposed on the drop set P, IMGS and IC will generate the same factor R. As a result, IC can be viewed as special member of the class of incomplete Gram-Schmidt factorizations and two promising directions for improving both IC and IMGS are indicated. The first direction concerns using the condition to improve the robustness of IC preconditioning. The second is motivated by the fact that when it exists and preconditions well IC is more efficient than IMGS in terms of both time and space required. As a result, the relationship can be used to derive ways to improve the efficiency of IMGS by making it look more like IC without compromising the existence of the incomplete factorization. Both of these directions are discussed in later chapters.

CHAPTER 4

Varieties of Incomplete Gram-Schmidt Orthogonalization

In the definitions and discussions of the family of incomplete Gram-Schmidt orthogonalization preconditioners above no restrictions were placed on the order in which the columns of A were orthogonalized (in the case of IC this corresponds to the order of elimination) or on the drop set P (except that it could not contain diagonal elements of R). Choices made with respect to these two aspects of the preconditioners yield different varieties of the basic algorithms of most interest: IMGS and ICGS. In this chapter, we first demonstrate that the incomplete Gram-Schmidt preconditioners can improve the convergence behavior of CGLS. We then examine the problem of pattern selection and column reordering of A and their effects on the performance of IMGS and ICGS. Before doing this, however, we describe, in Section 4.1, the environment used for the empirical testing in the remainder of the thesis and, in Section 4.2, the basic iterative algorithm to which the preconditioners are applied, CGLS.

4.1 Experimental environment

The experiments in this section and the remainder of the thesis use a suite of tests that include matrices from applications problems and from the RUA and RRA sets of the Harwell-Boeing collection. Characteristics of the matrices, including the number of rows (m), the number of columns (n), the number of non-zeros (nnz(A)), the number of the non-zeros in the normal equations (nnz(A^TA)), the density of the matrix and the density of the normal equations, are given in Table 4.1. The density of a matrix $B \in \Re^{m \times n}$ is denoted $dense(B)$ and expresses, as a percentage, the ratio of actual non-zero elements to the maximum possible, i.e., $100(nnz(B)/mn)$. The collection includes 30 matrices, of which 10 are square. The size of the matrices vary considerably both in terms of dimensions – $115 \leq m \leq 16640$ and $82 \leq n \leq 3564$ – and number of non-zero elements – $421 \leq nnz(A) \leq 78,298$. The sparsity of the matrices ranges from less than 1% to slightly more than 7 %. As expected, both the sparsity and non-zero element count increase significantly for the normal equations associated with the test matrices

– sparsity reaches the neighborhood of 40 % for some matrices and one problem has almost 400,000 non-zero elements. Most are reasonably conditioned but there are a few which are ill conditioned. The matrices are grouped according to their sources. The first set, AMOCO1 to WELL1033, is from the RRA portion of the Harwell-Boeing collection. The matrix AMOCO1 is from a seismic tomography problem while BELLADIT and BELMEDT are from information retrieval problems. The group from CONEV8 to STRAT8, is from computational fluid dynamics problems where, for some algorithms, they are used to compute projections. The final group of rectangular matrices, BNL1 to WOODW, is from a collection of linear programming problems available on NETLIB. We have also included a group of square matrices from the RUA set of the Harwell-Boeing collection, FS_760_1 to STEAM2.

name	m	n	nnz(A)	nnz(A^TA)	dense(A)	dense(A^TA)
amoco1	1436	330	35210	27686	7.43	25.42
belladit	374	82	1343	2395	4.38	35.60
bellmedt	5831	1033	52012	372255	0.86	34.89
illc1850	1850	713	10608	5633	0.80	1.11
well1850	1850	713	10608	5633	0.80	1.11
well1033	1033	321	5765	2469	1.74	2.40
conev8	3362	484	15852	5135	0.97	2.19
dunes8	5414	771	25430	6998	0.61	1.18
strat8	16640	2205	78298	21757	0.21	0.45
bnl1	1576	632	9152	28005	0.92	7.01
fffff800	854	525	6235	10625	1.39	3.85
gen	1500	780	3276	5816	0.28	0.96
nzfri	3521	624	15903	8406	0.72	2.16
pilot4	1000	411	5145	6899	1.25	4.08
scsd6	1350	148	5666	2248	2.84	10.26
seba	1028	516	4874	52432	0.92	19.69
shell	1775	537	4900	2748	0.51	0.95
ship12s	2869	1042	8284	6388	0.28	0.59
standata	1075	360	3038	1833	0.79	1.41
woodw	8405	1099	37478	21525	0.41	1.78
fs_760_1	760	760	5976	13957	1.03	2.42
fs_760_3	760	760	5976	13957	1.03	2.42
gre_115	115	115	421	692	3.18	5.22
hwatt_2	1856	1856	11550	27445	0.34	0.80
mc_fe	765	765	24382	73254	4.17	12.52
orsreg_1	2205	2205	14133	24203	0.29	0.50
pde_9511	961	961	4681	6420	0.51	0.70
pores_2	1224	1224	9613	12723	0.64	0.85
saylr4	3564	3564	22316	38793	0.18	0.31
steam2	600	600	13760	20237	3.82	5.62

Table 4.1 Characteristics of test matrices

For the rectangular matrices the experiments comprise two major sets of problems defined by consistent and inconsistent right-hand side vectors. Both sets were also run for the square matrices for the sake of completeness.

For the first set of problems, we generate a right-hand side vector consistent with a solution vector whose components are all equal to 1. Table 4.2 contains the quantities of $\| b - Ax \|_2$, $\| b - Ax \|_2 / \| b \|_2$, $\| b \|_2$, $\| x - x^* \|_2$ and $\| x - x^* \|_2 / \| x^* \|_2$ for the solutions computed using SPARSPAK-B – a QR factorization via Givens rotations with a preordering of rows and columns of A to reduce fill-in.

name	$\| b - Ax \|_2$	$\| b \|_2$	$\frac{\|b-Ax\|_2}{\|b\|_2}$	$\| x - x* \|_2$	$\frac{\|x-x^*\|_2}{\|x^*\|_2}$
amoco1	4.7310e-13	3.6970e+02	1.2796e-15	3.1390e-12	1.7279e-13
belladit	1.4040e-13	1.5630e+02	8.9827e-16	3.0110e-14	3.3250e-15
bellmedt	4.9930e-12	9.6360e+02	5.1816e-15	9.6310e-13	2.9965e-14
illc1850	2.2070e-12	6.8090e+03	3.2412e-16	2.6570e-10	9.9505e-12
well1850	2.3620e-12	6.8030e+03	3.4719e-16	1.8020e-11	6.7485e-13
well1033	1.4640e-12	6.6110e+03	2.2144e-16	5.8660e-10	3.2740e-11
*conev8	3.9960e-18	4.1760e-18	9.5689e-01	2.2000e+01	1.0000e+00
dunes8	1.1260e-16	1.5250e-03	7.3836e-14	4.2580e-12	1.5334e-13
strat8	1.3830e-14	9.4830e-03	1.4583e-12	9.1340e-11	1.9451e-12
bnl1	2.4080e-12	3.0500e+03	7.8950e-16	2.3450e-11	9.3279e-13
*fffff800	7.0780e-11	2.4580e+05	2.8795e-16	3.8070e+00	1.6615e-01
gen	5.8240e-14	1.1260e+02	5.1722e-16	5.4550e-12	1.9532e-13
*nzfri	1.3050e-09	3.3500e+05	3.8955e-15	1.6330e+00	6.5372e-02
pilot4	3.3020e-10	7.9130e+04	4.1728e-15	1.7390e-10	8.5778e-12
scsd6	1.1370e-12	8.8600e+01	1.2832e-14	2.9090e-13	2.3911e-14
*seba	3.6290e-12	4.7720e+03	7.6047e-16	1.0000e+00	4.4022e-02
*shell	1.3580e-10	1.1920e+04	1.1392e-14	2.3110e+01	9.9727e-01
ship12s	4.6770e-14	3.6730e+01	1.2733e-15	2.5130e-12	7.7850e-14
*standata	2.0490e-12	1.2010e+03	1.7060e-15	4.7660e+00	2.5119e-01
woodw	4.4370e-09	5.7990e+04	7.6513e-14	1.5340e-09	4.6272e-11
fs_760_1	2.2370e-07	4.5360e+08	4.9316e-16	1.7580e-12	6.3769e-14
*fs_760_3	6.8880e-06	4.5360e+08	1.5185e-14	1.6190e+01	5.8727e-01
gre_115	3.7850e-15	1.0970e+01	3.4503e-16	2.2770e-14	2.1233e-15
hwatt_2	1.8260e-14	8.0000e+00	2.2825e-15	2.6860e-12	6.2347e-14
mc_fe	1.1080e+02	3.9550e+13	2.8015e-12	3.4060e-11	1.2314e-12
orsreg_1	1.1840e-09	2.2700e+02	5.2158e-12	1.5290e-11	3.2561e-13
pde_9511	9.3220e-14	1.2140e+01	7.6787e-15	1.2840e-13	4.1419e-15
pores_2	3.2540e-08	2.5250e+07	1.2887e-15	1.9620e-10	5.6080e-12
saylr4	6.0240e-10	1.8020e-01	3.3429e-09	9.0620e-09	1.5179e-10
steam2	7.8540e-05	5.2660e+10	1.4914e-15	8.1450e-09	3.3251e-10

Table 4.2 SPARSPAK-B solutions of the consistent problems

Most of the matrices have a solution, x, with small relative error. A set of 7 matrices, however, show some difficulties in obtaining accurate solutions (the set is marked with *). Of this set, 2 matrices are known to be very ill conditioned. In exact arithmetic all of the row sums of the matrix **conev8** are 0 and, therefore, $b = 0$ for $x = 1$. The value for $\| b \|_2$ clearly shows numerical rank deficiency. The matrix **fs_760_3** is very ill conditioned with a maximum eigenvalue magnitude of 10^9 and a minimum eigenvalue magnitude of 10^{-9} [62]. The remaining 5 marked matrices are moderately ill conditioned. Of course, the fact that neither SPARSPAK-B nor the incomplete Gram-Schmidt preconditioners perform column pivoting for stability reasons also contributes to the lack of accuracy of the solutions. Adding this capability to IMGS or ICGS is a topic for future investigation. Note that SPARSPAK-B was able to produce small relative residuals for all of the matrices except CONEV8. So, with one exception, even the moderately and very ill conditioned problems are solved well from the point of view of minimizing the residual.

For the inconsistent problems, we randomly generate a right-hand side vector whose components are less than 1 in absolute value. Table 4.3 contains the norm of the residuals, the norm of right-hand side vector b, and the relative residuals for the least squares problems. For the rectangular problems, the relative residual norms show that the random right-hand sides do indeed generate inconsistent problems. Since these residual norms are, by definition, nontrivial for inconsistent problems we also include in the table the absolute and relative residuals for the associated normal equations. These must approach 0 for an acceptable solution to the least squares problem. Small relative residuals for the normal equations were achieved for all of the matrices except for the the square and ill conditioned FS_760_3.

In the remainder of the thesis, we assess the accuracy of the solutions produced by the iterative methods by comparing their relative residuals (or absolute residuals for those problems with small $\| b \|_2$, e.g., the consistent problems of DUNES8, STRAT8 and SAYLR4) to those produced by SPARSPAK-B. For the consistent problems, we also examine the relative error in x keeping in mind the poor conditioning of the 7 marked matrices. For the inconsistent problems, we have the additional comparison of the relative residuals for the associated normal equations produced by the iterative method solution to those listed in Table 4.3 for the SPARSPAK-B solutions. The solution can also be compared to that produced by SPARSPAK-B. But, since both are approximations and there is no guarantee that the solution by SPARSPAK-B is more accurate this test must be interpreted carefully.

All of the codes for the varieties of incomplete Gram-Schmidt orthogonalization and incomplete Cholesky factorization used in this thesis were implemented in standard Fortran. The packages SPARSPAK-A used for Cholesky factorization and SPARSPAK-B are also in Fortran but benefit from the more careful consideration typical of a numerical software package. All tests were run on a single processor of an Alliant FX/2800. In addition, to the algebraic quantities mentioned above, floating point operation counts and the cpu time were also collected in the experiments and are used to assess the efficiency of the various approaches.

name	$\| Ax - b \|_2$	$\| b \|_2$	$\frac{\|Ax-b\|_2}{\|b\|_2}$	$\| A^T(Ax - b) \|_2$	$\frac{\|A^T(Ax-b)\|_2}{\|A^Tb\|_2}$
amoco1	9.7860e+00	2.1830e+01	4.4828e-01	3.9910e-12	9.4753e-15
belladit	7.1920e+00	1.1060e+01	6.5027e-01	8.4600e-14	7.4933e-16
bellmedt	3.2480e+01	4.3800e+01	7.4155e-01	3.9540e-12	5.1632e-15
illc1850	9.8460e+00	2.4630e+01	3.9975e-01	3.3820e-10	4.4453e-15
well1850	9.8460e+00	2.4630e+01	3.9975e-01	2.6260e-09	3.4516e-14
well1033	7.8480e+00	1.8660e+01	4.2057e-01	6.2560e-10	1.0782e-14
conev8	2.1450e+01	3.3380e+01	6.4260e-01	2.5380e-17	7.8868e-14
dunes8	2.4170e+01	4.2160e+01	5.7329e-01	9.9590e-16	2.5477e-13
strat8	3.9540e+01	7.4270e+01	5.3238e-01	3.3350e-12	9.6470e-12
bnl1	1.3510e+01	2.2900e+01	5.8995e-01	1.8440e-10	1.2020e-13
fffff800	5.6850e+00	1.6970e+01	3.3500e-01	2.8950e-06	1.8921e-11
gen	1.6410e+01	2.2360e+01	7.3389e-01	1.5440e-12	1.3798e-14
nzfri	1.7750e+01	3.4200e+01	5.1900e-01	1.6660e-09	4.0486e-16
pilot4	1.2640e+01	1.8370e+01	6.8807e-01	1.9950e-10	2.5798e-15
scsd6	1.2500e+01	2.1180e+01	5.9017e-01	8.6440e-12	5.8603e-15
seba	9.0720e+00	1.8630e+01	4.8695e-01	1.1830e-10	3.9341e-14
shell	1.3470e+01	2.4130e+01	5.5822e-01	1.2120e-10	9.6727e-16
ship12s	1.9460e+01	3.0790e+01	6.3202e-01	2.7400e-11	1.5063e-12
standata	8.5670e+00	1.9e+01	4.5089e-01	6.7300e-09	1.0861e-12
woodw	2.4780e+01	5.2710e+01	4.7011e-01	2.9410e-08	4.8716e-14
fs_760_1	1.3710e-14	1.5780e+01	8.6882e-16	2.3870e-07	1.6553e-15
fs_760_3	1.4530e+01	1.5780e+01	9.2078e-01	4.8170e+06	3.3404e-02
gre_115	3.9290e-15	5.9140e+00	6.6435e-16	2.0360e-15	3.6461e-16
hwatt_2	1.5930e-04	2.4670e+01	6.4572e-06	5.1980e-04	1.7183e-05
mc_fe	6.2280e-11	1.5870e+01	3.9243e-12	3.3160e+06	2.8709e-11
orsreg_1	1.2080e-10	2.6970e+01	4.4790e-12	2.7400e-06	8.7539e-12
pde_9511	6.6450e-13	1.7930e+01	3.7060e-14	2.8130e-12	6.7734e-14
pores_2	2.5070e-08	2.0280e+01	1.2361e-09	1.0900e+00	2.7027e-08
saylr4	1.6880e-07	3.4390e+01	4.9084e-09	1.3330e-03	1.0422e-08
steam2	2.6650e-13	1.4090e+01	1.8914e-14	1.6390e-04	5.5259e-15

Table 4.3 SPARSPAK-B solution of the inconsistent problems

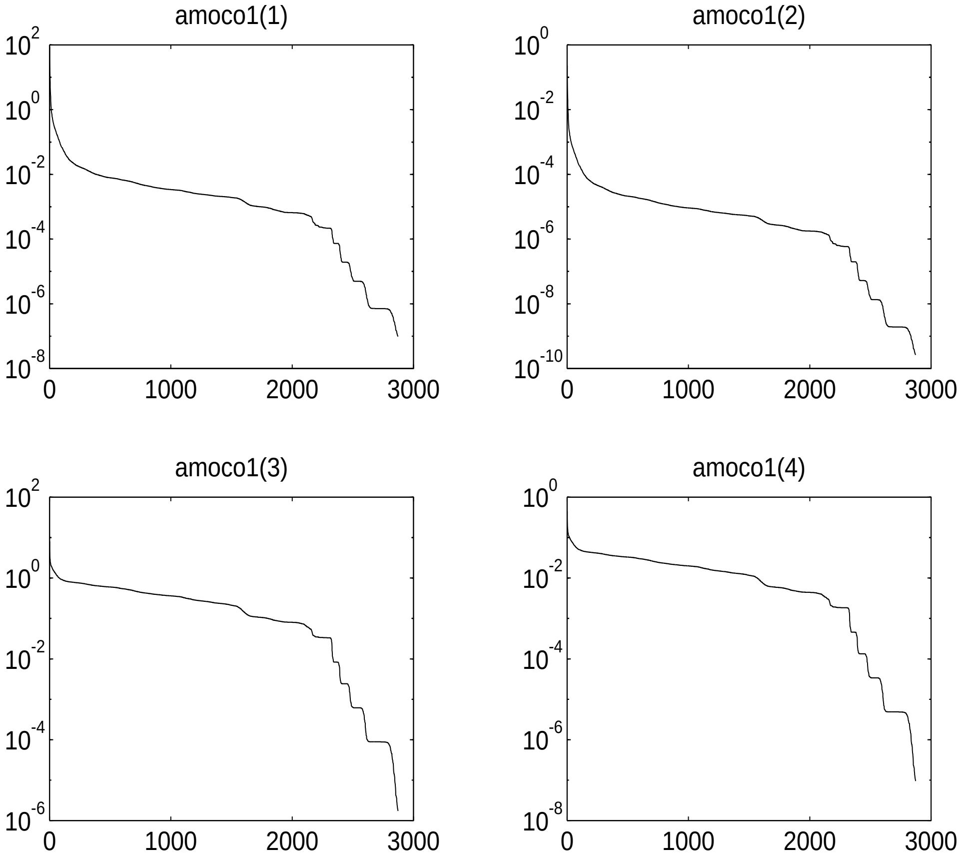
4.2 The basic iterative method: CGLS

Since the normal equations are symmetric positive definite when A is full rank, they can be solved using the conjugate gradient algorithm (CG). It is possible, however, to organize the computations of CG applied to the normal equations in such a way that A^TA need not be formed explicitly. The resulting method, CGLS [56], is used in this thesis as the basic iterative method to solve the least squares problems and, for the square matrices, the nonsymmetric linear systems. Of course, even though CGLS does not form the normal equations explicitly its convergence depends on their spectrum. We would, therefore, expect CGLS to have difficulty converging for problems that are not very well conditioned. Applying CGLS to the test suite confirms this expectation.

In this discussion, we restrict ourselves to the consistent test problems. The differences in performance and accuracy for the inconsistent test problems will be summarized at the end of the chapter. Since the test problems are consistent we can use $\| x - x^* \|_2 / \| x^* \|_2 \leq 10^{-6}$ as the condition that determines acceptable accuracy. Unfortunately, this proved to be a very difficult condition for CGLS to fulfill. Only 6 of the 30 test problems produced the desired accuracy within n iterations. After $2n$ iterations, a total of 9 matrices satisfy the requirement. Finally, after $2m$ only 2 more are added for a total of 11 out of 30 matrices. If the accuracy condition is altered to constrain the residual by requiring $\| b - Ax \|_2 \leq 10^{-6}$ or $\| b - Ax \|_2 / \| b \|_2 \leq 10^{-6}$, the situation improves somewhat. After n iterations, 10 matrices have satisfied the requirement. An additional 5 matrices have acceptable errors after $2n$ steps and after $2m$ iterations a total of 16 matrices out of 30 are solved satisfactorily.

Figures 4.1, 4.2 and 4.3, show typical examples of the behavior of CGLS over $2m$ iterations on a consistent problem. (Of course, the latter two also have more information.) The profiles consist of three basic types and most of the other experiments result in a variation on these. The solution of WOODW without preconditioning in Figure 4.3, is the worst type – an initial interval of rapid improvement (in this case disappointly short) followed by stagnation at an unacceptable accuracy. Figure 4.1 contains the same initial interval of rapid improvement and its distinguishing feature – a staircase effect with a general concave shape. That is, there is a series of bursts of improvement followed by temporary staganation. The bursts tend to have longer bouts of stagnation early producing a staircase profile that culminates in a cliff-like series of short stagnation intervals and rapid and increasingly significant improvement intervals. The other problems with this general shape tend to vary the relative sizes of the stagnation intervals and the slope of the improvement intervals while preserving general concave shape. The matrix BELLADIT produces the behavior in Figure 4.2 when solved without preconditioning and represents the third typical and most common profile observed in the tests. This is a form of the staircase profile above but without the exagerated flat stagnation and steep improvement intervals. Instead, there is a general improvement in accuracy with moderately sloped improvement intervals and relatively short stagnation intervals. Note also, that for each matrix the convergence profile of the residual and the solution are very similar in terms of overall profile. This is typical for all of the consistent experiments. Clearly, some form of preconditioning is required for CGLS to be a viable solution technique for these problems.

Since A^TA is a symmetric positive definite matrix and we are essentially solving it with CG, a natural preconditioner to attempt first is the IC factorization. Its use can improve the performance of CGLS dramatically. Figure 4.2 and 4.3 contain two examples which illustrate the improvement observed for the consistent problems. The matrix BELLADIT without preconditioning has a large region of steady but slow convergence before reaching its limiting accuracy. With IC preconditioning this point can be reached in 40 % of the unpreconditioned time. The matrix WOODW is an example that stagnates very early and remains so. IC preconditioning improves the accuracy from a few digits to many without increasing the

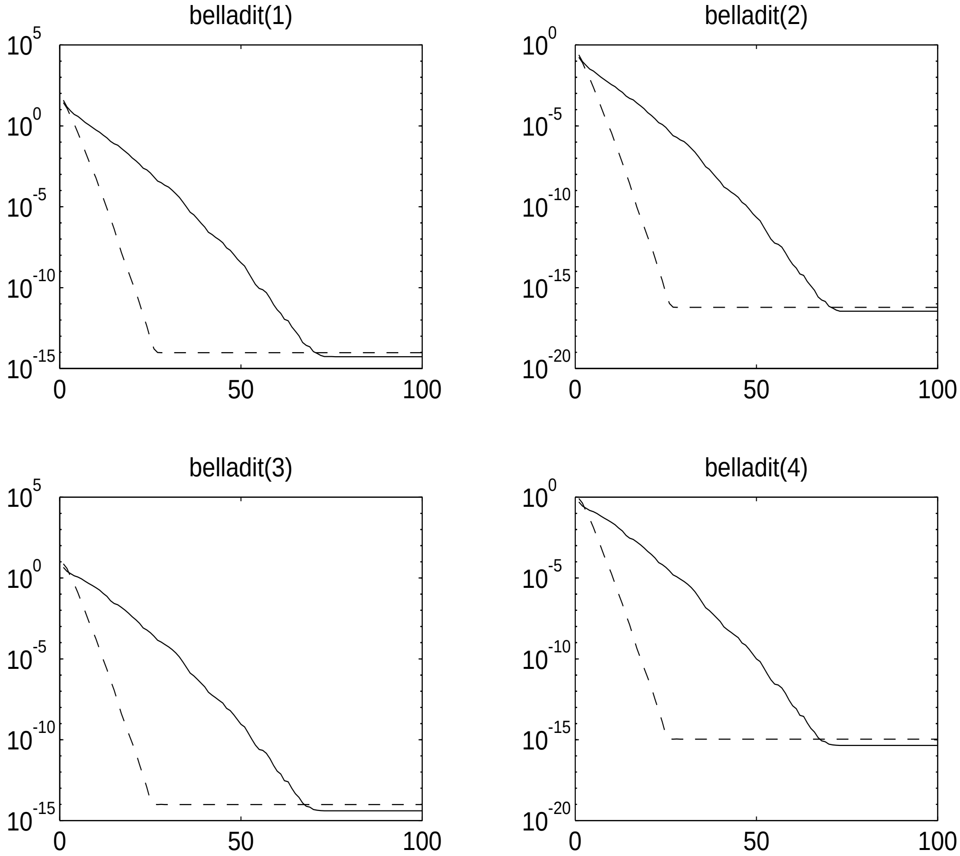


(1): $\| Ax - b \|_2$. $\quad (2)$: $\| Ax - b \|_2 \,/\, \| b \|_2$.
(3): $\| x - x^* \|_2$. $\quad (4)$: $\| x - x^* \|_2 \,/\, \| x^* \|_2$.

Figure 4.1 CGLS applied to consistent problem for AMOCO1

number of iterations needed to reach the limiting accuracy point. Unfortunately, IC preconditioning cannot be considered robust for the test suite. The IC factorization completes successfully for only 13 of the 30 matrices. This assumes the use of the standard pattern of non-zero elements kept in the IC factors, i.e., the non-zero pattern of the normal equations. We will see later how this can be improved.

For the remainder of this thesis we shall, in general, report results for consistent problems only. Inconsistent problem results will be mentioned only where they make a special point. This allows a reduction in the amount of data presented and is justified by our observation that, while the behavior of these methods vary across problems defined by different matrices, the qualitative behavior of a consistent and inconsistent problem for a given matrix is the same. So the conclusions we reach based on the data presented for the consistent problems are also supported by the data from inconsistent problems.

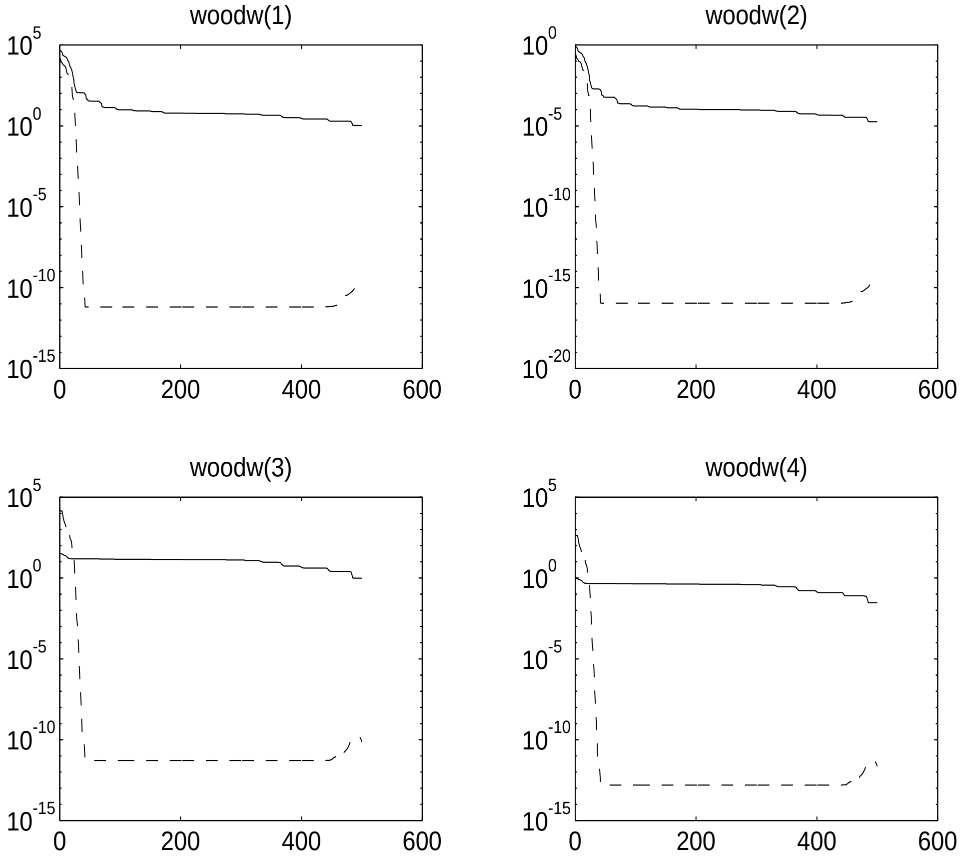


(1): $\| Ax - b \|_2$. (2): $\| Ax - b \|_2 \,/\, \| b \|_2$.
(3): $\| x - x^* \|_2$. (4): $\| x - x^* \|_2 \,/\, \| x^* \|_2$.

Figure 4.2 CGLS with and without preconditioning applied to consistent problem for BELLADIT

Figures 4.4, 4.5, 4.6, and 4.7 plot $\| A^T(Ax - b) \|_2$ for $2m$ iterations of the consistent and inconsistent test problems for four of the test matrices. For AMOCO1 and BELLMEDT the qualitative behavior is essential the same for consistent and inconsistent problems. Of course, the actual magnitude of the residuals differ but the trend to 0 is the same.

The matrix WELL1850 is less well conditioned than the previous matrices and its normal equation residual oscillates more but the behavior is similar for both the consistent and inconsistent problems – two slow regions of general convergence punctuated by a single spike at approximately the same iteration. The ill conditioned matrix ILLC1850 has much more variation in its residual but both consistent and inconsistent demonstrate the same behavior. The inconsistency does not mask any of the ill conditioning due to the random right-hand side vector.



(1): $\| Ax - b \|_2$. (2): $\| Ax - b \|_2 \,/\, \| b \|_2$.
(3): $\| x - x^* \|_2$. (4): $\| x - x^* \|_2 \,/\, \| x^* \|_2$.

Figure 4.3 CGLS with and without preconditioning applied to the consistent problem for WOODW

An examination of the error in x reveals similar trends. The convergence behavior of the the quantity of $\| x - x^* \|_2 \,/\, \| x^* \|_2$ for consistent problems and the quantity of $\| x - x_{QR} \|_2 \,/\, \| x_{QR} \|_2$ for inconsistent problems over $2m$ iterations show qualitatively the same behavior for a given matrix. Of course, the comparison of x to x_{QR} does not yield an error as small as x compared to x^* due to the fact that x_{QR} itself contains errors. But the convergence profile is similar and the resulting normal equation and least squares residuals are as good or better for the iterative method solution as for those of the QR solution. Figure 4.8 illustrates this for AMOCO1. The leftmost graph plots the absolute error in x compared to x^* for the consistent problem (solid line) and x_{QR} for the inconsistent problem (dashed line). The rightmost uses the same conventions but plots the relative error in x. Other than the shift in magnitude the trends are the same.

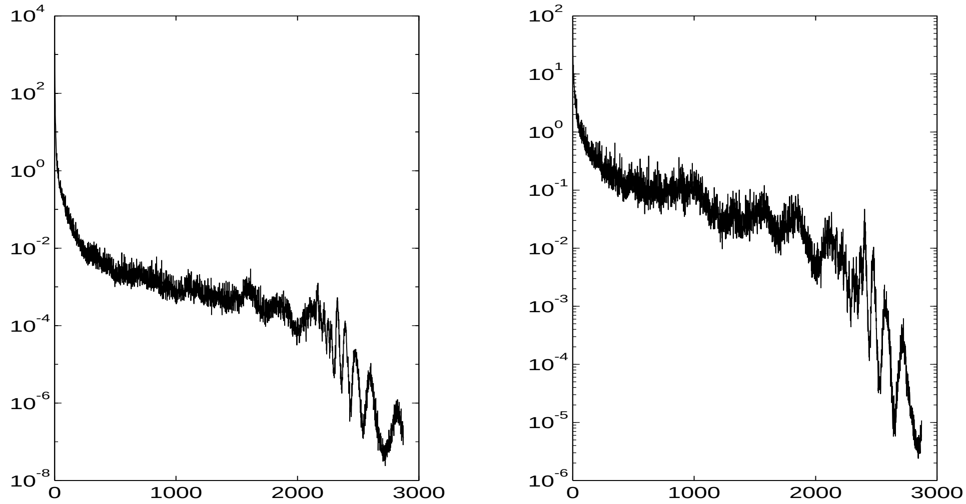


Figure 4.4 $\| A^T Ax - b \|_2$ for CGSL on consistent and inconsistent problems for AMOCO1

4.3 Varieties of pattern selection strategies

The IC preconditioner used earlier to demonstrate the benefits of preconditioning CGLS has a very straightforward definition of the drop set P – the complement of the positions of the non-zero elements in the normal equations. Not surprisingly, the choice of P can be crucial not only to the quality of the preconditioner but also to the combined efficiency of the two phases of the algorithm: preconditioner computation and iterative method. In this section, we consider the performance implications of the method of selecting P for IMGS and ICGS.

4.3.1 Dynamic pattern selection

One of the most complex ways of selecting P is to use a dynamic strategy. By *dynamic* we mean that the set P is unknown until the factorization is complete. Certain criteria are imposed on the value of a matrix element and at some point during the factorization a decision is made to keep or drop it based on the criteria (see [65] for a detailed discussion of this topic). Such a strategy has the advantage that it can adapt to numerical values encountered during the factorization and not just precondition based on its sparsity pattern.

In the case of IMGS and ICGS, after each step a test is made on the r_{ij} whose final values were produced. (Recall that it is possible to organize the computations so that partial results of several r_{ij} are computed in each step). The element is dropped if

$$| r_{ij} | \leq \epsilon\beta$$

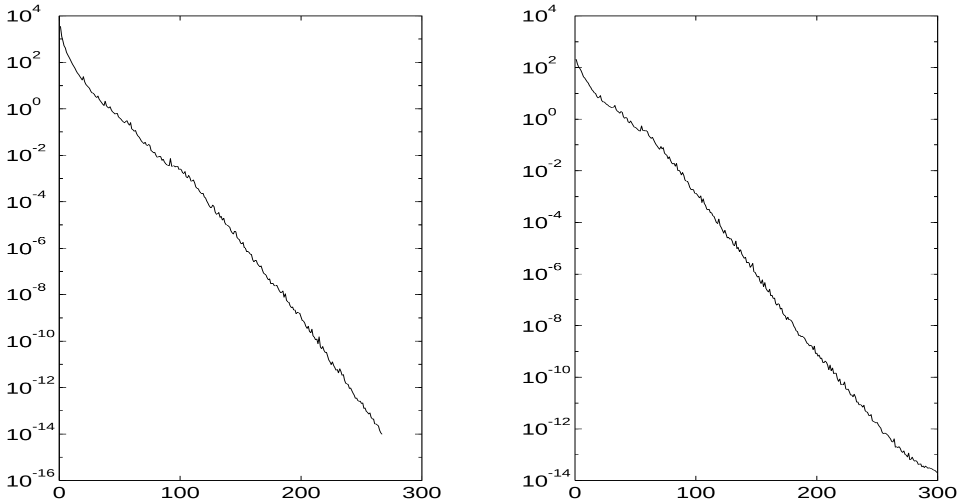


Figure 4.5 $\| A^T Ax - b \|_2$ for CGSL on consistent and inconsistent problems for BELLMEDT

where $0 \leq \epsilon \leq 1$ is a drop tolerance and β is some reference magnitude. For example, β can be updated on each step and is often taken to be $| r_{ii} |$ or the maximum $| r_{ij} |$ in a row or column of R. The use of a characteristic of all of the elements that are in a set to which dropping is applied to set β can cause difficulties with implementation and cost. An example of this is discussed below in Section 4.3.3. It could also be an estimate of $\| A^T A \|$ or an overall bound on the elements of the successive normal equations produced during the factorization. Such a bound can be derived for IMGS and ICGS. Since $A^T A$ is positive definite, the element of maximum magnitude is on the diagonal. One step of IMGS or ICGS preserves the positive definiteness of the normal equations (IC does not) and does not increase the magnitude of the diagonal elements. Therefore, we can bound magnitude of the elements of the sequence of normal equations produced during the factorization by 1 if the columns of A are normalized such that $\| a_i \|_2 = 1$.

This normalization and associated magnitude bound ($\beta = 1$) are used in our implementation of dynamic selection of P. Note that, qualitatively speaking, this is less conservative than the relative magnitude test with $\beta = | r_{ii} |$ since $| r_{ii} |$ tends to decrease from its initial value of 1. Of course, this does not imply that the sparsity is better preserved by the choice $\beta = 1$ since the magnitudes of the elements at any point in the factorization can be influenced in both directions by elements that may be kept by a tighter bound.

Varying ϵ, however, for a given matrix does significantly effect the sparsity and efficiency of the computations as well as the quality of the preconditioner. When $\epsilon = 1$, R is a diagonal matrix and ICGS and IMGS are equivalent to the scaling that normalizes the column vectors. When $\epsilon = 0$, ICGS and

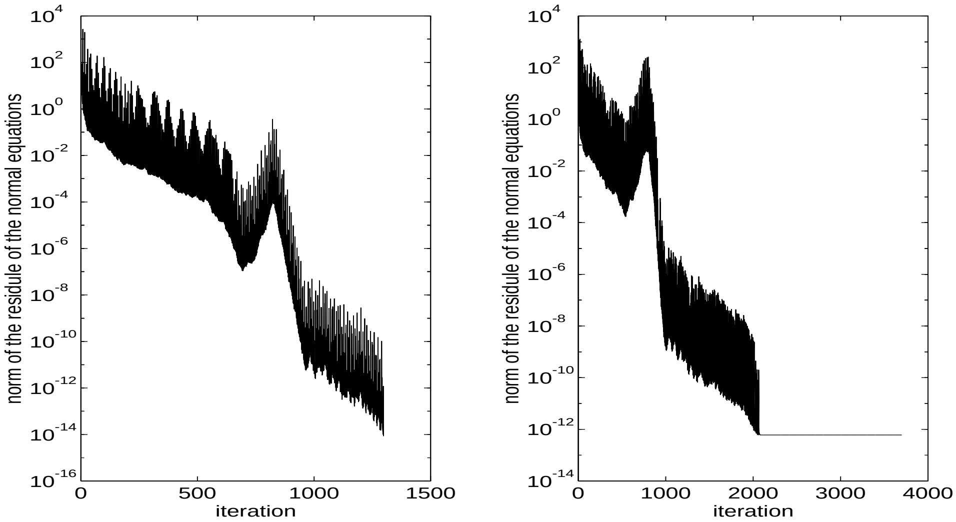


Figure 4.6 $\| A^T Ax - b \|_2$ for CGSL on consistent and inconsistent problems for WELL1850

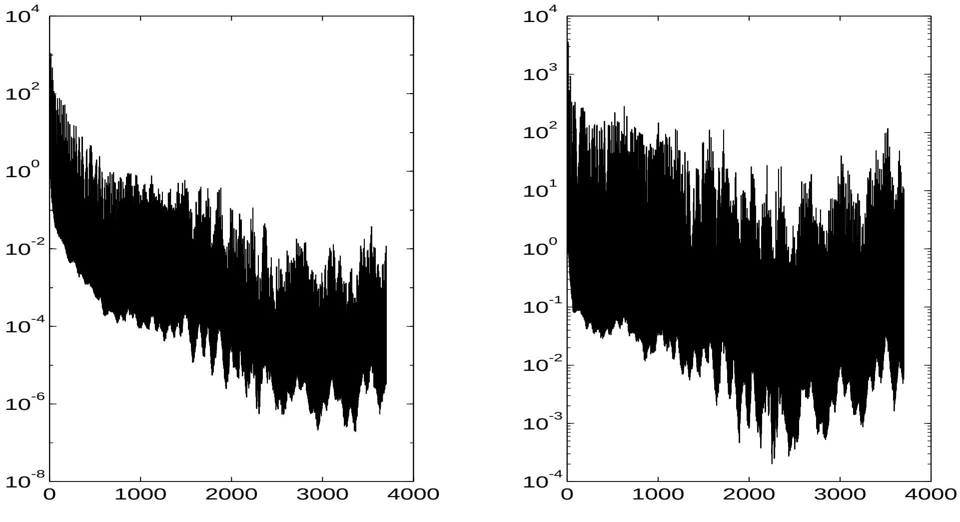
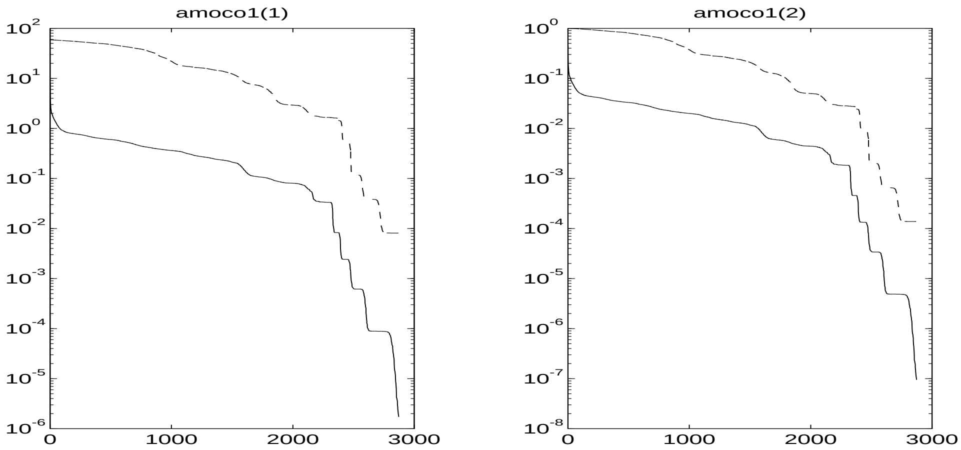


Figure 4.7 $\| A^T Ax - b \|_2$ for CGSL on consistent and inconsistent problems for ILLC1850



(1): absolute error   (2): relative error

Figure 4.8 Errors in x of CGLS applied to consistent and inconsistent problems for AMOCO1

IMGS are equivalent to CGS and MGS respectively. Figure 4.9 shows the variation in the number of non-zeros kept in R as ϵ is varied from one extreme to the other. We see that, for these two examples, the number of non-zeros of R is most sensitive when ϵ is small, $0 \leq \epsilon \leq 0.02$. When ϵ is larger, for example when $0.02 \leq \epsilon \leq 1$, the number of non-zeros is close to the minimum value of n and approaches it very slowly. The rapid decay early in the interval tends to be slowed if $\beta = | r_{ii} |$ is used in the dropping test. The method proposed by Jennings and Ajiz [43] is an example of dynamic selection P on IMGS that has appeared in the literature.

The results of solving the test suite using dynamic pattern selection with $\epsilon = 0.02$ are in Tables A.1 and A.2 in the Appendix. The choice of ϵ is motivated by the observation above. However, since the number of non-zeros in R tends to be near its minimum for this value we can expect some difficulties with the quality of preconditioning across such a wide range of matrices. The results of choosing ϵ more carefully for the various matrices are presented in Chapter 7 and two examples are discussed at the end of this section.

Some general conclusions are apparent from the data. Both ICGS and IMGS successfully complete the computation of R using dynamic pattern selection for all of the test problems. As expected, the number of non-zeros in R differs for ICGS and IMGS for the same ϵ, but ICGS is tends to keep more non-zeros than IMGS (see Figure 4.10). For three examples, AMOCO1, SEBA and FFFFF800, ICGS keeps approximately twice as many or more elements than IMGS. For the few problems where the number of non-zeros kept by IMGS is larger the difference is small. Somewhat surprisingly, however, when ICGS does keep more elements it does not necessarily converge significantly faster than IMGS.

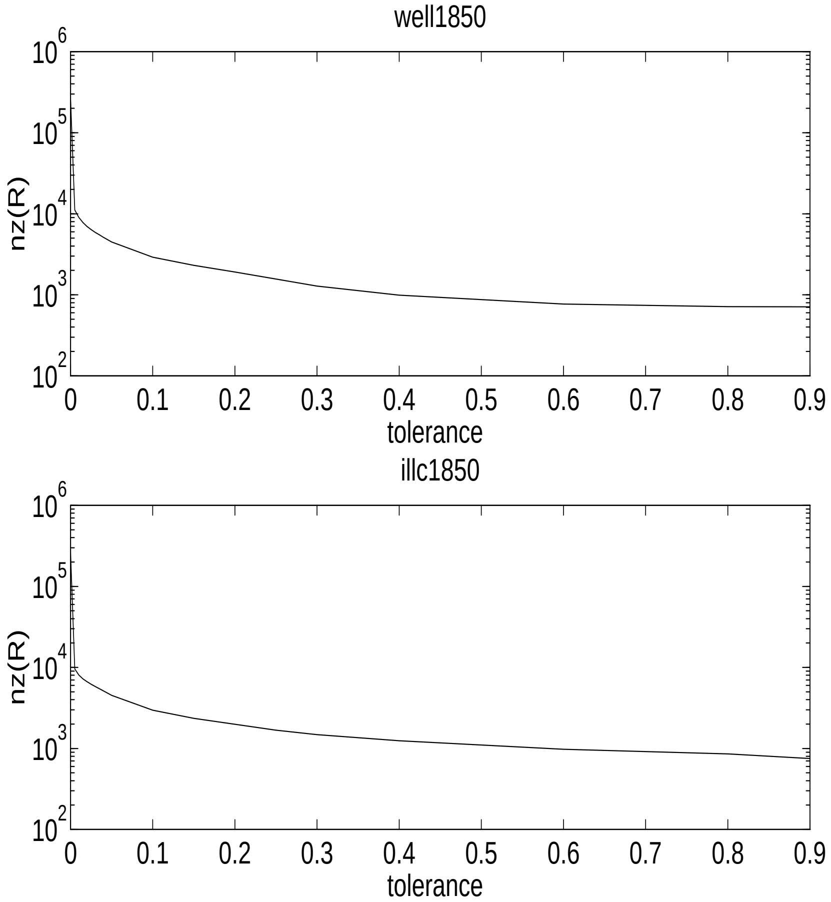


Figure 4.9 Number of non-zeros in R vs. ϵ

Both the ICGS and the IMGS preconditioners reduce greatly the number of iterations for CGLS on the test problems. The choice of ϵ, however, does cause some problems. ICGS does not converge to an acceptable level within $2n$ iterations for 6 of the 30 matrices and IMGS fails to do so for 4 matrices. So IMGS does have a slight advantage with respect to overall robustness for this ϵ. On the problems that converge for both preconditioners, the IMGS version requires fewer iterations, with one exception (NZFRI). Figure 4.11 shows the ratio of the number of iterations required when preconditioning by ICGS to that when preconditioning with IMGS with $\epsilon = 0.02$. Notice that IMGS does have an advantage, but for most of the problems it is only slight. The two large spikes are the matrices BNL1 and SEBA for which ICGS does not converge using $\epsilon = 0.02$. Both IMGS and ICGS preconditioning fail to cause convergence in $2n$ steps for the matrix PORES2 so it is not included in the figure.

Even though the quality of the preconditioner produced by IMGS does appear to be superior for this choice of ϵ its overall efficiency is worse than ICGS. The iterative method portion of the algorithm

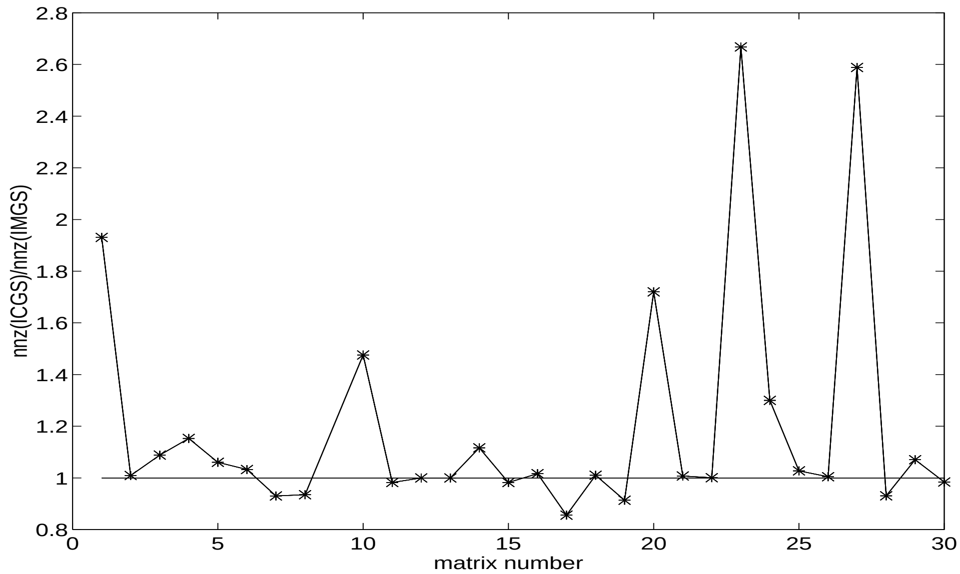


Figure 4.10 Ratio of the number of elements kept in R: $nnz(R_{ICGS})/nnz(R_{IMGS})$

performs similarly for the two methods assuming the dynamic dropping does not result in a large difference in the sparsity of the preconditioners. The major source of performance differences in this portion is the quality of the preconditioner.

The phase that computes the preconditioner is a completely different situation. When implementing ICGS, it is possible to organize the computations so that it performs fewer floating point computations than IMGS. This is done by exploiting an observation of Saad on the implementation of his ILQ preconditioner to which ICGS is related [58], (see Section 4.3.4). When computing q_j from a_j in ICGS, the computation of each of the r_{ij}, for $i = 1, \ldots, j-1$, involves a dot product of q_i and a_j. In IMGS a similar loop involves the dot product of q_i and $a_j^{(i-1)}$. Both loops require an update of a_j with a series of sparse vector triads. Saad pointed out, when considering loops equivalent to ICGS, that the q_i vectors are expected to be much denser than a_j and the other initial columns of A. As a result, the ICGS update should be driven by the sparsity pattern of a_j. This was a key aspect in the efficiency of the implementation of his ILQ preconditioner. The basic computation of IMGS does not have this property. The same q_i vectors are involved but the dotproducts require the $a_j^{(i)}$, $i = 1, \ldots, j-1$, a series of updated versions of a_j. These are denser than a_j and the form of operand reuse does not allow the same optimization opportunity. So despite the advantage in the quality of the preconditioner and the attendant reduction in iterations, IMGS typically has a larger cost in terms of operations. This can be mitigated in two circumstances: problems for which IMGS produces a much better preconditioner; and problems for which the same preconditioner can be computed once and then used for a series of right-hand side vectors.

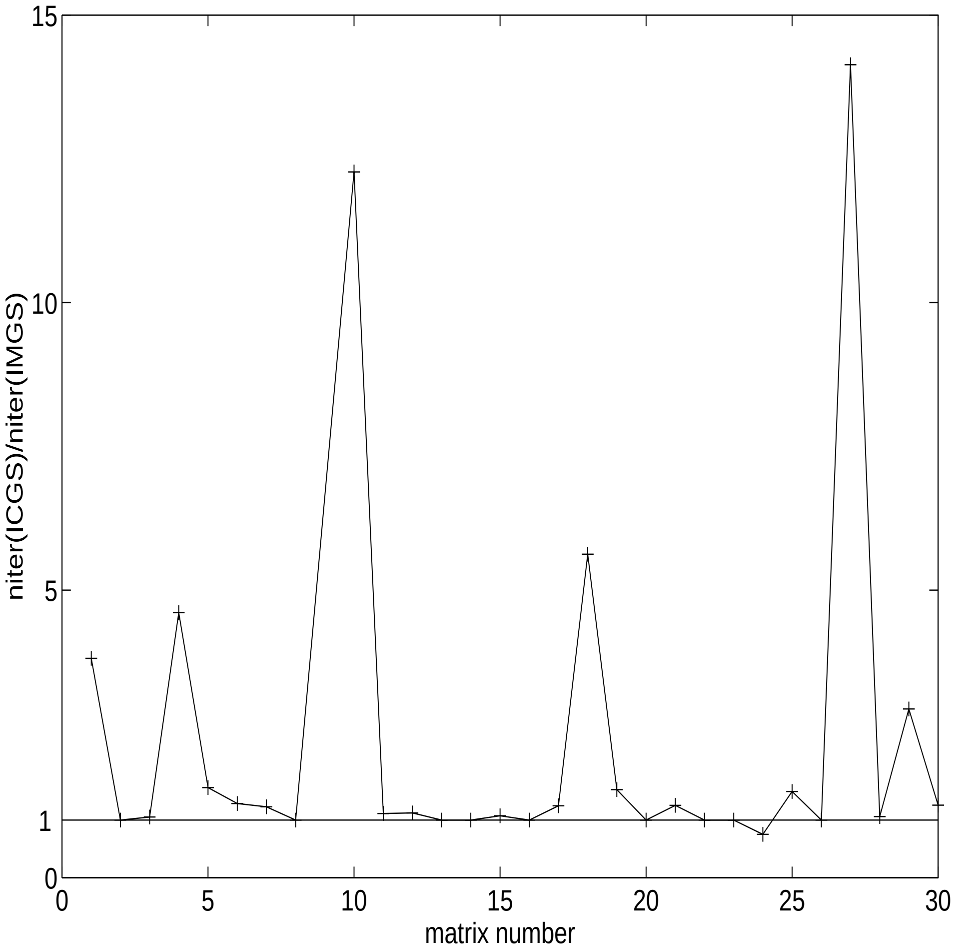


Figure 4.11 Ratio of iterations for ICGS to iterations for IMGS

Figure 4.12 summarizes the the relative operation counts of ICGS and IMGS for the test suite. In the leftmost graph, ratios of operations counts of ICGS to IMGS are plotted for the preconditioner computation phase (solid line) and the combined count for both phases (dashed line). While most of the ratios plotted earlier had ICGS dominating and therefore produced ratios above 1, this plot shows a striking reversal. The operation count to compute R_{ICGS} (which is directly proportional to computing time in our experiments) is well below that required for R_{IMGS} – most ratios are below 0.1. This translates directly into a similar advantage in the ratio of total operations to solve the problem. While the ratios are not as small due to the degradation in preconditioning quality for ICGS, its general superiority is clear. It is only for the 4 problems where IMGS produces a profoundly superior preconditioner that the ratio of total operations rises above 1. The rightmost graph plots the ratio of the time to compute the preconditioner to the total time for IMGS (solid line) and ICGS (dashed line). This clearly confirms the preconditioner as the source of the problem for IMGS. ICGS spends less than 30% of its time in the preconditioner computation while, for many problems, IMGS spends a very large fraction of its time there.

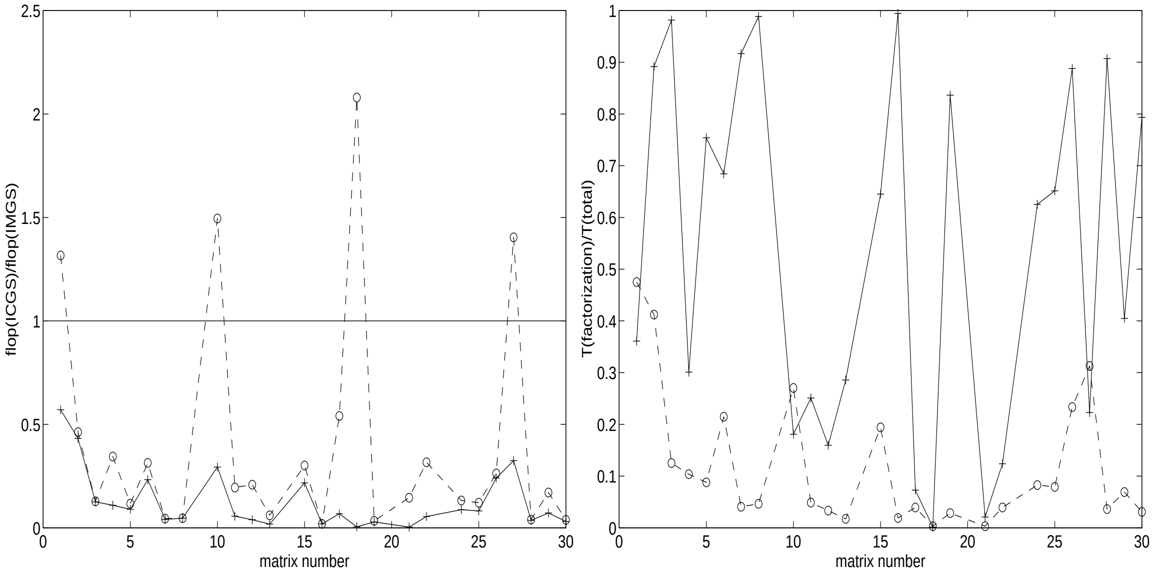


Figure 4.12 Operation count ratios and preconditioner computation time to total time ratio for ICGS and IMGS

The results so far indicate the effectiveness of IMGS and ICGS as a preconditioner for CGLS. They also highlight some of the complexities of the problem. Approaches to improving the efficiency of IMGS are discussed later in the thesis. For the remainder of this section we concentrate on issues related to the selection of the drop set P.

The main drawback with the dynamic approach is that there is no clear way of selecting ϵ. The proper value of ϵ varies from problem to problem, and its use is crucial. If ϵ is too small, R becomes too dense and the computational cost soars. On the other hand, if ϵ is too large then the number of dropped elements becomes large and the quality of the preconditioner may not be satisfactory.

Figure 4.13 shows an example of increasing ϵ to improve the performance IMGS. The trend in the number of non-zeros kept in R shows a significant drop as ϵ increases beyond the 0.02 used above. Even though there is a corresponding increase in the number of iterations, the reduction in the cost of the preconditioner (solid line in both lower graphs) more than outweighs the very slow increase in the cost of the iterative phase of the computations (dashed line in both lower graphs) to yield a total time and operation count (dash–dot line in the lower graphs) that decreases steadily until $\epsilon \approx 0.2$ beyond which it remains constant because the preconditioner is no longer changing. Also note that as the performance improves, the bottleneck of the preconditioner computation in IMGS is reduced substantially from the absurd levels when $\epsilon \leq 0.02$. So there is some hope that by manipulating ϵ in the case of dynamic selection and the drop set P in general, IMGS may be turned into a viable method that can exploit its superior preconditioning capability.

Unfortunately, it is not always clear which way to refine ϵ. Figure 4.14 shows an example for which ϵ must be substantially reduced from $\epsilon = 0.02$ to improve performance. The matrix PORES2 caused CGLS preconditioned by both ICGS and IMGS to fail. The figure shows the change in the various performance parameters of IMGS with ϵ changing from 0.0002 to 0.00001. We can see that as ϵ decreases the preconditioner improves dramatically in quality. At $\epsilon \approx 0.00005$ the minimum computation time is achieved.

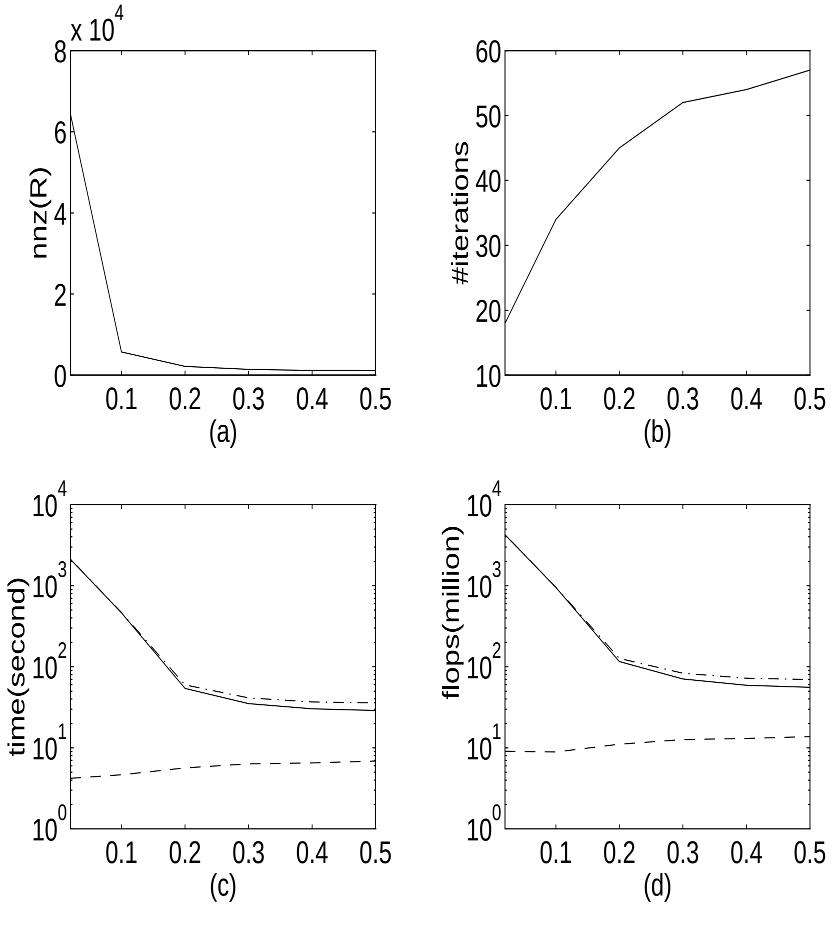


Figure 4.13 Performance vs. ϵ (BELLMEDT)

A difficulty common to all of the pattern selection strategies is that, since we are only dropping from the matrix R, it may not be the case that Q will be satisfactorily sparse. In Section 4.4 it is shown that careful reordering of the columns of A before computing the incomplete factorization can mitigate this difficulty considerably. It is possible, however, to have wide variation in the sparsity of Q if no mitigating

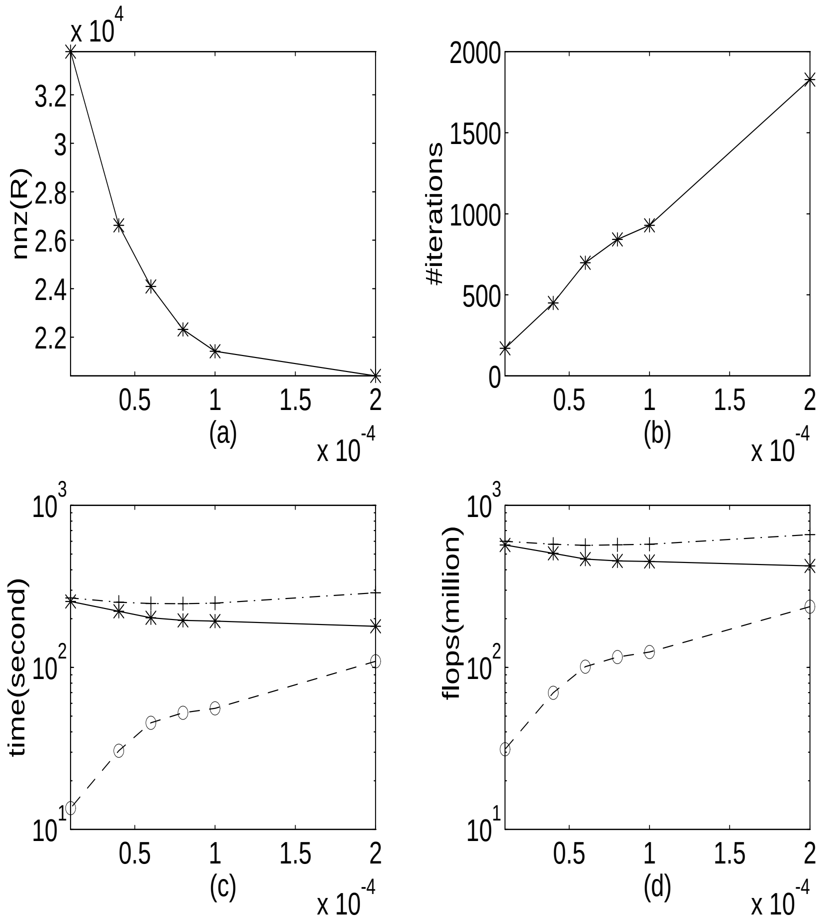


Figure 4.14 Performance vs. ϵ (PORES_2)

action is taken. Figure 4.15 plots the number of non-zeros in each column of A, Q_{IMGS}, and Q_{MGS} for the matrix ILLC1850 without column reordering using dynamic pattern selection with $\epsilon = 0.02$. For this matrix, the sparsity pattern of Q_{ICGS} is the same as that produced by IMGS. Two basic facts are clear for this example. First note that there is large increase in the density of the columns of Q_{MGS} compared to those of A and that this increase is more pronounced for those later in the orthogonalization sequence. Second, note that Q_{IMGS} does keep the density of the columns lower than MGS by a reasonable amount for all but a few of the later columns. The matrix BELLADIT demonstrates exactly the opposite behavior. Figure 4.16 plots the number of elements per column of the A, Q_{IMGS}, and Q_{MGS} and clearly shows that there is very little difference between MGS and IMGS.

All incomplete orthogonalization methods suffer due to increased work accompanying a lack of sparsity preservation in Q, but when dynamic pattern selection is used the method is particularly sensitive. This is due to the fact that at any step of the algorithm all non-zero elements in R that appear in

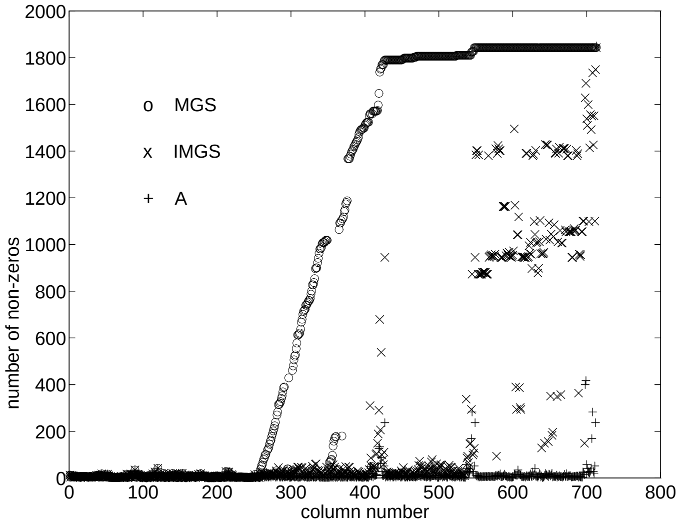


Figure 4.15 Number of the elements per column of Q from IMGS and MGS and A (ILLC1850)

positions that are non-zero in the corresponding column or row of the *complete* Cholesky factor L^T must be evaluated and then checked for dropping. When Q stays sparse the computation of the elements of R that are dropped is not too costly but if Q_{IMGS} approaches Q_{MGS} in density the cost of IMGS approaches that of MGS. The use of a static pattern can mitigate this by keeping the number of evaluations of elements in R low at the potential cost of degrading the quality of the preconditioner. This is investigated in the following section.

4.3.2 Static pattern selection

In contrast to the dynamic pattern selection strategy discussed above, *static* pattern selection determines the positions in the drop set P before the factorization starts. This specifies exactly which elements of R must be evaluated in each step of IMGS and ICGS and makes the dynamic nature of the data structure holding the evolving Q much simpler. In general, as mentioned above, this also minimizes the number of computations that must be performed only to decide that they did not have to be performed based on dynamic dropping. A static strategy also allows us to arrange the computations in a favorable way. For example, we can choose the pattern of R to be of some structure that preferable for the parallel processing such as banded or bordered block diagonal, etc.

The disadvantage of a static strategy is that with the lack of knowledge of elements of R we may not be able to produce a good preconditioner. Before the factorization, we only have knowledge of the values and positions of the elements of A. In this section we investigate the performance of three static pattern selection techniques.

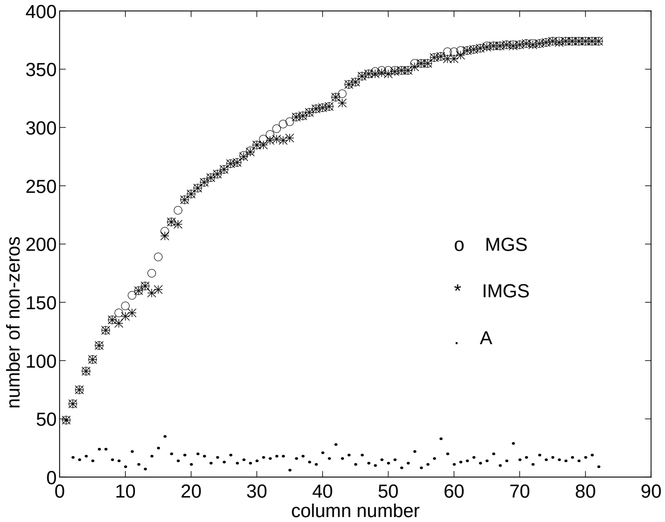
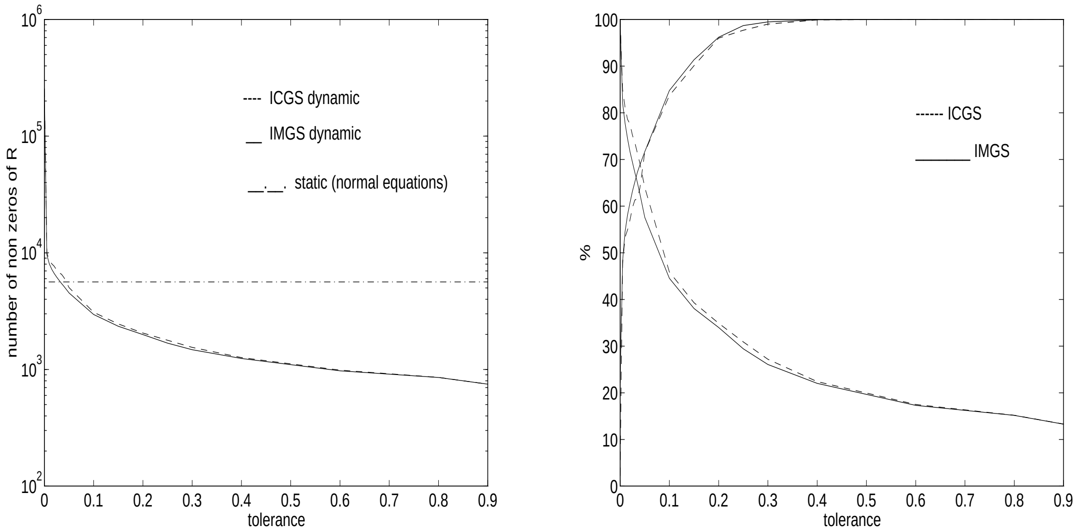


Figure 4.16 Number of the elements per column of Q from IMGS and MGS and A (BELLADIT)

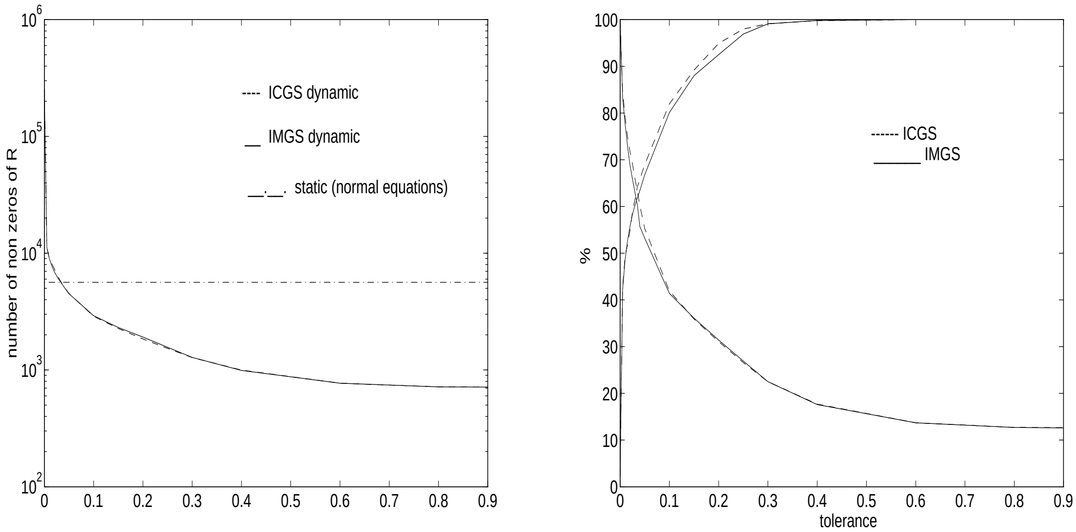
The most natural set of non-zero positions to specify for R (and therefore its complement for P) is the non-zero positions of the normal equations. This is the same set that is used by IC, but when it is used for IMGS and ICGS the factorization is guaranteed to exist. We examined the quality of the pattern of normal equations by comparing the set of elements in the normal equations to that produced by the dynamic strategy described above. Let P_d and P_{ne} denote the set of positions selected by the dynamic strategy and the normal equations respectively. (Note that these are the sets of elements that are *kept* in R, not the drop sets.) The values of $\frac{|P_d \cap P_{ne}|}{|P_d|}$ and $\frac{|P_d \cap P_{ne}|}{|P_{ne}|}$ as a function of ϵ illustrates the relationship between the two strategies. Figures 4.17 and 4.18 plot these functions for ILLC1850 and WELL1850 respectively. Both figures also include the number of elements in the static set, and the dynamic sets for ICGS and IMGS. The two matrices have the same non-zero pattern and differ in their conditioning (WELL1850 is much better conditioned).

The characteristics of the curves are similar despite the difference in conditioning. From the leftmost graph in the figures, we see that in the interval of ϵ where the non-zero count for the dynamic strategy changes rapidly, the count for the normal equations is a reasonable constant average of the two dynamic extremes. In the region, where the dynamic strategy becomes insensitive to ϵ we know that typically poor preconditioning results. So assessment must be deferred until the convergence behavior can be compared. The rightmost graph in the figures demonstrates that the normal equations set is usually a good compromise. When ϵ is small, $\epsilon < 0.02$ for example, the quantity $\frac{|P_d \cap P_{ne}|}{|P_d|}$ is over 80%. This means that over 80% of the positions of normal equations are also in the set that determined by dynamic strategy. Given that the dynamic strategy can be costly with ϵ due to increased density of Q the normal



Left: number of non-zeros vs.ϵ
Right:$\frac{|P_d \cap P_{ne}|}{|P_d|}$ and $\frac{|P_d \cap P|ne|}{|P_{ne}|}$ vs. ϵ.

Figure 4.17 Dynamic vs. normal equation pattern selection strategy for ILLC1850



Left: number of non-zeros vs.ϵ
Right:$\frac{|P_d \cap P_{ne}|}{|P_d|}$ and $\frac{|P_d \cap P|ne|}{|P_{ne}|}$ vs. ϵ.

Figure 4.18 Dynamic vs. normal equation pattern selection strategy for WELL1850 equations may simplify the computations and not degrade convergence excessively. When $\epsilon > 0.1$ the quantity $\frac{|P_d \cap P_{ne}|}{|P_{ne}|}$ is over 80%. This means that over 80% of elements selected by dynamic strategy are in the set of positions in the normal equations. In between , $0.02 \leq \epsilon \leq 0.1$ at least one of these quantities are over 60%. Based on these trends we can say that normal equations appear to be a reasonable candidate position set for ICGS and IMGS. If the preconditioning is not satisfactory the closeness to the dynamic dropping indicates that minor increases in the density of R can approach the set produced by the superior (but more costly) dynamic selection strategy.

The examples of the performance of ICGS and IMGS with static pattern of normal equations are shown in Table A.3 and A.4. From the results we see that ICGS and IMGS are robust in the sense that both factorizations are able to compute the factor R for all the test problems. Comparing the capability of accelerating convergence of CGLS, we find that both preconditioners are able to reduce greatly the number of iterations needed by CGLS. But the effectiveness of the methods in reducing the number of iterations differ. For example, problem SEBA converges to the required accuracy in 70 iterations using IMGS preconditioning but it cannot converge in $2n$ (1032) iterations using ICGS preconditioning with the same pattern of R. Problem ORSREG_1 converges in 312 iterations with IMGS preconditioning and does not converge in $2n$ (4410) iterations using ICGS. For all the test problems, CGLS takes fewer iterations to converge using IMGS preconditioning than using ICGS preconditioning, with only one exception. For the matrix SHELL, CGLS preconditioned by IMGS takes 23 iterations while preconditioned by ICGS it takes 21 iterations, i.e., a trivial difference.

For those test problems that CGLS preconditioned by ICGS and IMGS both converge to the given accuracy within $2n$ iterations, Figure 4.19 shows the ratio of the number of iterations needed by CGLS preconditioned by ICGS to the number needed when IMGS is used. The profile is very similar to that seen for dynamic dropping indicating that IMGS is a superior preconditioner.

Figure 4.20 addresses the efficiency issue that caused us to favor ICGS for the dynamic dropping strategy in all but the divergent problems. It shows the ratio of the ICGS operation count that for IMGS for the preconditioner phase (solid line) and both phases (dashed line) in the leftmost graph. The rightmost graph plots the ratio of the preconditioning time to total time for IMGS (solid line) and ICGS (dashed line). While in general IMGS is still less efficient overall there has been a substantial change relative to ICGS. Note that IMGS still spends most of its time in the preconditioner for many problems, but its slowdown in the preconditioner phase compared to ICGS is much smaller than for the dynamic selection strategy. Its overall efficiency has also improved relative to ICGS. Therefore, there is still hope that the efficiency of IMGS can be improved while keeping its superior effectiveness as a preconditioner.

Next, we compare in more detail the performance of dynamic pattern selection with static pattern selection. Comparing the data from Table A.1 with the data from Table A.3, we see that using ICGS with a static pattern does not greatly reduce the cost of the factorization indicating that ICGS exploits

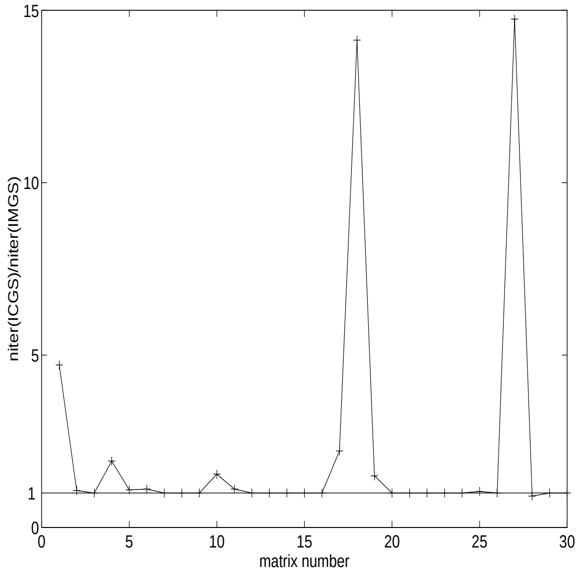


Figure 4.19 Ratio of iterations for ICGS to iterations for IMGS preconditioning with static pattern of normal equations

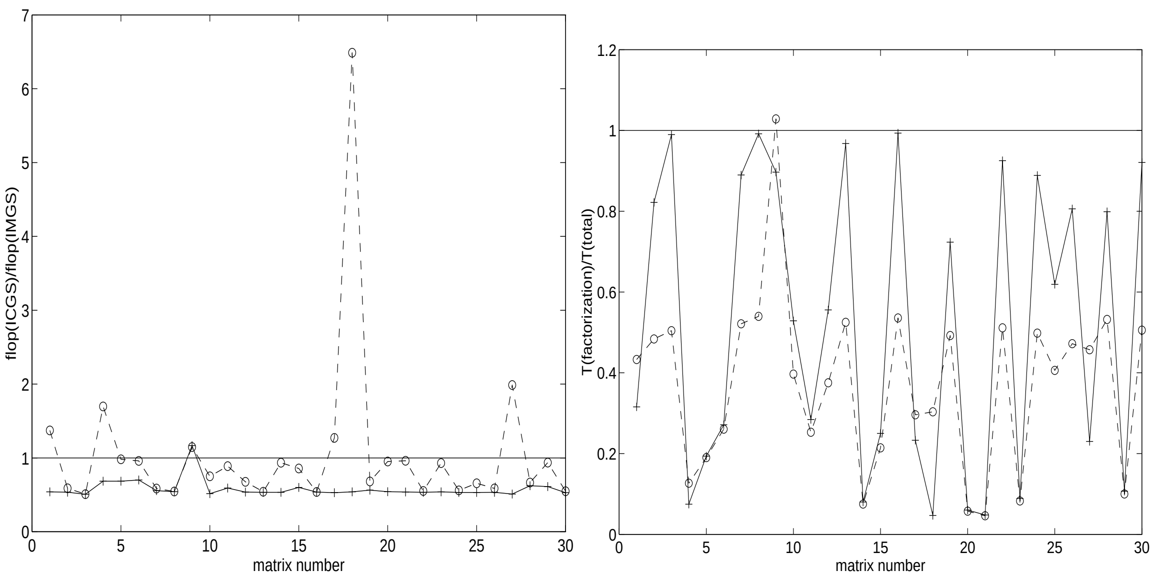


Figure 4.20 Performance comparison for ICGS vs. IMGS with static pattern selection

the available sparsity well already. Comparing the convergence behavior, we can find good cases for both dynamic and static patterns. For example, the problems NZFRI and BNL1 favor a static pattern while problems SAYLR4 and WOODW favor a dynamic pattern.

Comparing Table A.4 with Table A.2 we find that when using a static pattern the computation cost for the IMGS factorization is greatly reduced. For example, in the problems WOODW, AMOCO1 and some others, the number of non-zeros in R is much greater using static pattern than using dynamic pattern with $\epsilon = 0.02$, but the total number of operations using the static pattern is less than when using the dynamic pattern.

For the matrix WOODW, the density of R using the static pattern is roughly twice of the density from using dynamic pattern. The increased density does not help convergence and the static requires about 25% more iterations. Nevertheless, the total cost for the solution of the problem is only about 22% of the cost when using dynamic pattern. This is due to the time it takes to compute the preconditioner. In general, this trend holds, i.e., IMGS using the static pattern of normal equations works better than dynamic pattern selection because of the lower computation cost on the IMGS factorization phase even if the preconditioner is less satisfactory.

For some problems, the time spent in the preconditioner computation completely dominates the total operation count and time. We now consider two more static selection strategies that can reduce further The matrices BELLMEDT and WOODW are used as examples. First, we add to the drop set P the positions where the magnitude of the elements of the normal equations is less than a given tolerance. Because we need to know the values of the elements of normal equation, we refer to this as the *numerical method.* This is based on the heuristic that if two columns of A, a_i and a_j, are almost orthogonal and $cos(a_i, a_j)$ is small, then we may not need to do anything to adjust their relationship. Therefore, we can choose the position set P as

$$P = \{(i,j)|cos(a_i, a_j) < \alpha\}$$

where $\alpha > 0$ is a given tolerance. Figure 4.21 shows the changes in the number of operations and the time as α goes from 0 (which is the normal equation method) to 1 for the two phases of the computation and their sum. Note that at some point in the interval the numerical method will drop all of the off-diagonal elements in R and the preconditioner is just column scaling. This point can be seen in the number of iterations curve where the curve becomes flat. The number of non-zero elements of R and the number of iterations are also plotted. Clearly, there is a tremendous benefit in taking $\alpha > 0$ and removing some of the elements based on initial angles. Fortunately, this sharp decrease tends to happen for those problems that need it most, i.e., the ones dominated by the preconditioner, and over estimating α does not degrade performance that much.

Of course, if the cost of evaluating the values of the normal equations is too high or if the structure of A is to be reused but not its values an alternative strategy is needed. The numerical method can

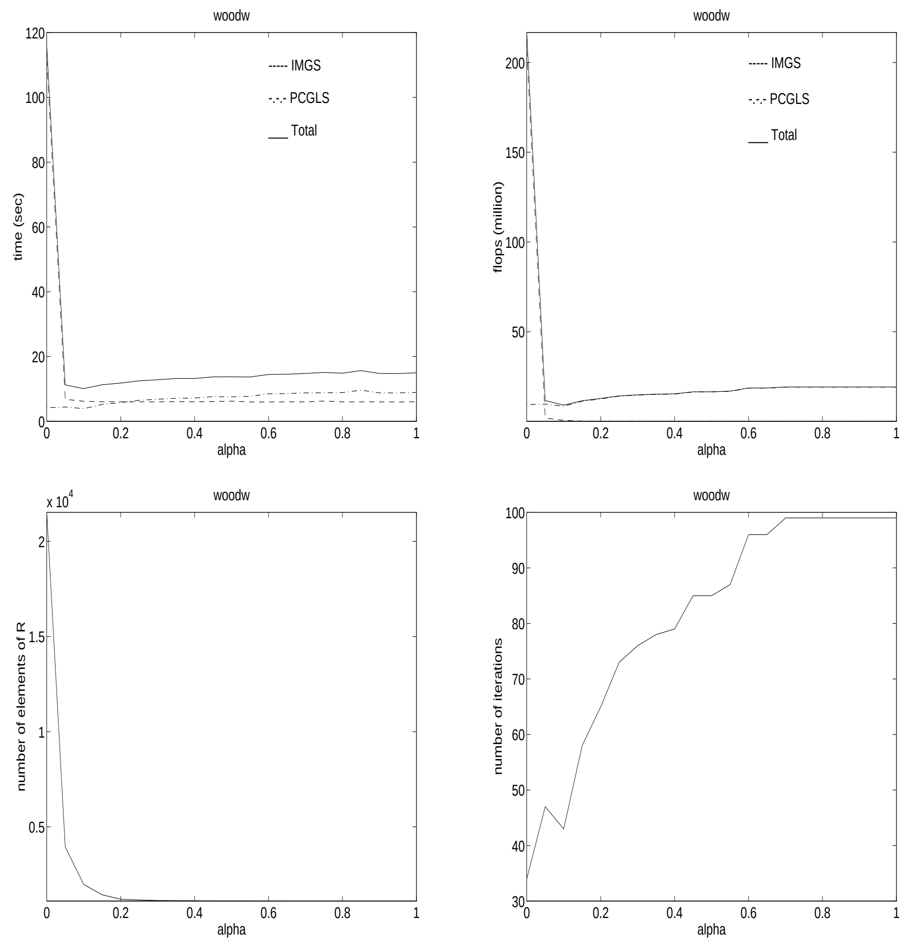


Figure 4.21 Example of numerical strategy of static pattern selection for WOODW

be easily altered to use nearness to structural orthogonality rather than numerical orthogonality. (Two vectors are said to be structural orthogonal if there is no common index of non-zero elements.) The resulting method is referred to as the *structural method.*

It sets the drop set to the following:

$$P = \{(i,j) | \frac{|I(a_i) \cap I(a_j)|}{|I(a_i) \cup I(a_j)|} < \alpha\}$$

where $I(a_k)$ is the set of row indexes of nonzero elements in column a_k. This condition is based on the following reasoning:

- If two columns a_i and a_j have a large number of common indices of non-zero elements relative to their total number of indices then the combination of these two columns will not introduce large amount of fill-in to Q. So it does not hurt to keep r_{ij}.

- If two columns a_i and a_j have a small number of common indices of non-zero elements and have large number of distinct indices of non-zero elements, then the combination of these two columns will introduce large amount of fill-in in Q. But since they are structurally close to orthogonal perhaps the preconditioner will not be affected significantly if r_{ij} is dropped.

Figure 4.22 shows the same information as a function of α for the structural strategy applied WOODW that was displayed for the numerical strategy above. The performance is very similar for this matrix. In general, the numerical and structural methods tend to behave in a similar fashion for those problems that were dominated by the preconditioner computation, i.e., ones that can tolerate some loss of preconditioning quality to save time. The structural method does not react as smoothly to its parameter, as expected, and of the two it does tend to be the one that will start to deteriorate in performance first.

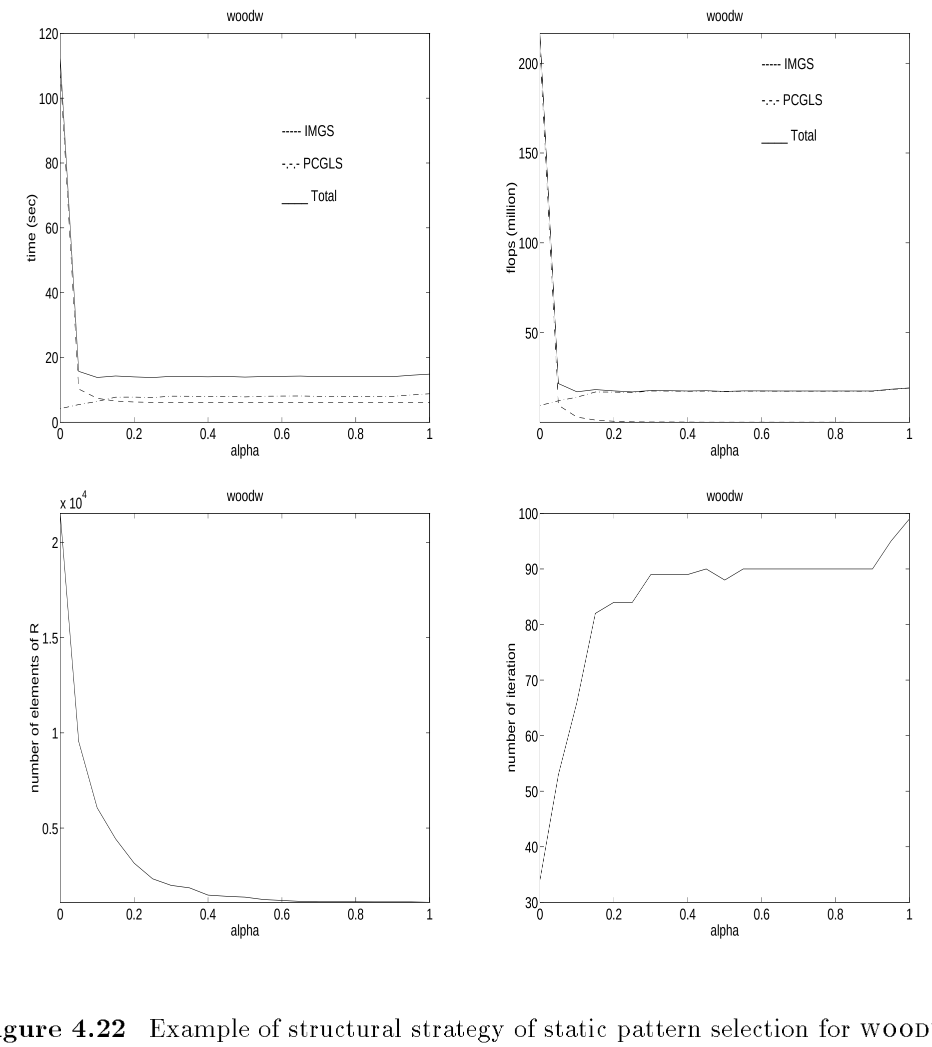


Figure 4.22 Example of structural strategy of static pattern selection for WOODW

The suite does, however, contain matrices for which the quality of the preconditioner is crucial. Two examples are STANDATA and SAYLR4. For these problems, the normal equation method would need to use the pattern for higher levels of retained fill-in in IC, i.e., IC(1) or IC(2), when driving IMGS or ICGS. An alternative to this is the semidynamic scheme discussed below.

4.3.3 Semidynamic pattern selection

Saad, [58], proposed a dropping strategy that falls between the dynamic and static dropping strategies discussed above. In this method, the incompleteness is determined by an integer, s, or set of integers, s_i, $i = 1, \ldots, n$. Only the first s_i largest elements are retained in each column (row) of R after updating. This allows static allocation of the data structures for R but does not determine in advance which elements are kept. We refer to this method of pattern selection as the semidynamic method.

There are advantages to this method in addition to static space. Since we save the s_i largest elements in row (column) i we do not run the risk of dropping an element with large magnitude just because it is in a particular position. This can improve the quality of the preconditioner. On the other hand, it is difficult to anticipate the number of elements that should be kept. Typically, the number of elements in the original row or column plus some added space is used for s_i. It is also possible to take a single integer s and allow all rows or column to grow until that point. The disadvantage of this is that later rows or columns tend to be denser and to use a constant that is large enough to accommodate them wastes space.

It is not so straightforward to apply this technique to the incomplete Gram-Schmidt methods. The easiest to modify is ICGS. At each step, ICGS evaluates a subset of the dotproducts $a_j^T q_i$, $i = 1, \ldots, j-1$. These form a column of R. The semidynamic test is then applied and a_j is updated to q_j using on those q_i for which the corresponding element in the column of R was kept. The savings here are related to keeping Q sparse, i.e., all of the inner products are computed but only some of the updates are performed keep Q sparse and simplifying future dotproducts. The method suffers from the same irony as the dynamic strategy, i.e., you have to compute something to find out you did not need to do so. Table A.5 in the Appendix shows the performance of ICGS using this form of the semidynamic strategy with $s = max(10, n/100)$. It successfully solves 25 of the 30 problems so the semidynamic strategy has not changed the trait that ICGS tends to not produce preconditioners with quality as high as those of IMGS.

It is interesting to compare the results to those presented earlier for ICGS with dynamic pattern selection using $\epsilon = 0.02$. The semidynamic strategy produces an R with significantly fewer elements for 2 of the 25 convergent matrices. On 5 matrices the semidynamic strategy is much worse and on the rest the dynamic and semidynamic yield more or less the same number of non-zero entries. As a result, we conclude that the limit on the number of elements kept per row is a reasonable approximation of the average number of elements per row in the preconditioner produced by the dynamic selection strategy. The quality of the preconditioner does suffer. More iterations are required by the semidynamic strategy for 15 of the 25 matrices. The total time is also disappointing for the semidynamic strategy. It is superior on 4 of the 25 matrices and more or less the same on 3 more. The dynamic strategy is superior on the remaining 17 matrices.

When attempting to apply the semidynamic method to the column form of IMGS some difficulties arise. At each step, the column version computes a dotproduct which is an element of a column of R and then uses the dotproduct and the corresponding q_i to update $a_j^{(i-1)}$ to $a_j^{(i)}$ which is then in turn used in the next dotproduct. Note that the complete update of $a_j^{(j-1)}$ must be performed before it can be decided which of the elements of the j-th column of R are to be dropped. At that point there is no point in dropping them since they made their contributions to the factorization during the update. In fact, if the update is allowed to stand and the q_j is kept for later use but the elements of the column of R are still dropped, the algorithm corresponds to performing MGS on A and then dropping all but the s_i largest elements in magnitude from column i of R. There are no cost savings relative to MGS and the identity $A = QR$ no longer exists. Instead, we have $A = Q(R + E)$ where E is the matrix consisting of dropped elements and Q is an orthogonal matrix when the exact arithmetic is performed. Table A.6 shows the performance of column IMGS with the semidynamic pattern selection. The result is clearly not competitive due to the lack of savings compared to MGS.

The semidynamic method can be applied to the row version of IMGS since it takes a q_j and applies it to all $a_i^{(j-1)}$ to form $a_i^{(j)}$. The required dotproducts form the j-th row of R and can be computed and then filtered by the dropping test. Only those that remain are used to update the corresponding $a_i^{(j-1)}$. So algorithmically, the semidynamic method makes sense. Unfortunately, it requires the use of a dynamic data structure on A as it evolves into Q which was one of the problems we were trying to avoid in the first place. This does not, of course, rule out the effectiveness of such a strategy if the dynamic data structures are handled well. Similar approaches have been used successfully in incomplete LU factorizations [65]. But it is preferred, if possible, to avoid the aggressive use of dynamic data structures while not sacrificing the quality of the preconditioner.

4.3.4 Dropping elements from Q

The major drawback of ICGS and IMGS is the computational and storage costs associated with the columns of Q becoming too dense (recall Figure 4.16). The advantage of attempting to keep Q sparse by controlling the non-zero pattern in R is the guaranteed existence of the factorization. If we are willing to pay the price of a possible 'breakdown' of the factorization, dropping can be applied to the columns of A as they evolve into the columns of Q. In the ICGS and IMGS column versions, we can add the dropping right before r_{ii} is computed or right before it is applied (after which is it easily updated). Given such dropping, the identity $A = QR$ no longer holds so there is no guarantee of existence of the factorization for the full rank A. Instead, let $\hat{q}_i$ be the column vector from which r_{ii} is computed as its norm in ICGS and IMGS and f_i be the vector of dropped elements from $\hat{q}_i$. We then have the following:

$$\hat{q}_i - f_i = r_{ii}q_i = a_i - \sum_{j=1}^{i-1} r_{j,i}q_j - f_i$$

Elements of Q can be dropped in many ways. For example, after the computation and normalization of a column q_i a dropping criteria similar to that used in the dynamic case could be used, e.g., $0 \leq \alpha \leq 1 = \| q_i \|_2$ and elements are dropped if small compared to α. A semidynamic restriction allowing only the largest s elements of q_i to be kept could also be applied.

Zlatev dropped elements in A as it evolved into R when using plane rotations to perform the reduction [65]. The problem of breakdown was handled by decreasing the drop tolerance and restarting the factorization if a failure occurred. The ILQ algorithm proposed in Saad's work, [58], is equivalent to ICGS with semidynamic pattern selection in both Q and R (ILQ will be used to refer to this form of ICGS for the remainder of the thesis). Saad uses a bound on the 2-norm of the vector comprising the elements dropped from each column of Q to assure the preservation of full rank provided that each step uses an ideal orthogonalization process. This result can be extended to ICGS and IMGS with the dropping on Q and is stated as Theorem 11. It is given without proof since its demonstration follows closely that given by Saad.

Theorem 11 *Assume that at every step $j \leq i$ of ICGS (IMGS), the column vector $\hat{q}_j$, as defined earlier, is computed to be orthogonal to $q_1, q_2, ..., q_{j-1}$, and that the vector of elements that are dropped at step j is f_j. If $Q_i = [q_1, \ldots, q_i]$ is of full rank then a sufficient condition for the matrix Q_{i+1} to be of full rank is $\| f_{i+1} \|_2 < r_{i+1,i+1}$.*

As Saad notes, Theorem 11 does not yield a practical method for controlling the loss of rank in Q since the assumptions are rarely met in practice.

Table A.7 contains the results of the ILQ algorithm applied to the test suite. The maximum number of non-zero elements in each column of R is $max(10, n/100)$ and the maximum number of non-zero elements in each column of Q is $max(20, 2m/100)$. No attempt was made to reduce the likelihood of loss of rank in Q. Fortunately, for these parameters, ILQ produces a preconditioner for all of the test problems. This fact may suggest that the rate of breakdown caused by dropping on Q is lower than that of IC factorization.

After comparing this set of examples with the set of examples of ICGS with semidynamic pattern selection without dropping on Q (data are shown in Table A.5), the following observations can be made:

- Although the maximum number of elements for each column of R is the same for both methods, the total number of elements in the resulting R matrices are different. By dropping on Q, we make

Q more sparse so that some elements of R become zero due to structural orthogonality. Therefore the total number of elements is reduced (see Figure 4.23).

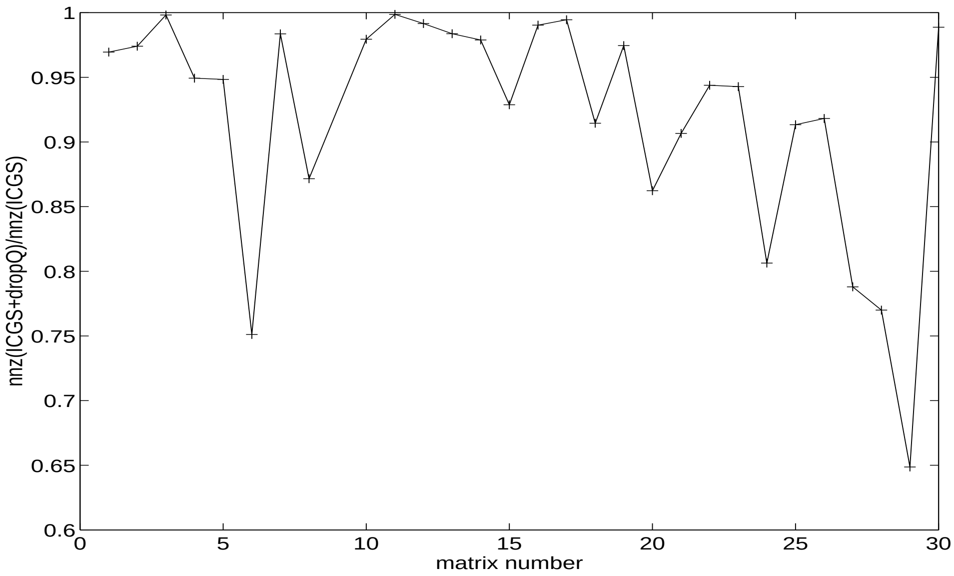


Figure 4.23 Ratio of the number of non-zero elements in R for ILQ to ICGS with dropping on R only

- By dropping on Q, the information carried by the dropped elements is lost and the number of iterations needed when preconditioning CGLS with ILQ is typically greater than that when preconditioning by ICGS with dropping on R only (see Figure 4.24).

- For most of the test problems, the cost of computing the preconditioner via ILQ, measured in both time and operation counts, is lower than that of ICGS dropping on R only. The drop in cost is typically much more than a factor of 2 as can be seen in Figure 4.25. However, there is a significant loss in the quality of the preconditioner. Figure 4.25 also plots the ratio of total time to solve the problem with ILQ to the total time when dropping on R only. Note that, even though the preconditioner exists for all of the matrices, both methods converge for only 24 matrices. On those problems where convergence is achieved dropping on both Q and R is inferior or not significantly superior on 8 problems. On the remaining problems it is superior to dropping on R only.

The final comparison that is informative with this data is to compare ILQ to ICGS with dynamic dropping on R. Recall the earlier comparisons of semidynamic to dynamic dropping on R where, generally, ICGS with dynamic dropping proved superior; and of ILQ to ICGS with semidynamic dropping on R where ILQ was superior in over half of the 24 problems that could be solved. Therefore, it only remains to compare ILQ to ICGS with dynamic dropping on R to complete the ICGS comparisons. Of the 24 convergent matrices, ILQ is slower on 7. For these problems the reduction in preconditioner

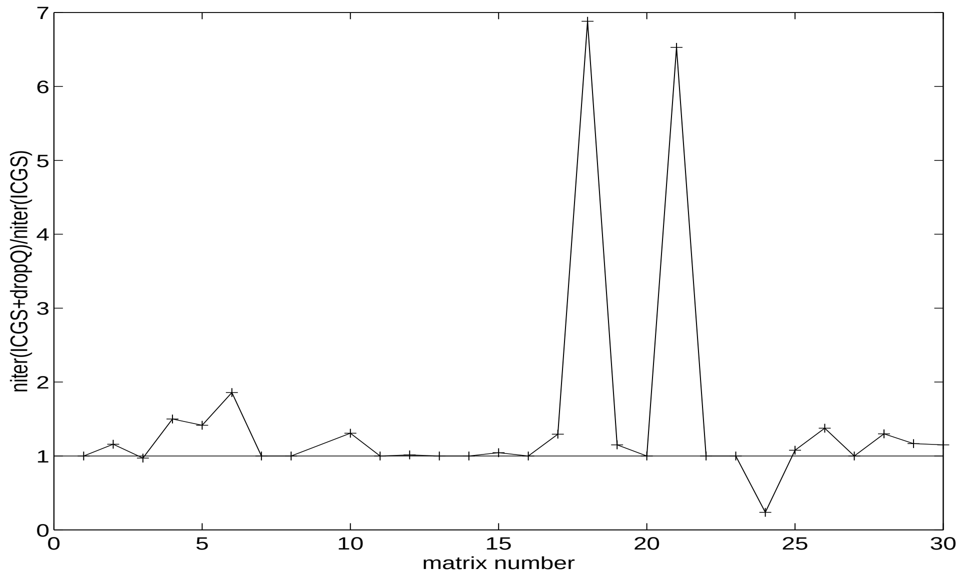
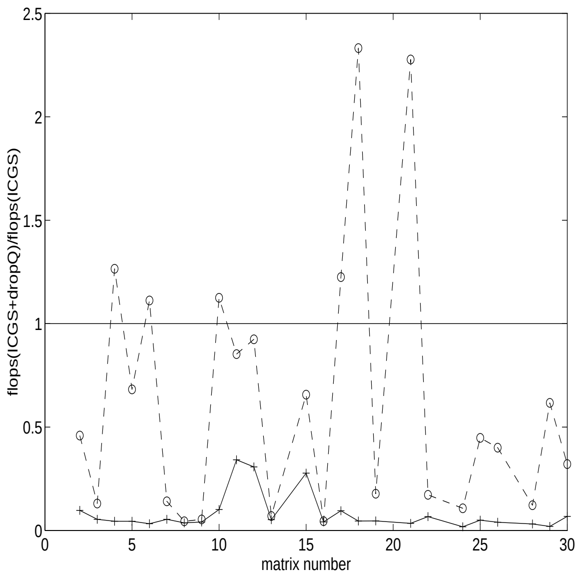


Figure 4.24 Ratio of iterations for CGLS preconditioned by ILQ to that for preconditioned by ICGS with dropping on R only



+ : preconditioner phase
o : total flops to solve problem

Figure 4.25 Ratio of total time for CGLS preconditionerd by ILQ to that for CGLS preconditioned by ICGS dropping on R only production cost is offset by the loss of quality. For 1 problem the two methods have more or less the same performance. For the remaining 16 problems ILQ is superior to varying degrees. It is 2 times faster on 6 matrices, 4 times faster on 6 matrices and 10 times faster on the remaining 4 matrices. So it is clear that applying the semidynamic dropping strategy to both Q and R can improve the overall efficiency of ICGS significantly. Unfortunately, it does not do it consistently and, since it is based on ICGS, the preconditioner is not always of sufficient quality to allow convergence in a reasonable amount of time.

It is straightforward to apply the dropping on Q to the column version of IMGS because Q is generated column by column. On the completion of the computation of each column, dropping is performed to keep Q sparse. At the same time, dropping can be performed on the column of R produced on that step. The performance of column-oriented IMGS with semidynamic dropping on Q and R is shown in Table A.8. The quality of the preconditioner drops considerably when this strategy is used and the resulting algorithm cannot be viewed as competitive.

When dropping on Q and R is attempted in the row version of IMGS, there is a basic question to be considered. Should dropping take place every time we update a column of A as it evolves toward Q or should dropping take place only at the stage that a column will no longer be updated ? If we keep all the intermediate values of columns of Q until there is no more updating, a dynamic data structure is needed along with more space. If we drop the elements after each update, there may be too much information loss so the quality of the preconditioner will degrade. Table A.9 contains the performance of IMGS with semidynamic dropping on Q and R after each column update. Clearly, the quality of preconditioning suffers.

Of course, we can combine dropping on Q with any of the pattern selection strategies used with ICGS and IMGS. The basic motivation is to reduce the cost of computing the preconditioner. Unfortunately, this often comes at the expense of degrading the quality of the preconditioner and introduces the possibility of breakdown. In addition, the added complexity may require the support of dynamic data structures. We have also seen that, in general, IMGS produces superior preconditioners but suffers from much higher computational costs to produce them. This is partially due to the ability to organize the computation of the ICGS preconditioner more efficiently in terms of exploiting sparsity. Chapter 5 considers a method of accelerating the computation of the IMGS preconditioner with only moderate sacrifice in the numerical stability of the production and without introducing the possibility of breakdown in exact arithmetic.

4.4 The effects of column reordering

It is well-known that matrix ordering plays an important role in sparse matrix computations. In this section we consider the use of 4 reordering schemes to reduce the time required to solve the linear least squares problem using preconditioned CGLS. Since the cost of CGLS preconditioned using IMGS or ICGS tends to be dominated by the cost of the computation of the incomplete factorization, we will focus on the reduction of the number of operations involved in the preconditioner production phase.

There are two main sparsity considerations when reducing the operation count of the algorithms in the IGS family for computing $A = QR$. First, the sparsity of R can be manipulated directly. This is addressed by the pattern selection strategies discussed above. It is implicitly assumed that increasing the sparsity of R helps keep Q sparse, i.e., fewer column combinations can mean less fill–in in Q. For static selection strategies, column reorderings of A have no effect on the sparsity of R. For dynamic stragies, however, column ordering can affect the sparsity of R and therefore may affect the sparsity of Q.

Of course, the column ordering does affect the quality of the preconditioner, for both static and dynamic pattern selection. Unfortunately, in general, it is not clear whether degradation or enhancement of quality should be expected. The latter expectation is based on the notion that if the complete Cholesky factor has less fill–in then the incomplete strategy will be discarding fewer elements and therefore less information. This reasoning ignores the fact that discarding a single large or key element of a factorization may do more damage than deleting many small elements. For dynamic and semidynamic pattern selection, reordering to increase the sparsity of the complete Cholesky factor has similar reasoning related to the preconditioning quality. If the sparsity of R is kept at a fixed level then for two different orderings which generate different amounts of fill–in, different values of the drop tolerance may be used. The assumption is that the reordering with larger fill–in will require a large drop tolerance to achieve the desired sparsity and therefore discard more significant information degrading the preconditioning quality. Similarly, if a fixed drop tolerance is used for two different orderings it is assumed that the ordering with less fill–in will discard less significant information while keeping fewer elements. Of course, as with the assumptions above about the number of fill–in elements predicting quality of preconditioning, such reasoning concerning drop tolererances can also be misleading for many problems. In general, there is very little that can be said a priori about the relative quality of the preconditioners for two orderings and application specific information or empirical testing must be relied upon.

Keeping the fill–in small in the complete Cholesky factorization is the motivation for three of the orderings we consider in this section: minimum degree ordering (MD), nested dissection (ND) ordering, reverse Cuthill-McKee (RCM) ordering. These three orderings are standards in sparse computations [33]. As the motivation above indicated they were originally designed to keep fill–in low when solving large sparse symmetric positive definite systems. So they will be applied to the normal equations A^TA and the resulting permutation will be used as a column reordering of A.

The MD ordering works well in keeping the amount of fill–in in the Cholesky factor small. It has been, in its various forms, very successful in improving the efficiency of sequential sparse Cholesky factorization codes. We use it as the standard method for generating a column reordering for A based on controlling the sparsity in R.

ND also attempts to control fill–in by reordering the matrix into block-bordered-diagonal form. Fill–in is limited to a certain block area of the matrix. This not only reduces the fill-in but also has beneficial consequences for the parallel processing. It is known, however, that this ordering tends to generate an IC preconditioner with less quality than other reorderings that are less concerned with minimizing fill–in [21].

The RCM ordering attempts to reduce the width of the profile of the Cholesky factor of the matrix. i.e., to cluster the non-zero elements in a band near the diagonal. This has significance for Gram-Schmidt-type factorizations of the matrix A. If the bandwidth of the Cholesky factor R of A^TA is ω then in the Gram-Schmidt factorization only ω consecutive columns are in active in each step. Hence, if Q is not needed for the preconditioning, then only ω columns of Q need to be stored during the factorization. This may lead to the savings in working space. It may also be beneficial to the iterative solution since the narrow profile of R can enhance the performance of the solution of the triangular system done in each iteration. Note that this ordering does not give any consideration to fill–in reduction. In the tests below, a modification motivated by reducing the fill–in in the matrix Q is introduced. We use the number of non-zero elements of the columns of A as the criterion for 'tie-breaking'. In each step of the RCM algorithm, there may be more than one node of the graph of A^TA that could be placed next in ordering. Among them, the node that corresponds to the column of A that has the smallest number of non-zero elements is numbered first, followed by the node with second smallest number of non-zeros in the corresponding column of A, and so on. Experiments showed that this tie-breaking strategy improves the performance of the ordering compared to the use of random tie-breaking. All results below assume tie-breaking based on column count.

Keeping R sparse may kept the sparsity of Q acceptable. Unfortunately, a small number of combinations with a single nearly dense column of Q can cause tremendous amounts of fill–in. Therefore, the second consideration with respect to column ordering is how it affects the sparsity of Q directly. This is addressed by the simplest ordering strategy we consider – the column weight ordering.

If the number of non-zero elements in a column is used as a weight we can sort the columns of the matrix in the order of increasing weight. This is the column weight ordering (CW). The ordering is very simple and easy to implement. The heuristic assumption is that the columns that are orthogonalized toward the end of the factorization tend to become the densest of Q. Therefore, put the densest columns of A in those positions and have all of the early combinations involve the sparsest columns of A. This has two assumed effects. The increase in density in the early and middle phases of the factorization will be kept small and the later columns that become fairly dense during the factorization were originally like that anyway so the amount of fill–in is kept relatively small.

Experiments were conducted to examine the effects of these ordering schemes on the performance of CGLS preconditioned by IMGS and ICGS for the test suite. The implementations of the MD, ND, and RCM algorithms in SPARSPAK-A, a well known package of solving large sparse symmetric positive definite linear systems, were used with the modification to the RCM algorithm mentioned above added to the code. The orderings tend to affect IMGS and ICGS in a similar fashion and we therefore discuss only those for IMGS.

To evaluate the performance influence of the ordering methods on CGLS preconditioned by IMGS using static pattern selection, we measure and compare the following quantities:

- **The number of operations to compute the preconditioner**. Let nf_0 and nf_1 be the number of operations needed for the computation of R_{IMGS} with and without reordering respectively. The quantity of $\frac{nf_0 - nf_1}{nf_0}$ gives the comparison of the two numbers. If the ordering reduces the number of operations of IMGS then the quantity is positive. If the ordering increases the operations, then the quantity is negative.

- **The number of iterations needed by CGLS preconditioned by IMGS**. Let $iter_0$ and $iter_1$ be this quantity of the method with and without reordering respectively. $\frac{iter_0 - iter_1}{iter_0}$ is used for measurement of the effectiveness of the ordering in accelerating the iterative method. If it is positive, then it does reduce the number of iterations, and hence reduce the computation time of the iteration phase since the number of operations per iteration is not changed when using static pattern selection.

- **The total number of operations**. Let tnf_0 and tnf_1 be the total number of operations for solving the linear least squares problems by CGLS preconditioned by IMGS with and without reordering respectively. The overall effect of the reordering can be seen from the quantity of $\frac{tnf_0 - tnf_1}{tnf_0}$.

In addition to the quantities listed above, when dynamic pattern selection is used we measure and compare the following two additional quantities:

- **The number of non-zero elements in R**. With the same ϵ, reordering changes the sequence of the operations and therefore affects the value of the non-zero elements of R and therefore the number that are kept. If this metric changes with the reordering then there will also be a change in the number of operations required to compute the preconditioner mentioned above. Let nzr_0 and

nzr_1 be the number of non-zeros elements in the factor R produced by IMGS with and without reordering respectively. The measure of $\frac{nzr_0 - nzr_1}{nzr_0}$ indicates the increase or decrease of the quantity.

- **The number of operations in the iterative phase.** Since the number of elements of R changes with the ordering, the number of operations per iteration also changes. Let nf_0 and nf_1 be the number of operations for IMGS factorization with and without ordering respectively. $\frac{inf_0 - inf_1}{inf_0}$ gives the relative comparison of this quantities.

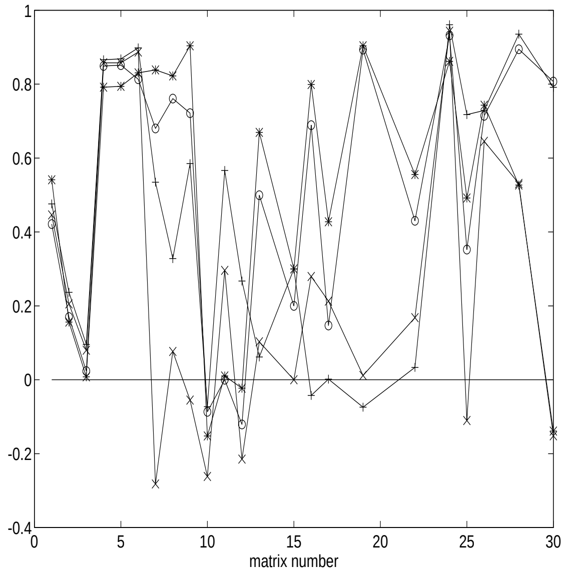
These two additional metrics coupled with the iteration count above allow us to isolate the change in the quality of the preconditioner due to reordering. The tables containing the details of the test results are in Appendix.

The most obvious observation from the data is that the reordering methods tested affect the numerical stability of IMGS. Without reordering, IMGS was able to produce a preconditioner for all matrices in the test suite. A least one matrix causes numerical breakdown of IMGS after each reordering method is applied. After ND FFFFF800 fails and after the other reorderings breakdown occurs for STANDATA. Since we know that in exact arithmetic the methods must succeed, the orderings have clearly affected the numerical stability of the incomplete factorization. Note that these two problems are, in general, difficult to solve. Factorization of the normal equations via SPARSPAK-A also breaks for these problems. The QR method SPARSPAK-B completes but results in a large error.

In general, when the orderings are applied to both IMGS and ICGS using static pattern selection, the number of operations required to produce the preconditioners drops. Figure 4.26 shows the relative change in the number of operations required to produce the IMGS preconditioner using static pattern selection. It can be seen that a fairly small fraction of the ordering problem combinations increased the operations. The CW and MD orderings are more likely to reduce the IMGS operations by a significant amount, while RCM is the least effective.

In general, for the static pattern selection the orderings improve the efficiency of the iterative method phase. Figure 4.27 shows the relative change in the number of iterations which is a direct measure of the change to the quality of the preconditioner for static pattern selection. It can be seen that most problems have a change between no degradation and a reduction in the number of iterations by a factor of 2. A small number degrade slightly. The matrix ORSREG improves considerably for all orderings. Three matrices have significant degradation and one, SEBA, degrades so much for all orderings except ND that it is not shown in Figure 4.27 to avoid making the rest of the plot illegible. The number of iterations to solve it increases from 70, which is less than n, to greater than $2n = 1032$.

We know that the cost of CGLS preconditioned by IMGS tends to be dominated by the computation of preconditioner. Therefore, the effects on the iteration phase of the computation will, in general, have little effect on the total cost to solve the least squares problem. For example, using the RCM ordering



$\frac{nf_0 - nf_1}{nf_0}$

+: CW ordering. $*$: ND ordering.
$\times$: RCM ordering. o: MD ordering.

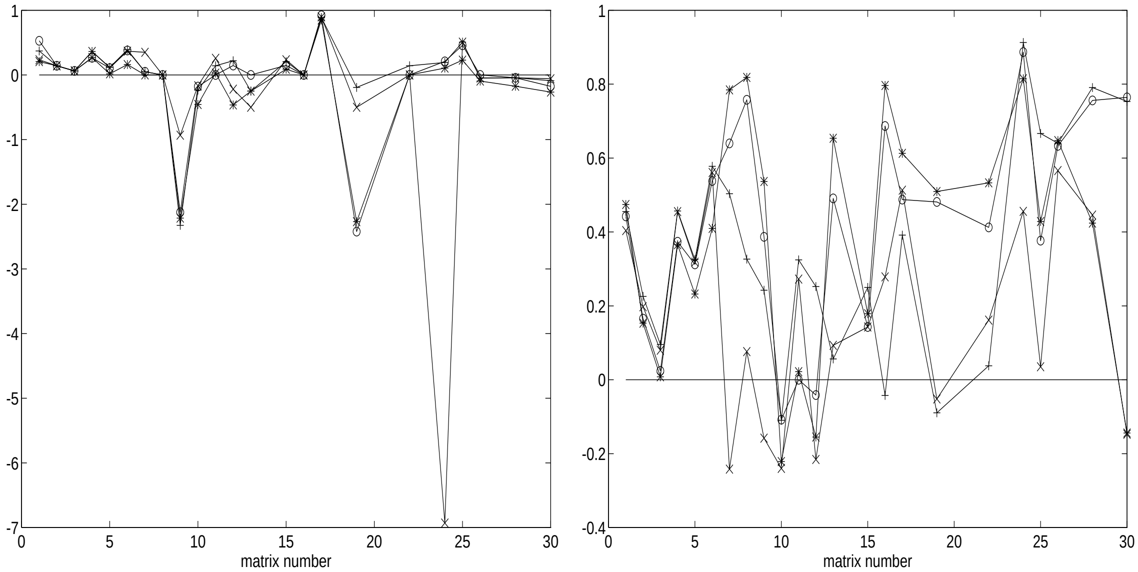
Figure 4.26 Relative change in the number of operations to produce the IMGS preconditioner (static)

on problem NZFRI , the number of iterations increased from 57 to 452. But, since the operation count to produce the IMGS preconditioner is reduced from 98.27×10^6 to 5.96×10^6, the overall cost to solve the problem is still reduced by about 50%. The rightmost graph in Figure 4.27 shows the relative change in the total cost to solve the problems.

It is interesting to note that the cases for which each ordering does not improve the performance of CGLS/IMGS occur for a small common set of test matrices. Further study in the problems may help to reveal the reason for the degradation and to aid in the design of a more suitable ordering algorithm.

When dynamic pattern selection is used, the application of the orderings cause, with the exception of a few problems, reductions, many significant, in the number of operations to produce the IMGS preconditioner. In addition, for a given ϵ the orderings tend to reduce the number of non-zero elements kept in R, see Figure 4.28. Among these methods, RCM is the least effective. MD and CW tend to provide the greatest improvement.

The reduction of the operation count to produce R and number of non-zero elements in R are certainly related. The interesting fact is they are not always affected by an ordering in the same direction. When CW is applied to PDE_9511 the number of non-zeros decreased while the operation counts increased.



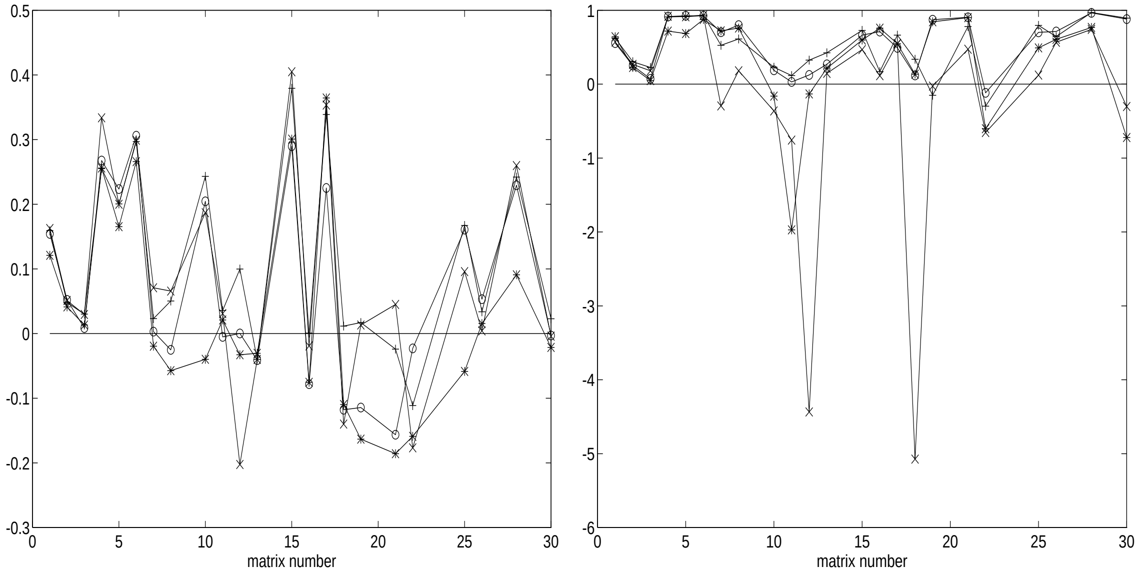
left: $\frac{iter0-iter1}{iter0}$ right: $\frac{tnf_0-tnf_1}{tnf_0}$
$+$: CW ordering. $*$: ND ordering.
$\times$: RCM ordering. o: MD ordering.

Figure 4.27 Relative change in the iteration count and total operation count (static)

This is most likely caused by the fact that the columns associated with the fewer remaining non-zeros of R combined to increase density of Q.

Although there are a few examples that show a great increase or great decrease in the number of iterations of CGLS, for most of the test problems, the rate of convergence only changes slightly for a fixed ϵ, see Figure 4.29. From Figure 4.30, we can see an improvement in the performance of the preconditioned CGLS algorithm for most of the problem/ordering combinations. The RCM ordering tends to improve the performance the least and is responsible for the most significant degradations. The CW and MD orderings tend to produce the most significant improvements.

In summary, the ordering methods tested here are in general effective in reducing the operations in the IMGS and ICGS factorizations. Therefore, they help reduce the total cost of solving linear least squares problems using CGLS preconditioned by IMGS or ICGS. Since these methods are not motivated by the sparse least squares problem, even though they are effective, some properties of least squares problems may not be exploited. For example, the mechanism of the fill-in in Q and R in Gram-Schmidt-type computations is different than that for fill-in in R in Cholesky factorization-type computations. Future work in this area should be concerned with identifying the important distinctions and exploiting them.



left: $\frac{nzr_0 - nzr_1}{nzr_0}$ right: $\frac{nf_0 - nf_1}{nf_0}$
$+$: CW ordering. $*$: ND ordering.
$\times$: RCM ordering. o: MD ordering.

Figure 4.28 Relative change in the number of non-zeros in R and the number of operations to produce the IMGS preconditioner (dynamic)

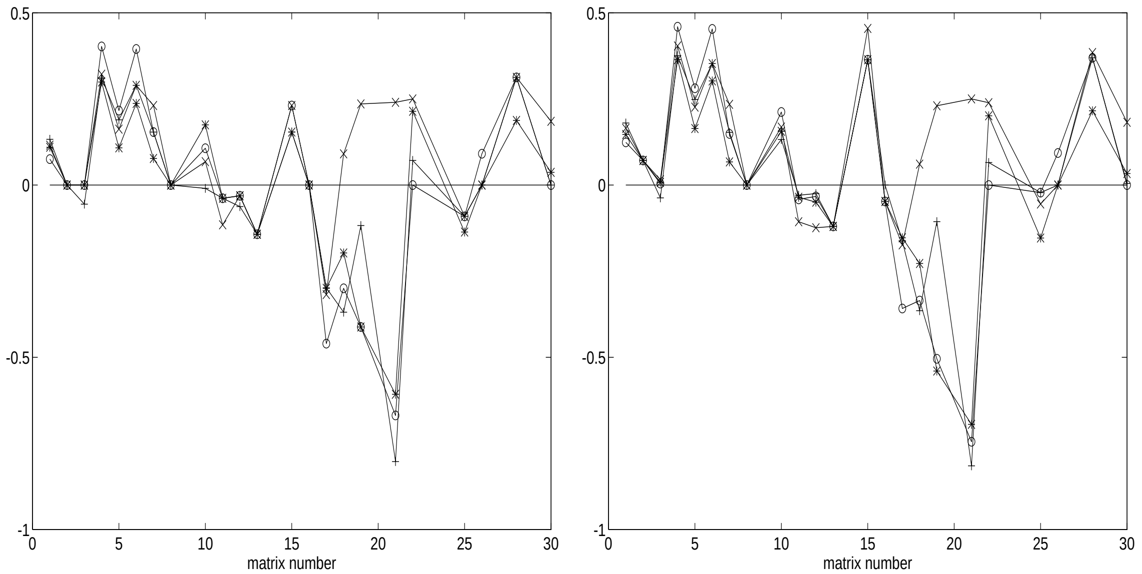
4.5 Comments on ICGS vs. IMGS preconditioning

From the experiments reported above, it is clear that with the same pattern selection strategy and the same reordering, IMGS preconditions more effectively than ICGS in the sense that fewer operations are required in the iterative phase to converge to the requested accuracy. This is consistent with the fact that MGS does not lose orthogonality as quickly due to inexact arithmetic CGS. In this section an heuristic explanation of the superiority of IMGS preconditioning over ICGS preconditioning is given.

Assume that the computation is done in exact arithmetic. Assume also that the loss of orthogonality after $i-1$ steps of CGS is due to some perturbation during the first $i-1$ steps of orthogonalization. Now consider the computation of i-th step of classical and modified Gram-Schmidt and investigate how the loss of orthogonality in previous steps affects the orthogonality of the i-th column of Q to the previous $i-1$ columns.

Let q_i be the orthogonalized i-th column of matrix A. Let $\epsilon_{kj} = q_k^T q_j$ where k and j are less than i. Since there is loss of orthogonality, there is at least one $|\epsilon_{kj}| > 0$. We first investigate the computation of q_i of CGS. At the i-th step of CGS,

$$r_{ji} \quad = \quad q_j^T a_i, \quad j = 1, 2, ..., i-1$$



left: $\frac{iter0-iter1}{iter0}$ right: $\frac{inf_0-inf_1}{inf_0}$
$+$: CW ordering. $*$: ND ordering.
$\times$: RCM ordering. o: MD ordering.

Figure 4.29 Relative change in the number of iterations and the number of operations in the iterative phase (dynamic)

$$\begin{aligned}\tilde{q}_i &= a_i - \sum_{j=1}^{i-1} r_{ji} q_j \\ r_{ii} &= \| \tilde{q}_i \|_2 \\ q_i &= \tilde{q}_i / r_{ii}.\end{aligned}$$

For any $k < i$, we then have

$$q_k^T q_i = (q_k^T a_i - \sum_{j=1}^{i-1} q_k^T q_j r_{ji})/r_{ii} = - \sum_{j=1, j\neq k}^{i-1} q_k^T q_j r_{ji}/r_{ii} = - \sum_{j=1, j\neq k}^{i-1} \epsilon_{kj} r_{ji}/r_{ii}$$

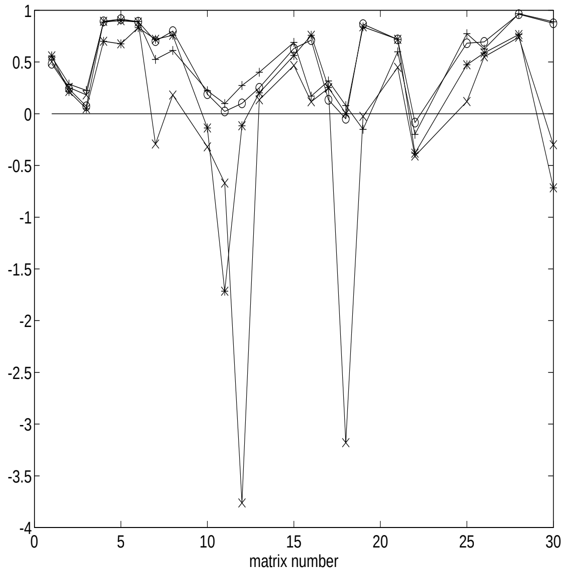
assuming that there is no error in normalization of q_k so that $q_k^T q_k = 1$.

We can see that the value of $q_k^T q_i$ could be affected by any of ϵ_{kj} where $j < i$ and $j \neq k$. In other words, the orthogonality of q_k^T with respect to q_i is affected by the orthogonality of q_k with respect to all the q_j such that $1 \leq j < i$.

Define the affecting region of position (k, i) to be the set of positions (l, s) such that $q_l^T q_s$ will affect the value of $q_k^T q_i$. The affecting region of (k, i) for CGS is shown in (a) of Figure 4.31.

Now consider the computation of q_i of modified Gram-Schmidt. At the i-th step of MGS,

$$\begin{aligned} r_{ji} &= q_j^T a_i^{(j-1)}, \quad j = 1, 2, ..., i-1 \\ a_i^{(j-1)} &= (I - q_{j-1} q_{j-1}^T)(I - q_{j-2} q_{j-2}^T)...(I - q_1 q_1^T) a_i, \quad j = 1, 2, ..., i-1 \end{aligned}$$



$\frac{tnf_0 - tnf_1}{tnf_0}$

$+$: CW ordering. $*$: ND ordering.
$\times$: RCM ordering. o: MD ordering.

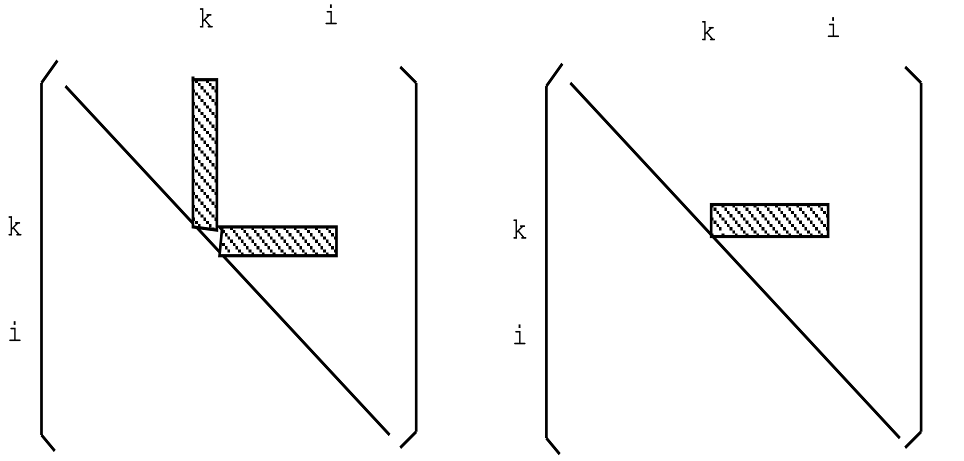
Figure 4.30 Relative change in the total operation count (dynamic)

$$\begin{aligned}
\tilde{q}_i &= a_i^{(i-1)} \\
&= (I - q_{i-1}q_{i-1}^T)(I - q_{i-2}q_{i-2}^T)\ldots(I - q_1q_1^T)a_i \\
r_{ii} &= \| \tilde{q}_i \|_2 \\
q_i &= \tilde{q}_i / r_{ii}.
\end{aligned}$$

Note that for any $k < i$, we now have

$$\begin{aligned}
q_k^T q_i &= q_k^T (I - q_{i-1}q_{i-1}^T) a_i^{(i-2)} / r_{ii} \\
&= (q_k^T a_i^{(i-2)} - r_{i-1,i}\epsilon_{k,i-1}) / r_{ii} \\
&= (q_k^T a_i^{(i-3)} - r_{i-2,i}\epsilon_{k,i-2} - r_{i-1,i}\epsilon_{k,i-1}) / r_{ii} \\
&\vdots \\
&= -(\sum_{j=k+1}^{i-1} r_{j,i}\epsilon_{k,j}) / r_{ii}.
\end{aligned}$$

Note the value of $q_k^T q_i$ is affected by only $\epsilon_{k,j}$ where $k < j < i$. This is different from the classical version. In other words, the orthogonality of q_k^T with respect to q_i is affected by only the orthogonality of q_k with respect to q_j where $k < j < i$. The orthogonality of q_k with respect to any of q_j that is



(a): classical Gram-Schmidt (b): modified Gram-Schmidt

Figure 4.31 Affecting region of $q_k^T q_i$ for CGS and MGS

computed before q_k will not affect $q_k^T q_i$. This is the reason that CGS, in general, loses orthogonality faster than MGS. The affecting region of (k, i) for IMGS is shown in the (b) of the Figure 4.31.

We can see this from another point of view. In CGS, a perturbation on the value of $q_i^T q_j$ will affect the values of $q_j^T q_k$ where $k > j$, and the value of $q_j^T q_k$ will then affect the values of $q_k^T q_l$ where $l > k$. So over all, the values of $q_l^T q_k$, $j < l, k$ will be affected. In MGS,a perturbation at the value of $q_i^T q_j$ will affect the values of $q_i^T q_k$ where $k > j$.

Define the affected region of (i, j) as a set of positions (k, l) such that the value of $q_k^T q_l$ will be affected by the value of $q_i^T q_j$. The affected region of (i,j) for CGS and MGS are shown in (a) and (b) of Figure 4.32 respectively.

From Figure 4.32 we can see that for a given position (i, j), the affecting region and affected region of (i, j) in CGS are both greater than their counterpart in MGS resulting in faster loss of orthogonality for CGS.

A similar explanation can be given for the fact that ICGS is in general a poorer approximation of an orthogonal factorization than IMGS.

In ICGS, for a given zero position set P, we have

$$
\begin{aligned}
r_{ji} &= \begin{cases} q_j^T a_i & (j,i) \notin P \\ 0 & (j,i) \in P \end{cases} \\
\tilde{q}_i &= a_i - \sum_{(j,i) \notin P} r_{ji} q_j \\
r_{ii} &= \| \tilde{q}_i \|_2
\end{aligned}
$$

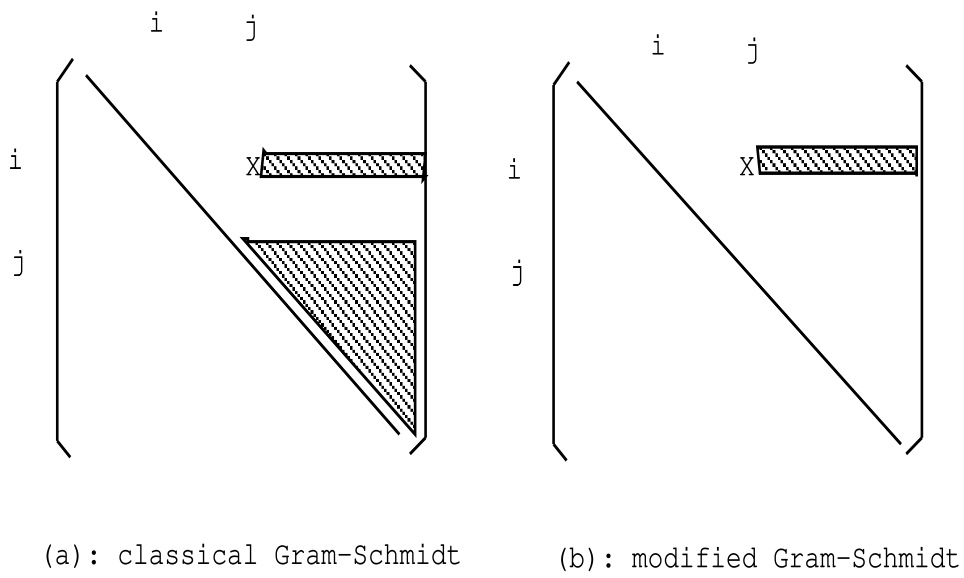


Figure 4.32 Affected region of $q_k^T q_i$ for CGS and MGS

$$q_i \quad = \quad \tilde{q}_i / r_{ii}$$

For any $k < i$, we then have

$$q_k^T q_i = (q_k^T a_i - \sum_{(j,i) \notin P} q_k^T q_j r_{ji}) / r_{ii}.$$

Note that if $(k, i) \in P$ then the first term $q_k^T a_i$ is the dropped element of ICGS. If $(k, i) \notin P$ then $q_k^T a_i = r_{ki}$ so that we have

$$q_k^T q_i = \begin{cases} -\sum_{(j,i) \notin P, j \neq k} \epsilon_{kj} r_{ji} / r_{ii} & (k,i) \notin P \\ (q_k^T a_i - \sum_{(j,i) \notin P} q_k^T q_j r_{ji}) / r_{ii} & (k,i) \in P \end{cases} . \tag{4.1}$$

While for IMGS, we have

$$\begin{aligned} r_{ji} \quad &= \quad \begin{cases} q_j^T a_i^{(j-1)} & (j,i) \notin P \\ 0 & (j,i) \in P \end{cases} , \quad j = 1, 2, ..., i-1 \\ \tilde{q}_i \quad &= \quad (I - q_{j_1} q_{j_1}^T)(I - q_{j_2} q_{j_2}^T) ... (I - q_{j_s} q_{j_s}^T) a_i, \ \ (j_s, i) \notin P \\ r_{ii} \quad &= \quad \| \tilde{q}_i \|_2 \\ q_i \quad &= \quad \tilde{q}_i / r_{ii} \end{aligned}$$

We then have

$$q_k^T q_i = (q_k^T a_i^{(j_s)} - \sum_{(j,i) \notin P, j > k} r_{j,i} \epsilon_{k,j}) / r_{ii}$$

where $j_s = max\{l|(l,i) \notin P \ and \ l \leq k\}$. Note that if $(k,i) \in P$ then the first term $q_k^T a_i^{(j_s)}$ is the dropped element. If $(k,i) \notin P$ then the first term $q_k^T a_i^{(j_s)}$ will be zero so that

$$q_k^T q_i = \begin{cases} -(\sum_{(j,i)\notin P, j>k} r_{j,i}\epsilon_{k,j})/r_{ii} & (k,i) \notin P \\ (q_k^T a_i^{(j_s)} - \sum_{(j,i)\notin P, j>k} r_{j,i}\epsilon_{k,j})/r_{ii} & (k,i) \in P \end{cases} \tag{4.2}$$

Comparing (4.1) with (4.2) we see that affecting region and affected region of a position (i,j) of ICGS is also greater than their counterparts for IMGS.

4.6 Summary

In Chapter 4, we have examined the performance trends of the two basic incomplete orthogonal factorization routines: ICGS and IMGS. A family of preconditioners was created by combining these two basic methods with various matrix ordering schemes and pattern selection strategies, to determine the drop set P. CGLS was taken as the basic iterative method to which the IGS family of preconditioners are applied. A test suite of matrices were used to determine a set of consistent and inconsistent least squares problems and nonsymmetric linear systems. In order to provide a basis of comparison for robustness and efficiency the test problems were solved using CGLS without preconditioning, CGLS with IC preconditioning, and the QR decomposition approach of SPARSPAK-B. A comparison of the first two solution approaches demonstrated the need for and potential gain from preconditioning. While the performance due to IC was substantial on many problems, the many failures to produce an IC factorization demonstrated the need to consider the robustness of the preconditioner. Extensive experiments for the various members of the IGS family were then presented.

The superiority in robustness of IMGS and IMGS compared to IC predicted in Chapter 3 was verified as the two methods were able to produce preconditioners for all problems. The quality of the preconditioner did vary, however, and depended on the problem and the algorithm used to set the drop set P. For some combinations, the IGS preconditioners were unable to cause convergence of CGLS. In general, the IGS family was robust with respect to the test suite in that all problems could be solved by at least one member.

The IMGS-based members were found to be more robust and to produce higher quality preconditioners than variants based on ICGS. For most of problems tested, fewer iterations were required when IMGS-based preconditioners were used. An heuristic explanation for this phenomenon was offered. However, the ICGS-based members typically required far fewer operations to produce their preconditioner due to an organization that allows them to more effectively exploit the sparsity in the problem. This was found to be particularly noticeable when the dynamic pattern selection strategy was used. As a result, ICGS often produced a lower solution cost.

Careful consideration was also given to the interaction of the pattern selection strategy for the drop set P and the various implementations of IMGS and ICGS. The relative inefficiency of IMGS compared to ICGS despite its superior preconditioning capability was mitigated by the use of static and semidynamic pattern selection strategies. For some problems, the slight degradation in preconditioning quality was compensated for by the reduced computational cost of the preconditioner. The dynamic pattern selection strategy tended to produce the best preconditioners for both IMGS and ICGS given that a good value of ϵ was known. A key task to improve the viability of IMGS versions is to reduce the excess cost of its preconditioner computation when dynamic pattern selection is used. The new algorithm described in Chapter 5 attempts to do this.

The ILQ preconditioner of Saad was identified with ICGS with semidynamic pattern selection on R and Q. The aggressive use of dropping on Q despite the risk of breakdown and degradation of preconditioning quality often succeeds in reducing the high cost for the computation of the ICGS preconditioner when Q experiences significant fill-in. The risk of breakdown appears to be less for ILQ than that for IC but it can still be a problem. The main idea of ILQ, the use of semidynamic dropping on Q and R can be applied to any member of the IGS family and does provide a promising avenue for future improvement. An alternate strategy, however, is pursued in the remainder of the thesis.

Finally, Chapter 4 investigated the influence of the application of a set of standard reordering methods to the matrix A in order to reduce the fill-in produced in Q and R without having to resort to the simultaneous semidynamic dropping of ILQ. Four ordering methods were tested. Certain combinations of reordering and pattern selection for the drop set P were able to reduce the cost of computing an IMGS-based preconditioner significantly without degrading the quality of preconditioner. In addition, the orderings are efficient so the overhead associated with their use is small.

CHAPTER 5

Compressed Incomplete Gram-Schmidt Orthogonalization

The work in the previous chapters shows that the family of incomplete Gram-Schmidt preconditioners are effective in accelerating the convergence of the iterative method CGLS but have certain problems. The main drawback is the high cost of computing the incomplete factorization relative to other preconditioners such as IC. Within the IGS family, the degree of costliness varies considerably – recall that ICGS was shown to be more amenable to exploiting the available sparsity than IMGS.

Chapter 4 considered several methods of reducing the cost of the incomplete factorization. Column orderings which keep the amount of fill–in in the complete Cholesky factor and an ordering based on the non-zero counts of the columns A were shown to be effective in reducing the cost of the incomplete factorization. Static, dynamic and semidynamic methods of selecting the drop set P associated with the matrix R were also shown to reduce the cost.

The main problem, however, was identified as the fill-in in the columns of A as they evolve into the columns of Q. The strategies of dropping elements of R reduce this indirectly and are sometimes not effective enough to make the IGS family competetive. A remedy for this problem is to apply dropping to elements of Q as well as R. Saad's ILQ preconditioner, which is equivalent to ICGS using semidynamic dropping on R and Q, was seen to be a viable approach to computing efficiently an IGS factorization. Unfortunately, due to the dropping on Q, the method may breakdown and it is based on ICGS. We would prefer to have an efficient form of IMGS due to its superior preconditioning properties and to avoid the breakdown problem. This chapter considers the development of such a variant of IMGS.

5.1 Algorithm of the CIMGS

The complete Gram-Schmidt factorization of the matrix A and Cholesky factorization of the normal equations A^TA generate, in the exact arithmetic, the same triangular factor R in different ways with potentially significant differences in computational costs. The main source of the increased cost of MGS and CGS relative to Cholesky factorization is the need to maintain and manipulate the data associated with the columns of A as they are transformed into the columns of Q. Since any attempt to drop elements from Q during the factorization results in the possibility of breakdown, we must explore methods that eliminate Q completely from their consideration.

The Cholesky factorization, essentially, compresses the information in A into the normal equations $A^T A$ and removes the need for Q in any form. This results in large savings in computational costs and there is no need to consider ways to maintain the sparsity of Q. The disadvantage of the use of Cholesky factorization is the degradation in robustness when inexact arithmetic is used. When incompleteness is introduced into both Gram-Schmidt and Cholesky factorizations, the IGS family has the advantage of guaranteed existence which is not the case for the IC factorization, but there is still a significant difference in cost with the advantage to IC when both factorizations exist. We would like a factorization that works on the compressed information of the normal equations, but is guaranteed to exist and, ideally, produces the preferred preconditioner from the IGS family – IMGS. In this chapter we derive the Compressed Incomplete Modified Gram-Schmidt factorization (CIMGS) which satisfies these requirements. A detailed description of the algorithm and a proof of the equivalence of IMGS and CIMGS in exact arithmetic follow.

Given a symmetric positive definite matrix $B \in R^{n \times n}$, and a zero position set $P \subset P_n$, CIMGS generates an upper triangular matrix $L^T \in R^{n \times n}$ as follows:

Algorithm [L]=CIMGS[B,P]

begin

for $k = 1, 2, \ldots, n,$

  if $b_{kk} \neq 0$ **then**

    $l_{kk} = \sqrt{b_{kk}}$

    for $j = k+1, k+2, \ldots, n$

      $b_{kj} = b_{kj}/\sqrt{b_{kk}}$

      $l_{kj} = \begin{cases} 0 & (k,j) \in P \\ b_{kj} & (k,j) \notin P \end{cases}$

    endfor

    for $j = k+1, k+2, \ldots, n$

      for $i = k+1, k+2, \ldots, n$

        $b_{ij} = b_{ij} - b_{ki} b_{kj} \quad (k,j) \notin P \; or \; (k,i) \notin P$

      endfor

    endfor

  else

    quit (incomplete factorization can not complete)

endif

endfor

end

There are several ways to structure the CIMGS along the lines of the different versions possible for the other members of the IGS family and Cholesky factorization. The version above is similar in structure to the rank–1 update form of Cholesky factorization and is the most straightforward implementation of CIMGS. It is also the most convenient version to use for the purpose of analyzing its properties. In the experiments below we use a more efficient form that is based on delaying the application of the rank-1 updates.

As with IMGS, pattern selection methods, such as those discussed in Chapter 4 can be used to improve the performance of CIMGS. Since CIMGS avoids the updates of the columns of Q which caused the difficulties in applying dynamic pattern selection to IMGS, all three pattern selection strategies, static, dynamic, and semidynamic can be applied in a straightforward fashion.

Note that this form of the algorithm makes clear one of the key aspects of CIMGS – some non-zero elements that are produced and updated by the rank-1 update are later dropped in the assignment to l_{kj}. Of the elements that are subsequently dropped, some make contributions to elements that are kept in R and others are completely unnecessary. This aspect of CIMGS will be discussed in detail later in the thesis. Its main consequence at this point is that the pattern selection operation involves only the statement that assigns values to l_{kj}. For static pattern selection, the statement takes the form:

$$l_{kj} = \begin{cases} 0 & (k,j) \in P \\ b_{kj} & (k,j) \notin P. \end{cases} ;$$

and for dynamic pattern selection,

$$l_{kj} = \begin{cases} 0 & |b_{kj}| < \epsilon \\ b_{kj} & \text{otherwise.} \end{cases} .$$

Both have the general form

$$l_{kj} = \begin{cases} 0 & \textit{rejected} \\ b_{kj} & \textit{accepted.} \end{cases}$$

The CIMGS algorithm, as expected, falls between Cholesky factorization, where all of the dropping conditions would be removed and IC where the dropping conditions in the rank-1 update would delete the elements in positions in P and update all others. As is seen above, the dropping conditions for CIMGS fall in between these two extremes and by doing so guarantee the existence of the incomplete factorization while gaining the computational advantage of fewer operations. Keeping this advantage as large as possible by avoiding unnecessary computations is discussed in Section 5.4.

5.2 Properties of CIMGS

We first show that CIMGS satisfies the main requirement listed above: when it is applied to A^TA it is equivalent to IMGS applied to A. The existence of the incomplete factorization follows and the breakdown problem is avoided along with the computational cost of dealing with Q.

Theorem 12 *Let $A \in R^{m\times n}$, $m \geq n$, $rank(A) = n$ and $B = A^TA$. If $R = IMGS(A,P)$ and $L^T = CIMGS(B,P)$, then $R = L^T$.*

Proof: We demonstrate the equivalence of CIMGS and IMGS by induction on n.

For $n = 1$,

$$R = \| a_1 \|_2 = \sqrt{a_1^T a_1},$$

$$L^T = \sqrt{b_{11}} = \sqrt{a_1^T a_1},$$

and so $R = L^T$.

Supposing that the theorem is true for $k = n-1$, we now prove it for $k = n$. After one step of the IMGS,

$$r_{11} = \| a_1 \|_2 = \sqrt{a_1^T a_1},$$

$$r_{1j} = \begin{cases} 0 & (1,j) \in P \\ \frac{a_1^T a_j}{r_{11}} & (1,j) \notin P \end{cases}$$

and columns 2 to n of the remaining rows of A are updated as

$$a_j^{(1)} = a_j - \frac{r_{1j}}{r_{11}} a_1, \quad 2 \leq j \leq n.$$

After one step of CIMGS the first row of L^T is given by

$$l_{11} = \sqrt{b_{11}} = \sqrt{a_1^T a_1},$$

$$l_{1j} = \begin{cases} 0 & (1,j) \in P \\ b_{1j}/\sqrt{b_{11}} & (1,j) \notin P \end{cases}$$

and columns 2 to n of the remaining rows of B are

$$b_{ij}^{(1)} = b_{ij} - b_{1i} b_{1j} \quad (1,j) \notin P \quad or \quad (1,i) \notin P$$

for $2 \leq i,j \leq n$. Clearly $r_{1j} = l_{1j}$ for all $1 \leq j \leq n$. The computation of the rest of R consists of applying IMGS to the matrix $A^{(1)} = \{a_2^{(1)}, a_3^{(1)}, \ldots, a_n^{(1)}\}$. The computation of the rest of L^T consists of applying CIMGS to the matrix

$$B^{(1)} = \begin{pmatrix} b_{22}^{(1)} & b_{23}^{(1)} & b_{34}^{(1)} & \dots & \dots & b_{2n}^{(1)} \\ b_{32}^{(1)} & b_{33}^{(1)} & b_{34}^{(1)} & \dots & \dots & b_{3n}^{(1)} \\ \vdots & \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \vdots & & & \vdots \\ b_{n2}^{(1)} & b_{n3}^{(1)} & b_{n4}^{(1)} & \dots & \dots & b_{nn}^{(1)} \end{pmatrix}$$

Since

$$a_i^{(1)^T} a_j^{(1)} = \begin{cases} a_i^T a_j & (1,i) \in P \; and \; (1,j) \in P \\ a_i^T a_j - r_{1i} r_{1j} & (1,i) \notin P \; or \; (1,j) \notin P \end{cases}$$

and

$$b_{ij}^{(1)} = \begin{cases} b_{ij} & (1,i) \in P \; and \; (1,j) \in P \\ b_{ij} - b_{1i} b_{1j} & (1,i) \notin P \; or \; (1,j) \notin P, \end{cases}$$

it can easily be seen that $A^{(1)^T} A^{(1)} = B^{(1)}$. Using the induction assumption, the rest of R computed by IMGS is equal to the rest of L^T computed by CIMGS. By induction, the theorem is true for any n. $\square$

Because of the equivalence of IMGS and CIMGS in exact arithmetic, the properties of IMGS discussed in the previous chapters also hold for CIMGS. Conversely, the properties of CIMGS that are proven below, e.g., the monotonicity property of CIMGS with respect to the position set, are also true for IMGS even though they are difficult to prove from that form of the algorithm.

It has been shown that if $P \subset \tilde{P}$ are two drop sets used for IC factorizations of a symmetric positive definite M-matrix, then $R \leq \tilde{R}$ where this inequality is taken componentwise, and R and $\tilde{R}$ are the corresponding IC factors [26]. Lemma 7 and Theorem 13 show that this monotonicity also holds for CIMGS when the normal equations are an M-matrix.

Lemma 7 *Let $A \in \Re^{n \times n}$ and $B \in \Re^{n \times n}$ be symmetric positive definite M-matrices where $A \leq B$. If R and T are the upper triangular matrices from the CIMGS factorization of A and B respectively with the same non-zero position set P, then $R \leq T$.*

Proof: We prove the proposition by induction on n. Clearly the result holds for $n = 1$. Assume that the result holds for matrices of order $n - 1$. Let

$$R = \begin{pmatrix} r_{11} & (r_{12}, r_{13}, \dots, r_{1n}) \\ 0 & R_{22} \end{pmatrix},$$

$$T = \begin{pmatrix} t_{11} & (t_{12}, t_{13}, \dots, t_{1n}) \\ 0 & T_{22} \end{pmatrix},$$

$$A = \begin{pmatrix} a_{11} & (a_{12}, a_{13}, \ldots, a_{1n}) \\ \begin{pmatrix} a_{12} \\ a_{13} \\ \vdots \\ a_{1n} \end{pmatrix} & A_{22} \end{pmatrix},$$

and

$$B = \begin{pmatrix} b_{11} & (b_{12}, b_{13}, \ldots, b_{1n}) \\ \begin{pmatrix} b_{12} \\ b_{13} \\ \vdots \\ b_{1n} \end{pmatrix} & B_{22} \end{pmatrix}$$

be the corresponding partitioned matrices for a matrix of order n, so that R_{22}, T_{22}, A_{22} and B_{22} are of order $n-1$. After one step of CIMGS on A and B, we have $0 < r_{11} = \sqrt{a_{11}} \leq \sqrt{b_{11}} = t_{11}$ and, since $b_{1i} \leq 0$, we also have

$$r_{1i} = \begin{cases} a_{1i}/r_{11} \leq b_{1i}/r_{11} \leq b_{1i}/t_{11} = t_{1i} & (1,i) \notin P \\ 0 = t_{1i} & (1,i) \in P. \end{cases}$$

Let f and g be the vectors of dropped elements from the first step of CIMGS applied to A and B, respectively and let $R_{12} = (r_{12}, r_{13}, \ldots, r_{1n})^T$ and $T_{12} = (t_{12}, t_{13}, \ldots, t_{1n})^T$. It follows that $f \leq g \leq 0$ and $R_{12} \leq T_{12} \leq 0$. R_{22} and T_{22} are the CIMGS factors of $A^{(2)} = A_{22} - R_{12}f^T - fR_{12}^T - R_{12}R_{12}^T$ and $B^{(2)} = B_{22} - T_{12}g^T - gT_{12}^T - T_{12}T_{12}^T$ and therefore,

$$\begin{aligned} A^{(2)} &= A_{22} - R_{12}f^T - fR_{12}^T - R_{12}R_{12}^T \\ &\leq B_{22} - R_{12}f^T - fR_{12}^T - R_{12}R_{12}^T \\ &\leq B_{22} - T_{12}f^T - fT_{12}^T - T_{12}T_{12}^T \\ &\leq B_{22} - T_{12}g^T - gT_{12}^T - T_{12}T_{12}^T \\ &\leq B^{(2)}. \end{aligned}$$

By the induction hypothesis, $R_{22} \leq T_{22}$, and so $R \leq T$ and the proposition is true for matrices of any order $n \geq 1$. $\square$.

Theorem 13 *Let $B \in R^{n \times n}$ be a symmetric positive definite M-matrix, and let $P_1 \subset P_2$ be zero position sets. If R_1 and R_2 are the CIMGS factors produced by using P_1 and P_2, respectively then $R_1 \leq R_2$.*

Proof: We prove the proposition by induction on n. For $n = 1$ the result holds trivially. Assume that it is true for matrices of order $n-1$. Let

$$B = \left(\begin{array}{cc} b_{11} & (b_{12}, b_{13}, \ldots, b_{1n}) \\ \left(\begin{array}{c} b_{12} \\ b_{13} \\ \vdots \\ b_{1n} \end{array} \right) & B_{22} \end{array} \right),$$

$$R_1 = \left(\begin{array}{cc} r_{11}^{(1)} & (r_{12}^{(1)}, r_{13}^{(1)}, \ldots, r_{1n}^{(1)}) \\ & R_{22}^{(1)} \end{array} \right),$$

and

$$R_2 = \left(\begin{array}{cc} r_{11}^{(2)} & (r_{12}^{(2)} r_{13}^{(2)} \ldots . r_{1n}^{(2)}) \\ & R_{22}^{(2)} \end{array} \right)$$

be partitioned so that B_{22}, $R_{22}^{(1)}$, and $R_{22}^{(2)}$ are of order $n-1$. After one step of CIMGS with P_1 and P_2, respectively, $r_{11}^{(1)} = \sqrt{b_{11}} = r_{11}^{(2)}$ and since $B_{1i} \leq 0$,

$$r_{1i}^{(1)} = \begin{cases} b_{1i}/r_{11}^{(1)} = b_{1i}/r_{11}^{(2)} = r_{1i}^{(2)} & (1,i) \notin P_1 \text{ and } (1,i) \notin P_2 \\ b_{1i}/r_{11}^{(1)} \leq 0 = r_{1i}^{(2)} & (1,i) \notin P_1 \text{ and } (1,i) \in P_2 \quad i = 2, \ldots, n \\ 0 = r_{1i}^{(2)} & (1,i) \in P_1 \text{ and } (1,i) \in P_2 \end{cases} \tag{5.1}$$

If $R_{12}^{(1)} = (r_{12}^{(1)}, r_{13}^{(1)}, \ldots, r_{1n}^{(1)})$ and $R_{12}^{(2)} = (r_{12}^{(2)}, r_{13}^{(2)}, \ldots, r_{1n}^{(2)})$ then (5.1) implies $R_{12}^{(1)} \leq R_{12}^{(2)}$.

If $f = (f_2, f_3, \ldots, f_n)$ and $g = (g_2, g_3, \ldots, g_n)$ be the vectors of elements dropped by CIMGS with P_1 and P_2, respectively, then

$$f_i = \begin{cases} b_{1i}/r_{11}^{(1)} = b_{1i}/r_{11}^{(2)} = g_i & (1,i) \in P_1 \text{ and } (1,i) \in P_2 \\ 0 \geq b_{1i}/r_{11}^{(2)} = g_i & (1,i) \notin P_1 \text{ and } (1,i) \in P_2 \quad i = 2, \ldots n. \\ 0 = g_i & (1,i) \notin P_1 \text{ and } (1,i) \notin P_2 \end{cases} \tag{5.2}$$

and so $0 \geq f \geq g$. Note that $R_{12}^{(1)} + f = R_{12}^{(2)} + g$.

Now let $B_{22}^{(1)}$ and $B_{22}^{(2)}$ be the reduced matrices of order $n-1$ produced by one step of CIMGS with P_1 and P_2, respectively. It follows that

$$\begin{aligned} B_{22}^{(1)} &= B_{22} - (R_{12}^{(1)} + f)^T (R_{12}^{(1)} + f) + f^T f \\ &\leq B_{22} - (R_{12}^{(1)} + f)^T (R_{12}^{(1)} + f) + g^T g \\ &\leq B_{22} - (R_{12}^{(2)} + g)^T (R_{12}^{(2)} + g) + g^T g \\ &= B_{22}^{(2)}. \end{aligned}$$

Let $R_{22}^{(1)}$ be the CIMGS factor for $B_{22}^{(1)}$ using P_1, $R_{22}^{(2)}$ be the CIMGS factor for $B_{22}^{(2)}$ using P_2, and T be the CIMGS factor for $B_{22}^{(2)}$ using P_1. By Lemma 7, $R_{22}^{(1)} \leq T$. By the induction hypothesis $T \leq R_{22}^{(2)}$. Together we have $R_1 \leq R_2$ and therefore the theorem is true for matrices of order n. □

Let R^c be the Cholesky factor of B. R^c is equal to the CIMGS factor produced by using the pattern $P^c = \emptyset$. Clearly, we have $P^c \subset P_1 \subset P_2$ and, by Theorem 13, $R^c \leq R_1 \leq R_2$. If $E_1 = R_1 - R^c$ and $E_2 = R_2 - R^c$, then we have $E_1 \leq E_2$. In this sense we say that R_1 better approximates R^c. This is a formal way of saying that if we drop fewer elements, i.e., we use a smaller drop set P, the resulting IMGS factor approximates the Cholesky factor in a componentwise sense.

The results above extend the properties of IMGS in exact arithmetic since we can view CIMGS as a particularly efficient implementation of IMGS. For the remainder of this section we consider the numerical properties of the CIMGS implementation and explore the relationship between CIMGS and Cholesky factorization. Theorem 14 first establishes an identity that provides a vehicle by which the numerical errors in CIMGS can be analyzed.

Theorem 14 *Let $B \in R^{n\times n}$ be a symmetric positive definite matrix. Let R be the upper triangular CIMGS factor with drop set P and U be the upper triangular matrix of elements dropped from B during the CIMGS factorization. We have*

$$(R+U)^T(R+U) - U^TU = B$$

Proof: We prove this by induction on n. Let

$$B = B^{(1)} = \begin{pmatrix} b_{11} & b_1 \\ b_1^T & B_{22} \end{pmatrix}, \; R = \begin{pmatrix} r_{11} & r_1 \\ 0 & R^{(2)} \end{pmatrix}, \; U = \begin{pmatrix} 0 & u_1 \\ 0 & U^{(2)} \end{pmatrix}$$

be partitionings of the matrices B, R and U respectively, so that B_{22}, R_{22}, and U_{22} are of order $n-1$. Note that all the diagonal elements of U are zero because the definition of the drop set P prohibits dropping the diagonal elements. After one step of CIMGS

$$\begin{pmatrix} r_{11} & 0 \\ r_1^T + u_1^T & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & B^{(2)} \end{pmatrix} \begin{pmatrix} r_{11} & r_1 + u_1 \\ 0 & I \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ u_1^T & 0 \end{pmatrix} \begin{pmatrix} 0 & u_1 \\ 0 & 0 \end{pmatrix} = B^{(1)} = B, \quad (5.3)$$

where

$$B^{(2)} = B_{22} - r_1^T u_1 - u_1^T r_1 - r_1^T r_1. \quad (5.4)$$

Note that we can write (5.4) as

$$B^{(2)} = (B_{22} - (r_1+u_1)^T(r_1+u_1)) + u_1^T u_1.$$

The first two terms are the reduced system produced by one step of Cholesky factorization and the last term is symmetric semidefinite. Therefore $B^{(2)}$ is symmetric positive definite.

The remaining steps of CIMGS are equivalent to applying CIMGS to $B^{(2)}$. By the induction hypothesis, $B^{(2)} = (R^{(2)} + U^{(2)})^T(R^{(2)} + U^{(2)}) - U^{(2)^T}U^{(2)}$. Substituting this into (5.3) and simplifying, we get

$$(R+U)^T(R+U) - \begin{pmatrix} 0 & 0 \\ 0 & U^{(2)^T}U^{(2)} \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ u_1^T & 0 \end{pmatrix}\begin{pmatrix} 0 & u_1 \\ 0 & 0 \end{pmatrix} = B.$$

This can be written as

$$(R+U)^T(R+U) - U^TU = B,$$

proving the Theorem. □

Note that $R+U$ is the upper triangular part of B after the completion of the algorithm and U^TU is symmetric positive semidefinite[1]. This relation says that CIMGS is equivalent to applying Cholesky factorization to $B + U^TU$ and then dropping U from the resulting triangular factor to get R. If B is symmetric positive definite, then $B + U^TU$ is also symmetric positive definite and the Cholesky factorization of $B + U^TU$ exists. This provides another, independent, proof that if A is full rank, the CIMGS factorization of A^TA exists. This relation also indicates that the error of CIMGS factorization due to the use of finite precision arithmetic is not only related to the condition of B but also to that of $B + U^TU$. The following makes this observation more precise.

Theorem 15 *Let B be a symmetric positive definite matrix and $\mu = 2^{-t}$ where t is the number of bits in the mantissa used in the finite precision computations. Let P be a given drop set of zero positions for the CIMGS algorithm. Let R be the triangular factor produced by CIMGS using the set P, and let U be the matrix of dropped elements. If*

$$\kappa(B) \geq C_1(n)\mu, \tag{5.5}$$

where $\kappa(B)$ is the condition number of B, then there is an error matrix E such that

$$(R+U)^T(R+U) = B + U^TU + E$$

and

$$\| E \|_2 \leq C_2(n)\mu \| B \|_2,$$

where $C_1(n)$ and $C_2(n)$ are constants that depend only on n.

The proof draws on results established by Wilkinson in his classic paper *A priori error analysis of algebra processes* [60], which we summarize before proceeding with the proof.

[1]Recall that in the version of the algorithm the elements of B were maintained separately from the elements of $L^T = R$.

Let $B = B^{(1)} = \begin{pmatrix} b_{11} & b_1 \\ b_1^T & B_{22} \end{pmatrix}$ and $T = \begin{pmatrix} t_{11} & t_1 \\ 0 & T^{(2)} \end{pmatrix}$ be its computed Cholesky factor. After the computation of one step of Cholesky factorization,

$$\begin{pmatrix} t_{11} & \\ t_1^T & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & B_{CHOL}^{(2)} \end{pmatrix} \begin{pmatrix} t_{11} & t_1 \\ & I \end{pmatrix} = B + E_{CHOL}^{(1)}.$$

where $t_{11} = fl(\sqrt{b_{11}})$, $t_1 = fl(b_1/t_{11})$, and $B_{CHOL}^{(2)} = fl(B_{22} - t_1^T t_1)$. The computation of the Cholesky factorization of B is equivalent to the computation of the first row of T and the computation of $B_{CHOL}^{(2)}$ followed by the factorization of $B_{CHOL}^{(2)}$. Wilkinson showed that

$$\| E_{CHOL}^{(1)} \|_2 \leq c_1 \mu \| B \|_2,$$

$$\| t_1^T t_1 \|_2 = \| t_1 t_1^T \|_2^2 \leq c_2 \| B \|_2,$$

$$\| B_{CHOL}^{(2)} \|_2 \leq c_3 \| B \|_2$$

and $B_{CHOL}^{(2)}$ is symmetric positive definite, where c_1, c_2 and c_3 are the constants that depend on n. By induction on n, he showed that if B is a symmetric positive definite matrix of order n, and if

$$C_3(n)\mu \| B \|_2 \| B^{-1} \|_2 \leq 1,$$

then Cholesky factorization can be completed without breakdown and the computed factor T satisfies the relation $T^T T = B + E_{CHOL}$ with

$$\| E_{CHOL} \|_2 \leq C_4(n)\mu \| B \|_2,$$

where $C_3(n)$ and $C_4(n)$ are constants depending on n.

Proof of Theorem 15: Our proof of the theorem is again by induction on n. Assume that the result is true for matrices of order $n-1$. Let $U = \begin{pmatrix} 0 & u_1 \\ 0 & U^{(2)} \end{pmatrix}$ and $R = \begin{pmatrix} r_{11} & r_1 \\ 0 & R^{(2)} \end{pmatrix}$ be the triangular matrix of the dropped elements and the CIMGS factor, respectively, and denote $L = R + U = \begin{pmatrix} l_{11} & l_1 \\ 0 & L^{(2)} \end{pmatrix} = \begin{pmatrix} r_{11} & r_1 + u_1 \\ 0 & R^{(2)} + U^{(2)} \end{pmatrix}$.

Consider the computation of the first rows of U and R in CIMGS. After one step of CIMGS,

$$\begin{pmatrix} l_{11} & 0 \\ l_1^T & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & B_{CIMGS}^{(2)} \end{pmatrix} \begin{pmatrix} l_{11} & l_1 \\ 0 & I \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & u_1^T u_1 \end{pmatrix} = B + E_{CIMGS}^{(1)}.$$

Note that in positions (i,j) with $(1,i) \notin P$ or $(1,j) \notin P$, the computation of $B^{(2)}$ is exactly the same as that in one step of Cholesky factorization of B. In positions (i,j) with both $(1,i)$ and $(1,j)$ in P, there is no updating, and so, no rounding errors. Therefore, The magnitude of the elements of $E_{CIMGS}^{(1)}$ is smaller than the magnitude of the elements of $E^{(1)}_{CHOL}$, because the errors are the same in some positions and zeros in all other positions. Using Wilkinson's estimates,

$$\| E^{(1)}_{CIMGS} \|_2 \le c_1\mu \| B \|_2 .$$

Since

$$\begin{aligned} B^{(2)}_{CIMGS} &= (B_{22} - r_1^T r_1 - r_1^T u_1 - u_1^T r_1) + G_1 \\ &= (B_{22} - l_1^T l_1 + u_1^T u_1) + G_1 \end{aligned}$$

where G_1 is the matrix $E^{(1)}_{CIMGS}$ without its first row and column, we have

$$\begin{aligned} \| B^{(2)}_{CIMGS} \|_2 &\le \| (B_{22} - l_1^T l_1) \|_2 + \| u_1^T u_1 \|_2 + \| G_1 \|_2 \\ &\le \| B_{22} \|_2 + \| l_1 l_1^T \|_2 + c_1\mu \| B \|_2 \\ &\le c_4 \| B \|_2 \end{aligned}$$

where c_4 is a constant depending on n.

From the computation of one step of CIMGS we can see that $B^{(2)}_{CIMGS}$ is also positive definite. Using the induction hypothesis,

$$B^{(2)}_{CIMGS} = L^{(2)^T} L^{(2)} - U^{(2)^T} U^{(2)} - E^{(2)}_{CIMGS}$$

and

$$\begin{aligned} \| E^{(2)}_{CIMGS} \|_2 &\le C_2(n-1)\mu \| B^{(2)}_{CIMGS} \|_2 \\ &\le c_5\mu \| B \|_2 \end{aligned}$$

where c_5 is a constant depending on n. So we get

$$L^T L - U^T U - E_2 - E_1 = B$$

where $E_1 = E^{(1)}_{CIMGS}$ and E_2 is $E^{(2)}_{CIMGS}$ augmented by a null first row and column. It follows that

$$\begin{aligned} \| E \|_2 &= \| E_1 + E_2 \|_2 \\ &\le \| E_1 \|_2 + \| E_2 \|_2 \\ &\le c_1\mu \| B \|_2 + c_5\mu \| B \|_2 \end{aligned}$$

and by letting $C_2(n) = c_1 + c_5$, the error bounds are established.

If the minimum eigenvalue of $B + U^T U$ is greater than the error, the factorization process will not break down. That is,

$$\frac{1}{\| (B + U^T U)^{-1} \|_2} \ge C_1(n) \| B \|_2,$$

completing the proof. □

Since $U^T U$ is symmetric positive semidefinite, the smallest eigenvalue of $B + U^T U$ is greater than or equal to the smallest eigenvalue of B. So we may expect that breakdown will be less likely to happen for CIMGS factorization than it is for Cholesky factorization. On the other hand, since the condition number does involve $B = A^T A$, we may also expect that, when A is ill-conditioned, CIMGS on B may breakdown earlier than IMGS on A. These expectations are demonstrated by the test problems. For FFFFF800, IMGS with static selection of the pattern of normal equations completes while CIMGS and Cholesky fail because of the numerical breakdown. For the matrices CONVE8, SHELL, and STANDATA, both IMGS and CIMGS produce a factorization but the Cholesky factorization experiences numerical breakdown for all of the orderings investigated.

5.3 Performance of CIMGS

In this section we consider the performance improvement that results from the elimination of Q from IMGS via the CIMGS implementation. We have already seen that the reduction in the operation count is not without its cost since the compression does cause some ill conditioned matrices to have difficulties numerically. For these matrices Cholesky also fails and the ICGS or IMGS preconditioners become viable options, i.e., their extra work becomes justified. The experiments in this section use an implementation that is different from the rank-1-based approach presented above in that it uses delayed updates. Each row of the normal equations is considered in turn and converted into the corresponding row of B which contains the elements in the same row of R and other elements which are in P but may be required for the computation of R. This organization facilitates the use of a very simple dynamic data structure to hold the row as it evolves from $A^T A$ to B. The fill–in elements are computed using a simple pattern union computation for each sparse row triad performed. No fast a priori symbolic factorization is used. There are two basic functions that a fast symbolic factorization could accelerate. The first is the computation of the non-zero pattern of B given a static P and $A^T A$. This pattern would include elements of R and elements in P which may or may not be useful. A fast symbolic Cholesky factorization of the normal equations could be used to generate a bounding pattern and when B is similar in density to the Cholesky factor this is probably the best approach. However, when B is much sparser than the Cholesky factor the simple pattern union computation may be more efficient. It may also be possible to derive a fast symbolic factorization algorithm for B but at present none is known. The second application of a special purpose symbolic factorization algorithm is the identification of the elements in B with positions in P which are not needed to compute elements of R, i.e., unnecessary computations. This will be discussed in Section 5.4 but, at present, no fast algorithm exists. As a result, we can view the performance results of CIMGS presented in this section as inferior to what is most likely possible. Fortunately, the results show that even this potentially inferior implementation results in significant savings compared to ICGS and IMGS with the various pattern selection and column ordering strategies presented in earlier chapters.

5.3.1 CIMGS performance with different pattern selection strategies

We first compare the performance of CIMGS to that of IMGS and ICGS for dynamic and static pattern selection with no reordering of the columns of A .

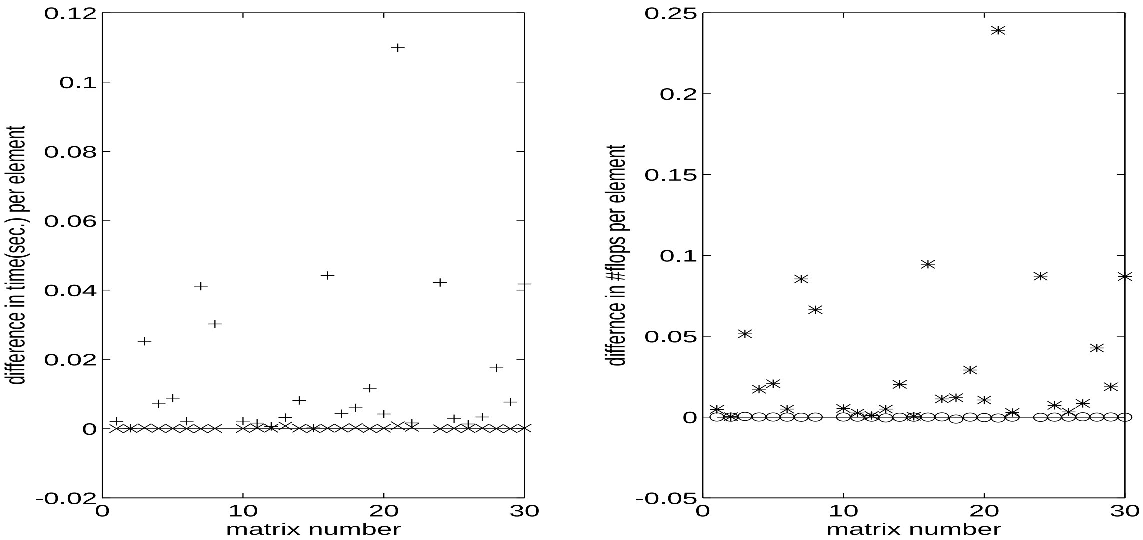
CIMGS has an advantage over IMGS in that static and dynamic pattern selection are easily inserted and there is not the profound difference in efficiency between the two strategies that is seen for IMGS. This can be seen in Figure 5.1 where the following quantities are plotted:

$$\frac{\frac{t^d_{IMGS}}{nnz(R)^d} - \frac{t^s_{IMGS}}{nnz(R)^s}}{\frac{t^d_{IMGS}}{nnz(R)^d}}, \frac{\frac{t^d_{CIMGS}}{nnz(R)^d} - \frac{t^s_{CIMGS}}{nnz(R)^s}}{\frac{t^d_{CIMGS}}{nnz(R)^d}}, \frac{\frac{f^d_{IMGS}}{nnz(R)^d} - \frac{f^s_{IMGS}}{nnz(R)^s}}{\frac{t^d_{IMGS}}{nnz(R)^d}} \quad and \quad \frac{\frac{f^d_{CIMGS}}{nnz(R)^d} - \frac{f^s_{CIMGS}}{nnz(R)^s}}{\frac{f^d_{CIMGS}}{nnz(R)^d}},$$

where t^s_{IMGS} and t^d_{IMGS} are the times for the factorization of IMGS using static and dynamic pattern selection respectively, t^s_{CIMGS} and t^d_{CIMGS} are the times for the CIMGS factorizations, f^s_{IMGS} and f^d_{IMGS} are the number of operations for the IMGS factorizations using static and dynamic pattern selection respectively, f^s_{CIMGS} and f^d_{CIMGS} are the times for the CIMGS factorizations, $nnz(R)^s$ and $nnz(R)^d$ are the number of elements of R when static and dynamic pattern selection is used respectively. For these experiments, the static pattern is chosen to be the non-zero pattern of the normal equations and $\epsilon = 0.02$ is used for dynamic pattern selection. The ratios plotted are the relative differences in work per element saved in R between static and dynamic pattern selection measured in terms of time and operations for both IMGS and CIMGS. There is a clear difference between the dynamic and static case for IMGS while virtually no difference for CIMGS. This is useful since the dynamic pattern selection strategy is useful for problems where the required preconditioning quality is not achievable using the simple static pattern selection strategies covered in this thesis.

CIMGS produces the preconditioner much more efficiently than IMGS for a given pattern selection strategy. Tables 5.1 and 5.2 show the number of matrices in various ranges of speedup measured in terms of operation count of CIMGS over IMGS for static and dynamic pattern selection. Note that for the static pattern selection only 29 matrices are included. This is due to the failure of CIMGS to produce a preconditioner due to numerical breakdown. The speedups based on time follow similar profiles. The data clearly shows substantial improvement in the production of the preconditioner – a factor of over 2000 in one case. As expected, the largest improvement is for dynamic pattern selection and, perhaps most importantly, there is no degradation compared to IMGS (except for the problem that causes breakdown).

ICGS was shown to be more efficient than IMGS in generating its preconditioner. Tables 5.3 and 5.4 compare ICGS to CIMGS. As expected, CIMGS does not improve as much in the preconditioning phase



$$+:\ \frac{\frac{t^d_{IMGS}}{nnz(R)^d}-\frac{t^s_{IMGS}}{nnz(R)^s}}{\frac{t^d_{IMGS}}{nnz(R)^d}},\qquad *:\ \frac{\frac{f^d_{IMGS}}{nnz(R)^d}-\frac{f^s_{IMGS}}{nnz(R)^s}}{\frac{t^d_{IMGS}}{nnz(R)^d}}$$

$$\times:\ \frac{\frac{t^d_{CIMGS}}{nnz(R)^d}-\frac{t^s_{CIMGS}}{nnz(R)^s}}{\frac{t^d_{CIMGS}}{nnz(R)^d}},\qquad \text{o}:\ \frac{\frac{f^d_{CIMGS}}{nnz(R)^d}-\frac{f^s_{CIMGS}}{nnz(R)^s}}{\frac{f^d_{CIMGS}}{nnz(R)^d}},$$

Figure 5.1 Difference of static and dynamic pattern selections in IMGS and in CIMGS

speedup	1-3	3-6	6-10	10-20	20-40	60-70	160-170
# matrices	5	4	5	11	2	2	1

Table 5.1 Number of matrices in selected ranges of the ratio of the IMGS operation count to the CIMGS operation count for the preconditioner using static pattern selection

compared to ICGS. In fact it degrades somewhat for some problems for both static and dynamic pattern selection. The degradation is not that severe – for the static case ICGS takes 80% of the operations of CIMGS and for the dynamic case for the best ICGS problem it requires 60% of the operations of CIMGS. Of course, the quality of the preconditioner can restore the preference to CIMGS for these problems.

In general, the quality of the preconditioners produced by CIMGS are the same as IMGS with the exception of the one problem for which the dynamic pattern selection fails and a set of problems for which CIMGS is unexpectedly superior. This can be seen in Figure 5.2 which plots for static and dynamic pattern selection the ratio of the difference in operation count for the iterative phase between IMGS and CIMGS to the operation count of the iterative phase of IMGS and a similar ratio for time. There are only 4 problems with differing time behavior and and 2 with differing operation count behavior. In all of these cases CIMGS is superior.

speedup	1-30	30-60	60-100	100-200	200-500	500-1000	>1500
# matrices	6	5	2	9	4	1	2

Table 5.2 Number of matrices in selected ranges of the ratio of IMGS operation to CIMGS operation count for the preconditioner phase with dynamic pattern selection

speedup	<1	1-5	5-7	7-10	10-20	35-45	190-200
# matrices	1	11	5	3	7	2	1

Table 5.3 Number of matrices in selected ranges of the ratio of ICGS operation count to CIMGS operation count for the preconditioning phase with static pattern selection

speedup	0.5-1	1-5	5-7	7-10	10-20	35-45	170-180
# matrices	3	8	5	3	7	2	1

Table 5.4 Number of matrices in selected ranges of the ratio of ICGS operation count to CIMGS operation count for the preconditioner phase using dynamic pattern selection

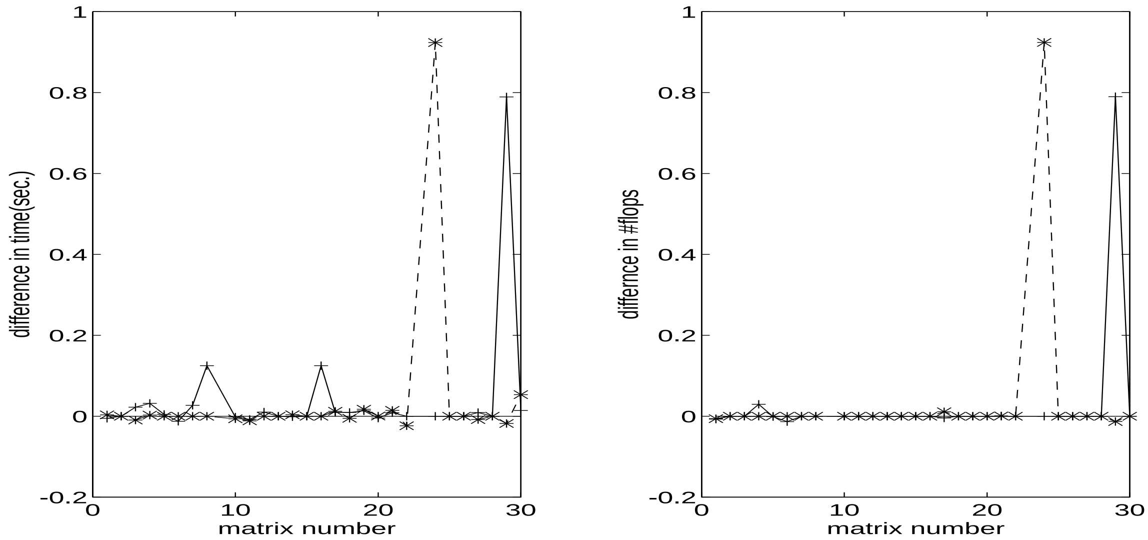


Figure 5.2 Relative time and operation count improvements of CIMGS over IMGS for dynamic (*) and static (+) pattern selection

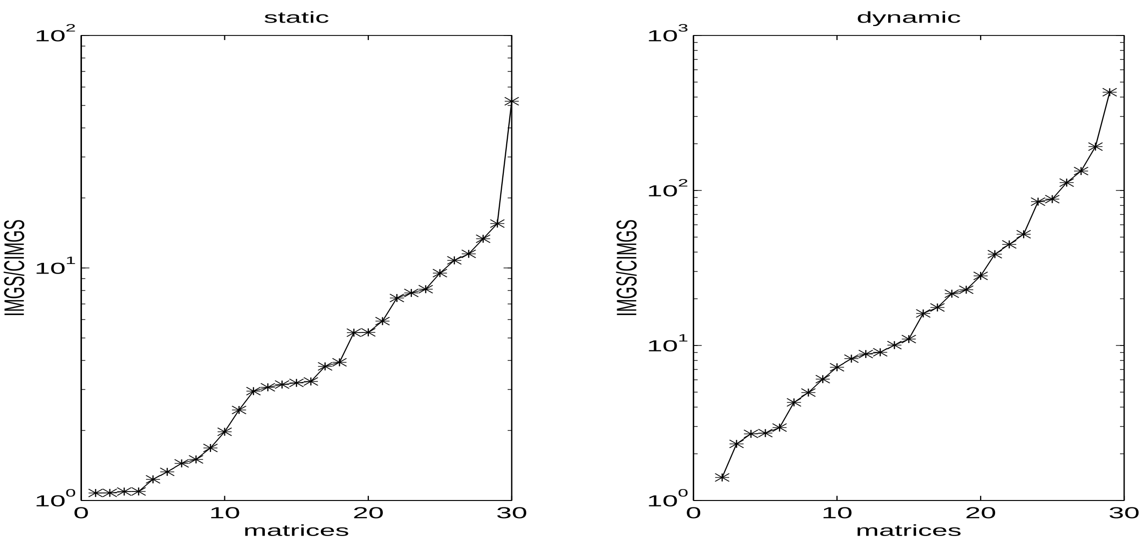


Figure 5.3 Ratio of total operation counts: IMGS to CIMGS (static and dynamic)

Of course the final metric of success is the overall reduction in operations and time to solve the least squares problems. Figure 5.3 demonstrates the reduction in total operations count to solve many problems by CIMGS compared to IMGS. A similar comparison to the more efficient ICGS is shown in Figure 5.4. ICGS is slightly better for 1 static problem and 2 dynamic problems but for the remainder CIMGS is the clear winner.

The techniques, discussed in the previous chapter, to reduce the density of R for static pattern selection – the numerical and structural methods – are also applicable to CIMGS. Figure 5.5 is an example of the numerical method of reducing the density of R on BELLMEDT. It shows the quantities $nnz(R)$, the number of iterations, the time for factorization (+), iterative phase (*) and total time (o), the number of operations for factorization (+), iterative phase (*) and total operation count (o) and the changes as α varies. Figure 5.6 is an example of the structural method of doing this.

As with IMGS and ICGS the performance of CIMGS with dynamic dropping is sensitive to the choice of ϵ. For example, when $\epsilon = 0.02$ we find that for BELLMEDT the factorization phase of CIMGS takes most of the time to solve the problem. We can then increase the size of ϵ to reduce the cost of factorization and to improve the overall performance, see Figure 5.7. On the other hand, for the problems SAYLR4 and PORES_2, the preconditioning quality of R is not good enough and the iterative phase takes an excessive amount of time. Reducing ϵ solves the problem and improves performance, see Figures 5.8 and 5.9.

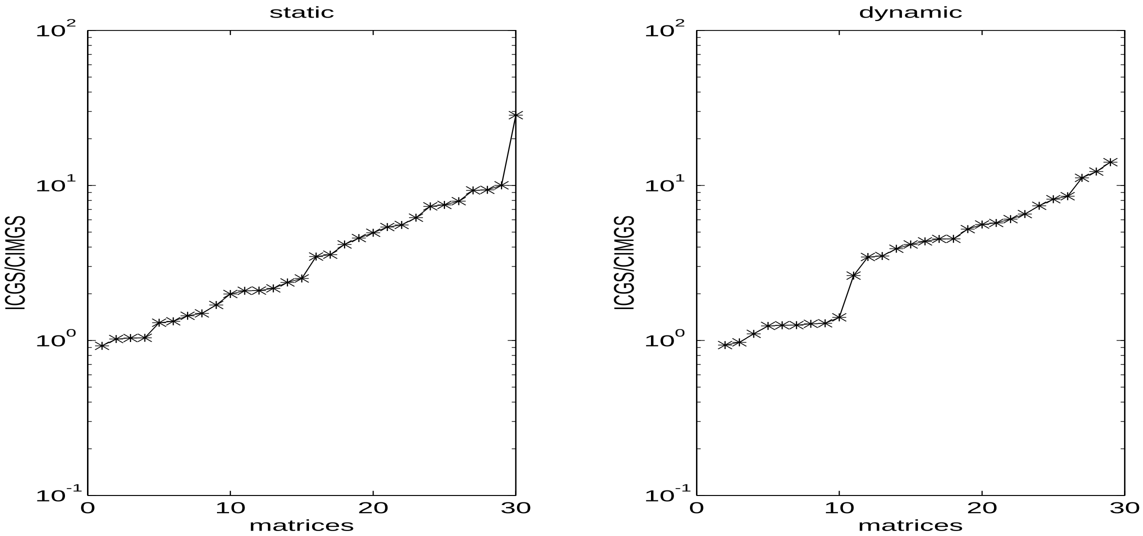


Figure 5.4 Ratio of total operation counts: ICGS to CIMGS (static and dynamic)

ordering	none	CW	ND	RCM	MD
failure rate	1/30	3/30	2/30	3/30	3/30

Table 5.5 Failure rates for CIMGS with column orderings to compute R

5.3.2 Column ordering and the performance of CIMGS

In this section we consider the effect of the four column reordering strategies that were applied to IMGS and ICGS in Chapter 4. Three of the four orderings were motivated by reducing the fill-in in the Cholesky factorization of the normal equations and therefore in R and indirectly in Q. Since the structure of CIMGS is similar to the structure of Cholesky factorization in some sense, we expect that these ordering methods would improve the performance of CIMGS as well. The fourth column ordering, which is based on the non-zero counts of the columns of A, was motivated by reducing fill-in in Q and would not be expected to be as effective.

We first examine the effects of the ordering methods on CIMGS with static pattern selection. Unfortunately, the orderings cause numerical breakdown of CIMGS on some of the matrices. The failure rates for each method is shown in Table 5.5.

Figures 5.10, 5.11 and 5.12 summarize the effects of the column ordering methods on CIMGS. The data for those problems that CIMGS with or without ordering fails to compute R are not shown in the figures. When it is reordered by the ND or MD orderings, the matrix ORSREG_1 causes difficulties even though the CIMGS factorization succeeds. The number of iterations required to converge to the required accuracy increase compared to the case without column ordering. Without ordering 312 iterations are

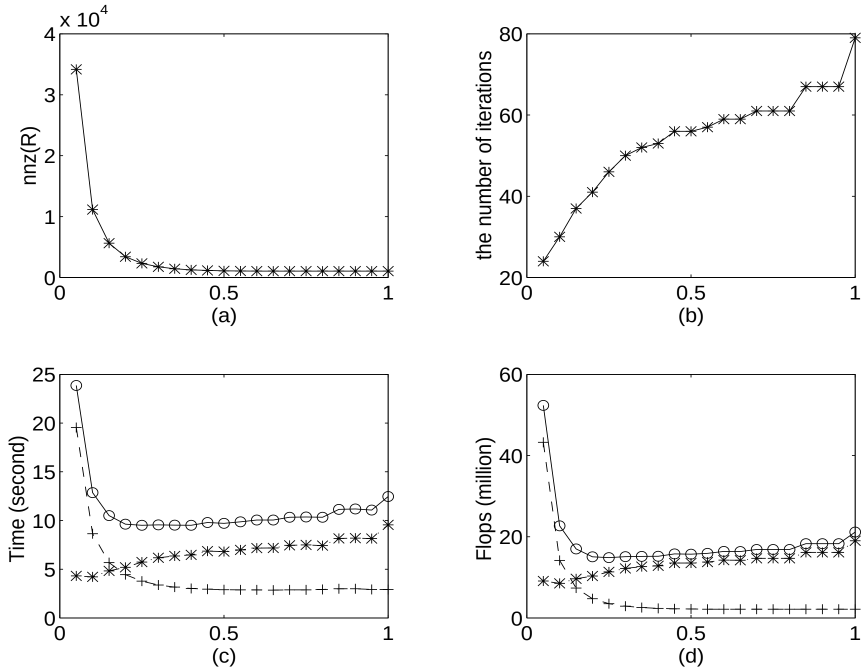


Figure 5.5 Example of numerical strategy for reducing the density of R of static pattern BELLMEDT

needed. Column ordering increases this to 4190 iterations for the ND ordering and 4185 iterations for the MD ordering. This problem is also excluded from the figures.

Figure 5.10 compares the number of operations needed to compute the CIMGS factorization with and without ordering. The MD ordering is the most effective method since it reduces the operation count for all of the problems. The CW ordering is the least effective although it still is able to reduce the operation count for most of the problems tested, the reductions are moderate. This is expected since the CW ordering is motivated by reducing the fill-in in Q and does not address the reduction of fill-in in R.

Figure 5.11 plots that ratio of number of iterations required by the reordered matrices to the number required by the original matrices. In general, the trend is to a reduction in the number of iterations but this is more true for some orderings than others. There are problems for which none of the ordering methods reduce the iterations, e.g., STRAT8 and PDE_9511. Once again the MD ordering tends to produce the most favorable results in that it reduces (or at least does not degrade) the iteration counts on the largest number of matrices. The other three methods are equally disappointing.

Since the trend is that the orderings reduce the operations in the CIMGS factorization phase and are also likely to reduce the number of iterations, the total computation cost of solving the linear least squares problems should, in general be reduced by the column orderings. The actual results are shown in Figure 5.12. The MD ordering is superior overall, reducing or not increasing the operation count

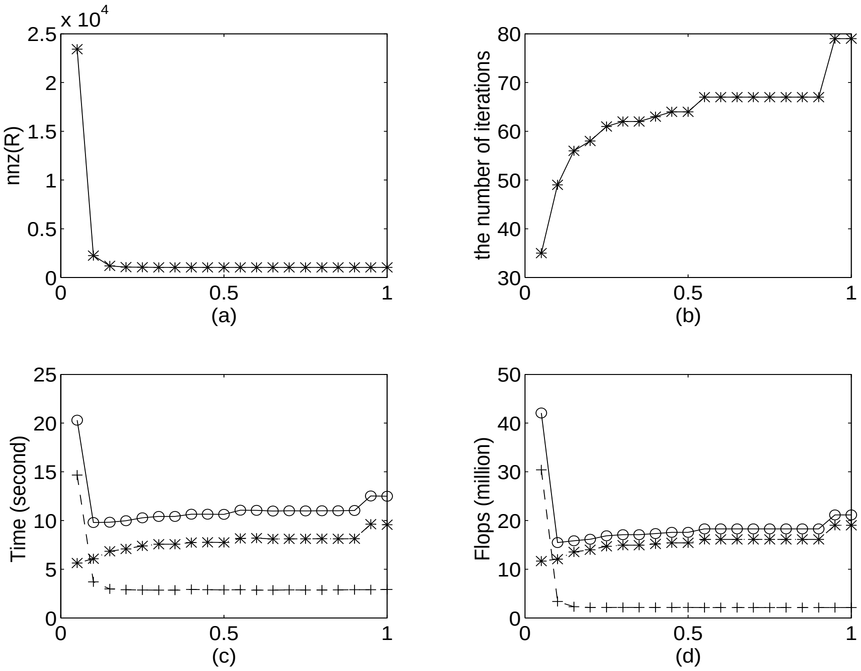


Figure 5.6 Example of structural strategy for reducing the density of R of static pattern BELLMEDT

for 24 of the 26 plotted problems. The others, ND, RCM and CW degrade on 5, 6 and 7 problems respectively and do not deliver the same level of improvement as MD on the remaining problems. Of course, this improvement did come at a price. The numerical robustness of CIMGS for certain problems was degraded by applying the column orderings.

When the orderings are applied to CIMGS with dynamic pattern selection, the density of R is affected since the order of the operations, and therefore the observed magnitudes at the point of dropping, change. Figure 5.13 shows that the number of non-zero elements in R is reduced on the majority of the problems by the CW, RCM, and MD orderings. The ND ordering increases and decreases the number of elements in R for a similar number of matrices.

Figure 5.14 shows that the ordering methods reduce the number of operations in the CIMGS factorization using dynamic pattern selection. But, comparing with the static results, we see that they are less effective.

As for the iterations, the MD and RCM orderings reduce the iterations for most of the problems. The ND and CW orderings tend to increase the number of iterations on the largest number of problems, see Figure 5.15.

The total cost of solving linear least squares problem using CIMGS preconditioning and dynamic pattern selection is reduced for most problems with any of the orderings, see Figure 5.16. The MD

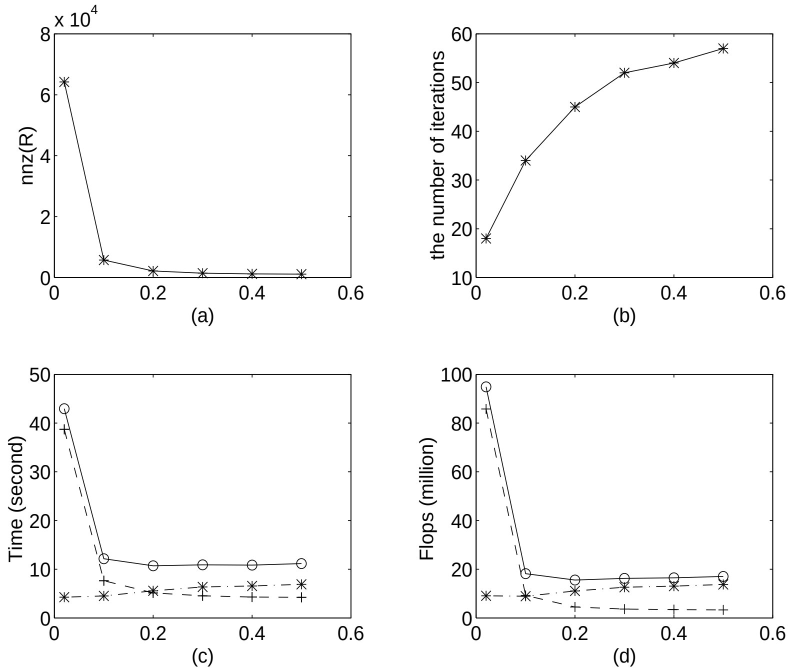


Figure 5.7 Example of increasing ϵ to reduce the cost of factorization phase of CIMGS BELLMEDT

ordering tends to improve the most and degrade the least on the problems whose total operation count decrease and increase respectively.

Based on the above discussion we conclude that the four ordering methods tested in this study do reduce the computation cost of CIMGS preconditioning and hence the total computation cost of solving the problems. They are not as effective in the iteration phase as in reducing the cost for the computation of preconditioner R, but they tend not to degrade the iteration phase substantially. The cost for all of this is the increased likelihood of numerical breakdown of the CIMGS factorization. One possible solution to this problem is the development of a CIMGS hybrid with IMGS or MGS. If a degradation in robustness is detected then a switch could be made to the more reliable (and costly) IMGS or MGS algorithms.

5.4 Symbolic analysis of CIMGS

In Section 5.3 we mentioned two uses for symbolic factorization in the implementation of CIMGS:

(1) determining the non-zero structure of B from which R is extracted,

(2) and distinguishing between the elements in B whose positions are in P that are necessary to compute R and those that are unnecessary.

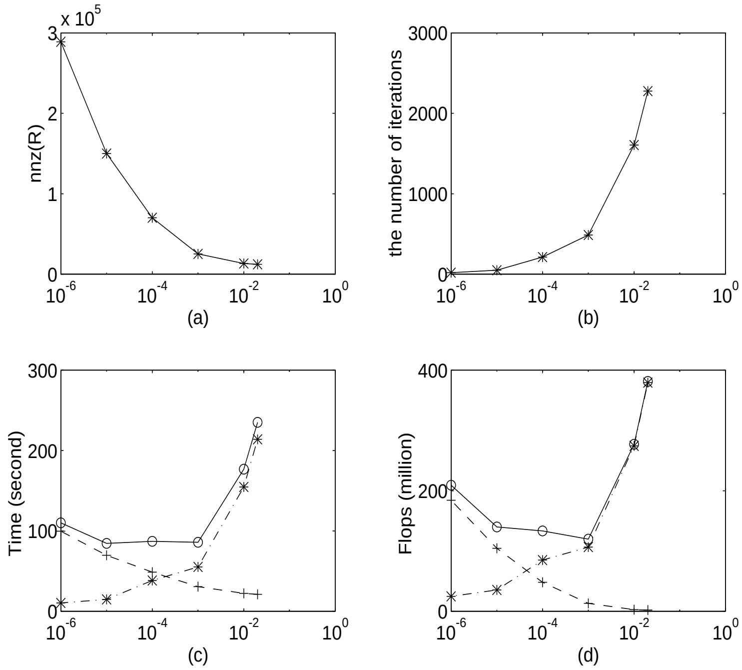


Figure 5.8 Example of decreasing ϵ to reduce the cost of iterative phase of CIMGS for SAYLR4

The first application requires a fast symbolic factorization like that available for sparse Cholesky factorization. The second is interesting from a characterization point of view so, regardless of cost, the necessary and unnecessary elements in P can be identified. If, additionally, a fast symbolic method for doing this identification is possible it would be valuable in creating a very efficient implementation of CIMGS. This Section examines the second of these issues. A static pattern selection strategy is assumed throughout.

We start the study with the following observations:

> **Observation 1**: Unlike IC factorization, CIMGS uses more positions in the matrix than those in the set of non-zero positions in R.

For example, suppose the normal equations of an $m \times 4$ matrix A has the non-zero pattern shown in (a) of Figure 5.17. The sparsity pattern of the corresponding factor R is shown in (b) of the same figure. The elements (1,3) and (1,4) are non-zeros in the normal equations of A but will be dropped in the computation. For IC, they can be dropped before beginning of the algorithm, because they will not affect the value of R. On the other hand, in CIMGS the value in (1,3) in the normal equations will affect the value in (2,3) because (1,2) is in R and (2,3) needs to be updated by the values of (1,2) and (1,3).

> **Observation 2**: Some of the non-zero elements of the normal equations do not affect the result of the CIMGS.

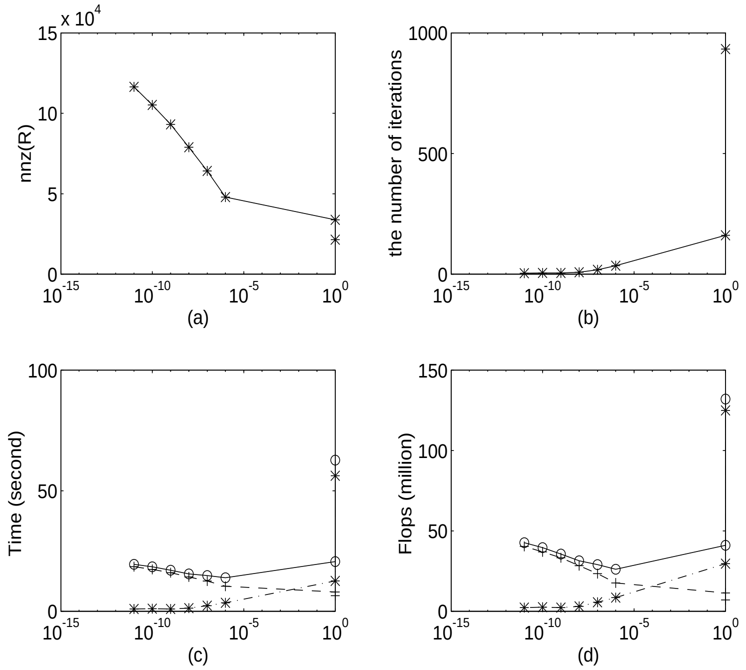


Figure 5.9 Example of decreasing ϵ to reduce the cost of iterative phase of CIMGS for PORES_2

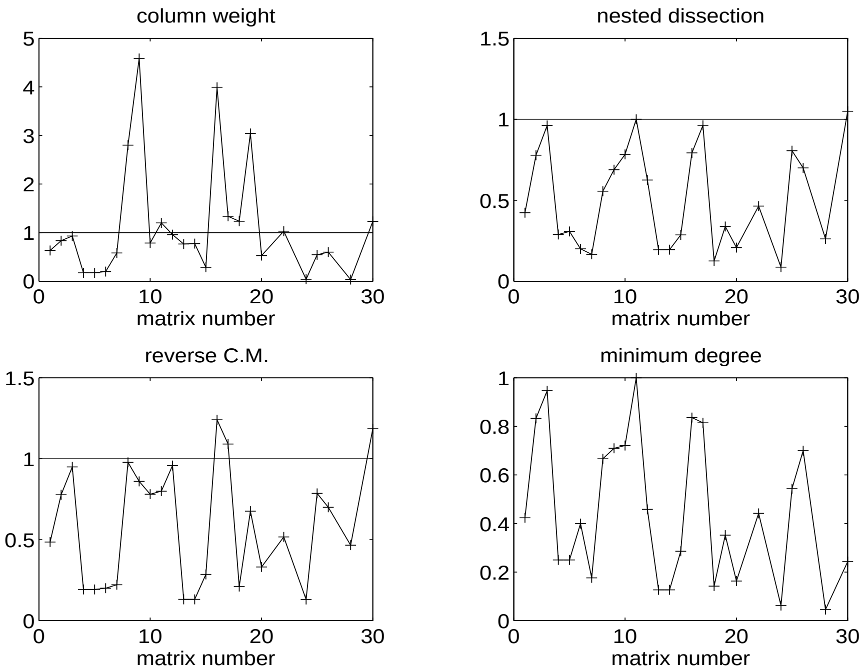
This can be illustrated by the same example used for Observation 1. In CIMGS, position (1,4) will affect the value of elements (2,4) and (3,4). But these two elements will eventually be dropped, so the updating is not needed and the element (1,4) can be dropped at the very beginning as would be done for an IC factorization. The updating of (2,4) and (3,4) requested by the CIMGS algorithm are actually unnecessary computations. The positions that will affect the value of R are shown in (c) of Figure 5.17.

Observation 3: Some 'fill-in' positions, ones that are neither in non-zero positions of R nor in non-zero positions of the normal equations, are needed for CIMGS.

Again using the example for Observation 1 for illustration, if the element (3,4) is added into R as shown in (b) of the Figure 5.18, then the computation of the fill-in element (2,4) is needed because it will affect the value of $(3,4)$ which is in R. Then the positions affecting the value of R are all the positions in the upper triangle as shown in (c) of Figure 5.18.

These observations show that the allocation of a static data structure for CIMGS is not a simple task. The set of the elements needed for CIMGS will be, in general, a subset of the set of non-zeros of the complete Cholesky factorization, which contains the set of the elements of R. We can not simply adapt the algorithm for Cholesky to CIMGS because the latter has two types of elements: those in P and those not in P, and they function very differently in the factorization.

To establish ways of predicting the space for the CIMGS factorization, some definitions are first needed:



flops(CIMGS with ordering)/flopsCIMGS

Figure 5.10 Comparison of the number of flops for the computation of the CIMGS factorization using static pattern selection with and without orderings

> **Sufficient positions for CIMGS**: A set of positions in the upper triangular factor is said to be sufficient if none of the positions outside the set will affect the factorization of the CIMGS.

For example, the non-zero positions for complete Cholesky factorization is a trivial example. This set of positions keeps all the possible non zero elements which are either useful or useless.

> **Necessary positions for CIMGS:** A set of positions in the upper triangular factor is said to be a necessary position set if the value of every element in the set will affect the factorization R.

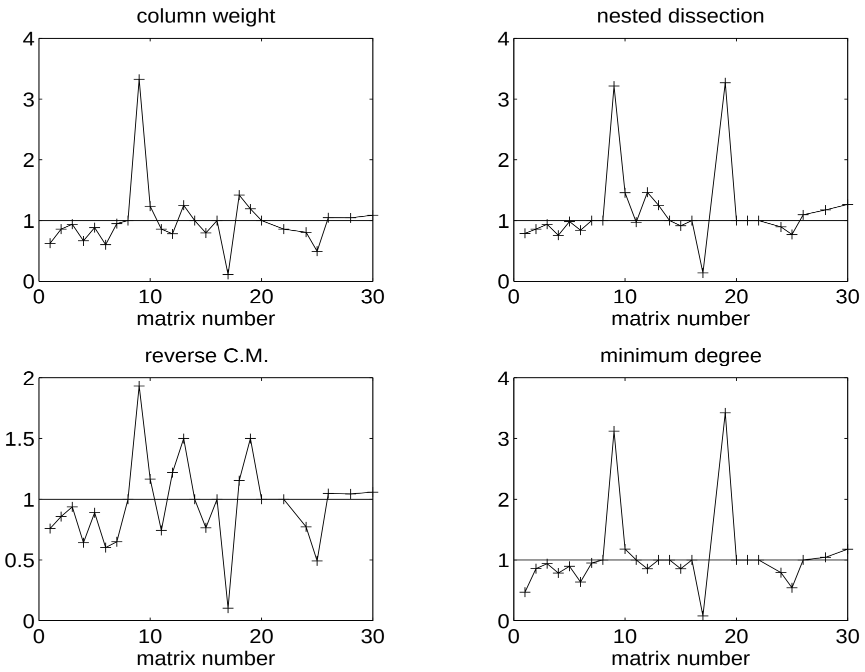
For example, the zero position set in R is a trivial example of a necessary position set.

If we can efficiently determine a sufficient and necessary position set, then by eliminating all the unnecessary computations we may make the computation of CIMGS more cost–effective.

As with complete Cholesky factorization, different versions of the CIMGS result by rearranging the order of computations. The following is a triad version of the CIMGS factorization:

begin

Algorithm [R]=CIMGS[B,P]



niter(CIMGS with ordering)/niterCIMGS

Figure 5.11 Comparison of the number of iterations by CGLS preconditioned by CIMGS using static pattern selection with and without orderings

for $i = 1$ **to** n

  for $k = 1$ **to** $i-1$

    if $(k,i) \in P$ **then**

      $B(i,i:n) = B(i,i:n) - B(k,i) * R(k,i:n)$

    else

      $B(i,i:n) = B(i,i:n) - B(k,i) * B(k,i:n)$

    endif

  endfor

  for $j = i$ **to** n

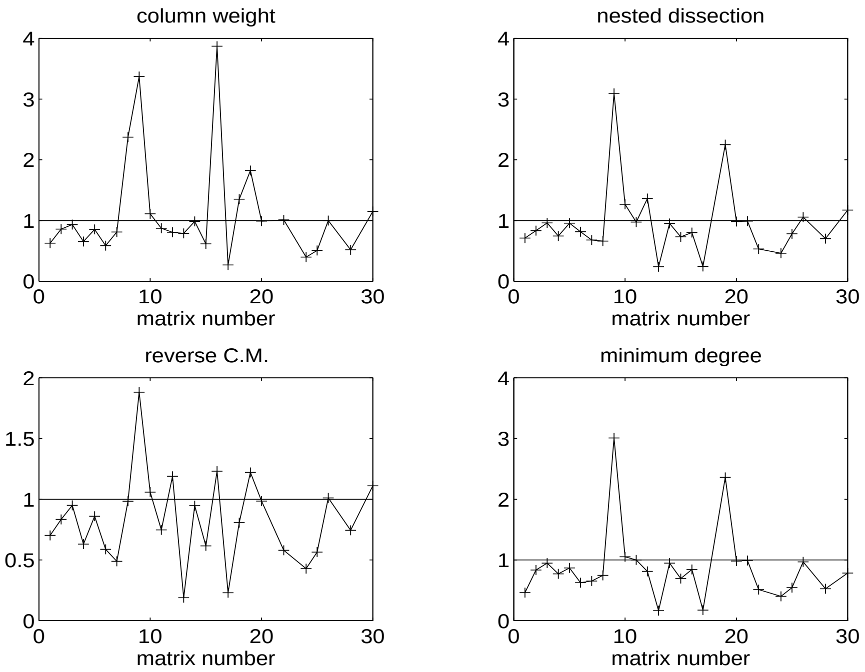
$$R(i,j) = \begin{cases} 0 & (i,j) \in P \\ B(i,j) & (i,j) \notin P \end{cases}$$

  endfor

endfor

end

At each step i, the i^{th} row of B, B_i, is a linear combination of the previous rows of R and B. So at each step previous rows are read and the current row is updated using a sparse triad. The remainder of



niter(CIMGS with ordering)/niterCIMGS

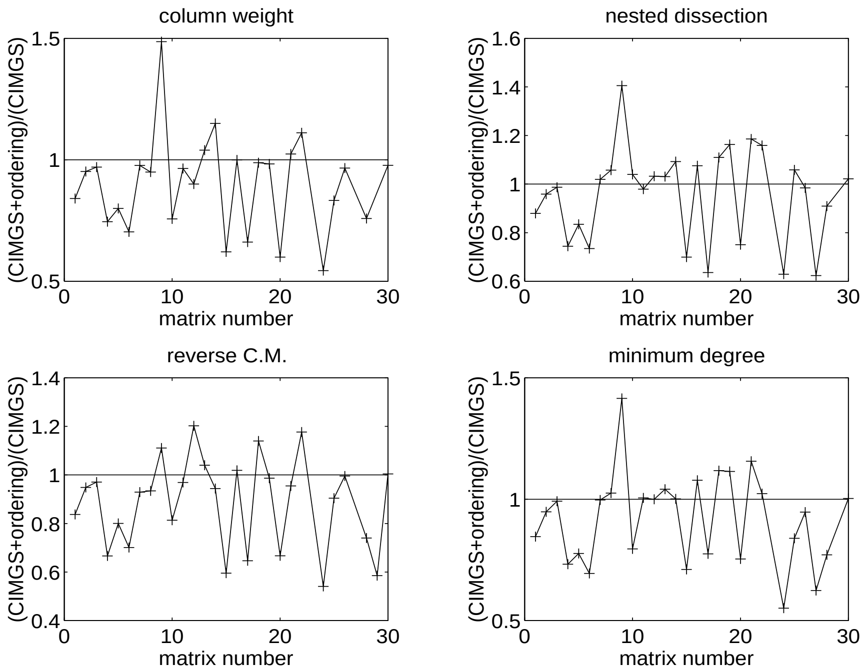
Figure 5.12 Ratio of the total number of operations to solve the test problems using CIMGS with static pattern selection with orderings

B is unchanged. The factor R is obtained by extracting elements from B in the set of allowed non-zero positions. The storage space for computing row i of B is thus the size of the union of some of the previous non-zero patterns. The update condition can be restated to show the two basic types of patterns that are involved. The pattern of of B_i is the union of rows R_j where $(j,i) \in nz(B)$ and $(j,i) \notin nz(R)$, and rows B_j if $(j,i) \in nz(R)$, $j = 1, 2, ..., i-1$ and $nz(C)$ denotes the non-zero position set of a matrix C. This analysis provides a simple algorithm to compute a sufficient set of non-zero positions for B in CIMGS.

For $i = 1, 2, ..., n$, let S_i be the set of non-zero positions of the row B_i of the CIMGS algorithm described above and assume that no exact cancellation occurs. Let T_i be the corresponding set for row R_i. Then we have

$$S_i = \left\{ \bigcup_{(j,i)\in P,\ 1\le j<i} T_j \right\} \bigcup \left\{ \bigcup_{(j,i)\notin P \ and \ (j,i)\in S_j,\ 1\le j<i} S_j \right\}$$

The set $S(A) = \bigcup_{i=1}^{n} S_i$ is the set of non-zero positions of B that are generated during the application of CIMGS to the coefficient matrix A of a linear least squares problem. On the other hand, this set may not be a necessary position set. There may be some elements that do not affect the computation of R. For example, let the normal equations of A have the sparsity pattern shown in Figure 5.19. If the



nnz(CIMGS with ordering)/nnzCIMGS

Figure 5.13 Comparison of the number of non-zero elements in R for CIMGS using dynamic pattern selection with and without orderings

pattern of R is chosen as the normal equations, then the position set S will be the set of all positions in the upper triangle. We know that not all of the fill-in elements are necessary because some will be dropped and more importantly they do not affect the final R. However, since $S(A)$ is a sufficient set we know that it contains all of the necessary elements to compute the factorization. Our task is to characterize these elements so as do define a way to isolate the set that determines with a minimum of space the CIMGS factorization.

In order to determine the sufficient and necessary position set for CIMGS, we first establish a graph model for the analysis.

Theorem 16 states conditions for an element of $S(A)$ to be a necessary element.

Theorem 16 *Let $S(A)$ is the sufficient set described above for the CIMGS factorization of the coefficient matrix A of a linear least squares problem using drop set P, $N(A)$ be the set of* all *necessary elements of A and $(i,j) \in S(A)$. If $(i,j) \in N(A)$ then one of the two following statements must be true:*

1. *$(i,j) \notin P$*
2. *$(i,j) \in P$ and there is a $(i,k) \notin P$ such that $(k,j) \in N(A)$ with $i < k < j$ or $(j,k) \in N(A)$ with $i < j < k$.*

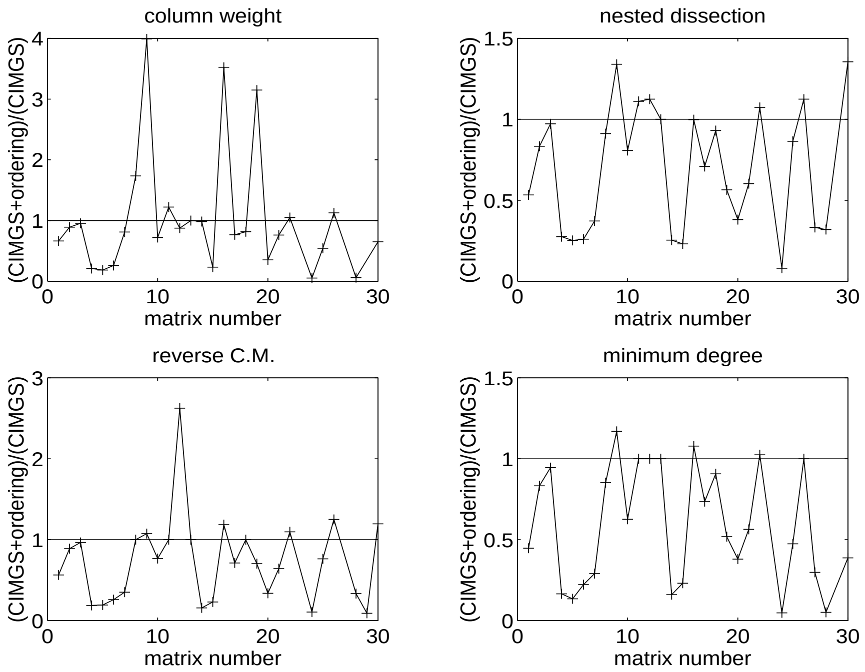


Figure 5.14 Ratio of the number of operations in CIMGS factorization using dynamic pattern selection with and without orderings

Proof: If $(i,j) \notin P$ then it is by definition a necessary element.

If $(i,j) \in P$, B_{ij} will be updated during the CIMGS factorization because $(i,j) \in S(A)$. If condition (2) is satisfied, then by the definition of the update process, the value of B_{ij} will affect the value of $B_{kj} \in N(A)$ with $i < k < j$ or the value of $B_{jk} \in N(A)$ with $i < j < k$. If the affected element is necessary then so is (i,j). If there is no $(i,k) \notin P$ which satisfies condition (2) then there is no way for the value of B_{ij} to affect the value of R and therefore $(i,j) \notin N(A)$. □

This result provides the basis for a procedure to determine $N(A)$. The procedure can be stated as a recursive algorithm to check if an element (i,j) is in $N(A)$ as follows:

Function check(i,j,P)

begin

if $(i,j) \notin P$ **then**

$check = $ **true**

elseif $\exists\,(i,k) \notin P,\ i < k < j$ **then**

$check = check(k,j,P)$

elseif $\exists\,(i,k) \notin P,\ i < j < k$ **then**

$check = check(j,k,P)$

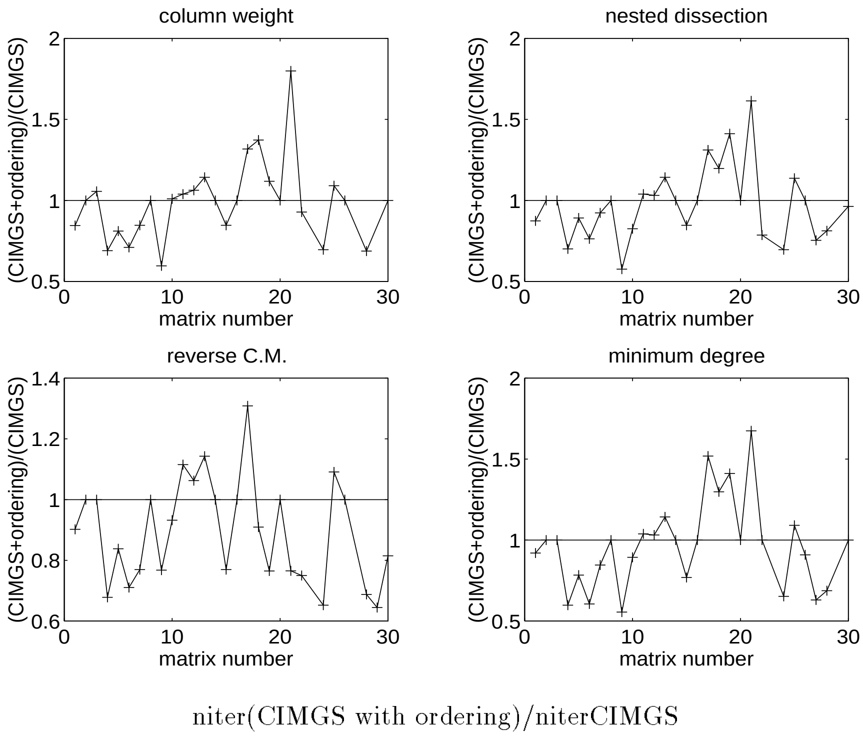


Figure 5.15 Ratio of the number of iterations by CGLS preconditioned by CIMGS using dynamic pattern selection with and without orderings

else

  check = **false**

endif

end

The algorithm terminates with a logical value because it recursively calls the function with the increasing values of the parameters and the size of the matrix is finite. Of course, applying this to all elements of $S(A)$ (or any other sufficient set) generates the following algorithm which determines $N(A)$:

Algorithm [N]= suf_nec[A,P]

begin

$N(A) = \emptyset$

compute sufficient set $S(A)$.

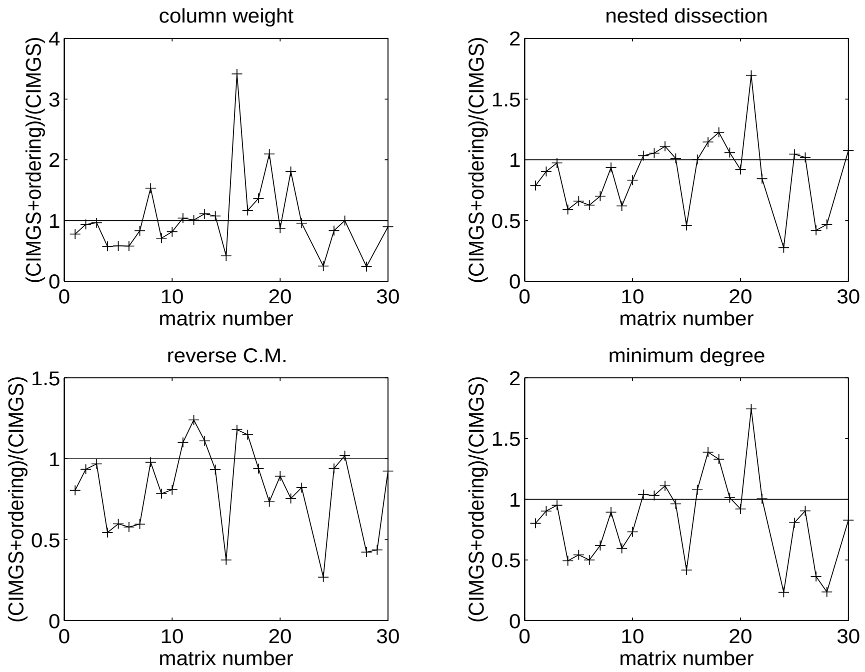
for $(i, j) \in S(A)$

  if *check(i,j,P)* **then**

    $N(A) = N(A) \cup (i, j)$

  endif

endfor



flops(CIMGS with ordering)/flopsCIMGS

Figure 5.16 Ratio of the total number of operations to solve the test problems using CIMGS with static pattern selection with and without orderings

end

The algorithms above to compute $S(A)$ and $N(A)$ are clearly brute force methods and not fast symbolic factorization algorithms of the type used for the Cholesky factorization and possibly needed for a very efficient CIMGS code. At present, it is not clear if a fast symbolic factorization algorithm to compute $N(A)$ of CIMGS is possible. However, we can apply the algorithms to our test problems and compare $S(A)$, $N(A)$ and the pattern of the complete and incomplete Cholesky factorizations to discuss the efficiency of CIMGS.

Let $nz(N(A))$ be the size of the set $N(A)$. Let $nz(R)$ be the number of non-zero elements in R. We know that CIMGS needs more space for the computation than the space for the storage of R. We compare these two quantities by computing the quantity $(nz(N(A)) - nz(R))/nz(N(A))$. This shows the percentage of the extra space needed for the CIMGS factorization compared to IC based on the non-zero pattern of R.

Let $nz(S(A))$ be the size of the set $S(A)$. The quantity $(nz(S(A)) - nz(N(A)))/nz(S(A))$ tells the percentage of the unnecessary positions in $S(A)$ and therefore relates to the unnecessary cost of CIMGS without symbolic analysis. This cost can be measured explicitly if we let $flops1$ and $flops2$ be the number

$$\begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} \times & \times & \times & \times \\ & \times & \times & \\ & & \times & \\ & & & \times \end{pmatrix} \end{array} \qquad \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \\ & & & \times \end{pmatrix} \end{array} \qquad \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} \times & \times & \times & \\ & \times & \times & \\ & & \times & \\ & & & \times \end{pmatrix} \end{array}$$

(a) (b) (c)

(a): Pattern of normal equations.
(b): Pattern of R.
(c): Pattern of sufficient and necessary positions.

Figure 5.17 Example of Observation 1

$$\begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} \times & \times & \times & \times \\ & \times & \times & \\ & & \times & \\ & & & \times \end{pmatrix} \end{array} \qquad \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \end{pmatrix} \end{array} \qquad \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} \times & \times & \times & \times \\ & \times & \times & \otimes \\ & & \times & \times \\ & & & \times \end{pmatrix} \end{array}$$

(a) (b) (c)

(a): Pattern of normal equations.
(b): Pattern of R.
(c): Pattern of sufficient and necessary positions.
$\otimes$ denote the fill-in positions

Figure 5.18 Example of Observation 3

$$\begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{pmatrix} \end{array} \qquad \begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \otimes & \otimes & \otimes & \otimes \\ & & \times & \otimes & \otimes & \otimes \\ & & & \times & \otimes & \otimes \\ & & & & \times & \otimes \\ & & & & & \times \end{pmatrix} \end{array}$$

(a) (b)

(a): Pattern of normal equations and R.
(b): Pattern of positions S.

Figure 5.19 Example of unnecessary positions in the set S

of floating point operations for the CIMGS factorization with and without the unnecessary computations. The percentage of the unnecessary computations is shown by the quantity $(flops1 - flops2)/flops1$.

Figure 5.20 shows the measurements of these quantities for CIMGS using the pattern of the normal equations. We see that for most of the matrices the price of the existence of the CIMGS factorization is an increase in the number of elements of over 60% compared to those required the store R (and therefore to compute an IC factorization). In some cases the space for CIMGS approaches that of a complete Cholesky computation. Even if this is the case, the number of operations for CIMGS could still be and is typically much less than Cholesky factorization. There is a benefit to the situation of CIMGS needing nearly the space of complete Cholesky and taking fewer operations. The fast symbolic Cholesky factorization can be used to generate a set of sufficient positions that is not too much of an upper bound and would therefore not generate a large amount of unnecessary work. When the space is much less than the Cholesky factorization, the efficiency of the present implementation depends on the number of non-zero elements being small enough so that a fast symbolic determination of $N(A)$ or some tight bounding set is not needed.

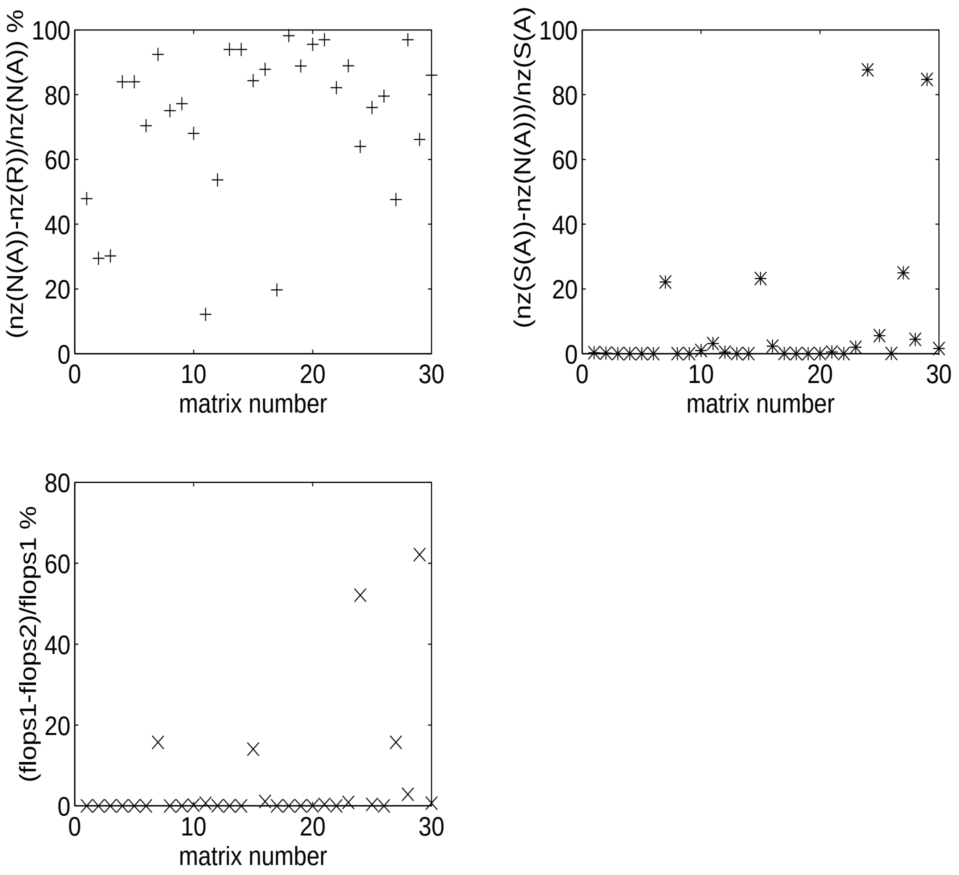


Figure 5.20 Examples of CIMGS without ordering with normal equation pattern

The data in Figure 5.20 also shows that for most of the test problems, the number of the unnecessary positions is usually small. But for some problems, it can be as much as 90%. In such case, a symbolic analysis to eliminate the unnecessary computation is a definite necessity. We can also see from the figure that the percentage of unnecessary computations is lower than the percentage of the unnecessary positions and small for most of the test problems.

As always, we consider the affect of column ordering on these aspects of CIMGS. Figure 5.21 shows the quantity $(nz(N(A)) - nz(R))/nz(N(A))$ of CIMGS with the four different orderings tested. The results are mixed but reductions do occur with the MD ordering, as usual, the most effective.

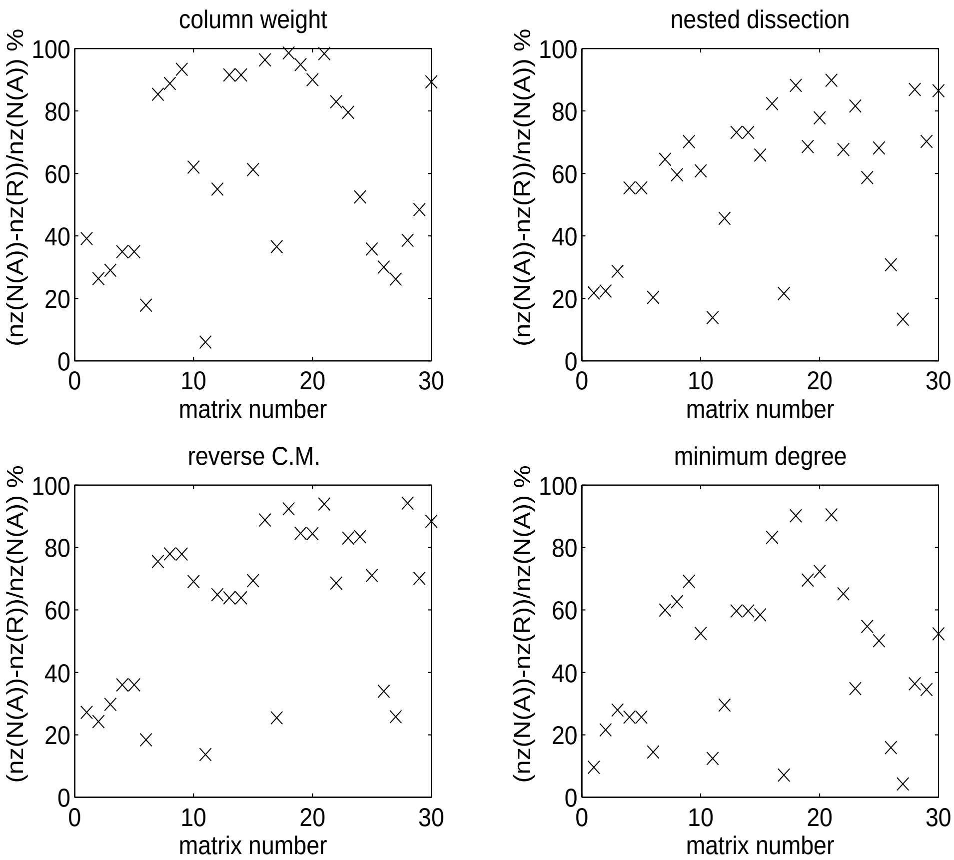


Figure 5.21 The percentage of the extra space for CIMGS with column ordering compared to IC

Figure 5.22 shows the quantity $(nz(S(A)) - nz(N(A)))/nz(S(A))$ of CIMGS with the four different orderings tested. Figure 5.23 shows the related quantity $(flops1 - flops2)/flops1$ and has similar implications. The interesting fact seen in these figures is that the RCM ordering makes $nz(S(A)) = nz(N(A))$. That is, after the columns are reordered by the RCM ordering, the set $S(A)$ is also a necessary position set. It is not clear if this is true in general or just for our test problems. Intuitively, the result is reasonable: RCM concentrates elements around the main diagonal and, therefore, the fill-in is concentrated in a smaller space making it more likely that they all influence one another despite being dropped. If such a result holds or nearly holds in general, the need for symbolic identification of $N(A)$ would be removed. There are factors that would mitigate this gain. If the RCM ordering increases $nz(N(A))$ while reducing the difference $nz(S(A)) - nz(N(A))$ or if the quality of the preconditioner suffers then there may not be an overall benefit.

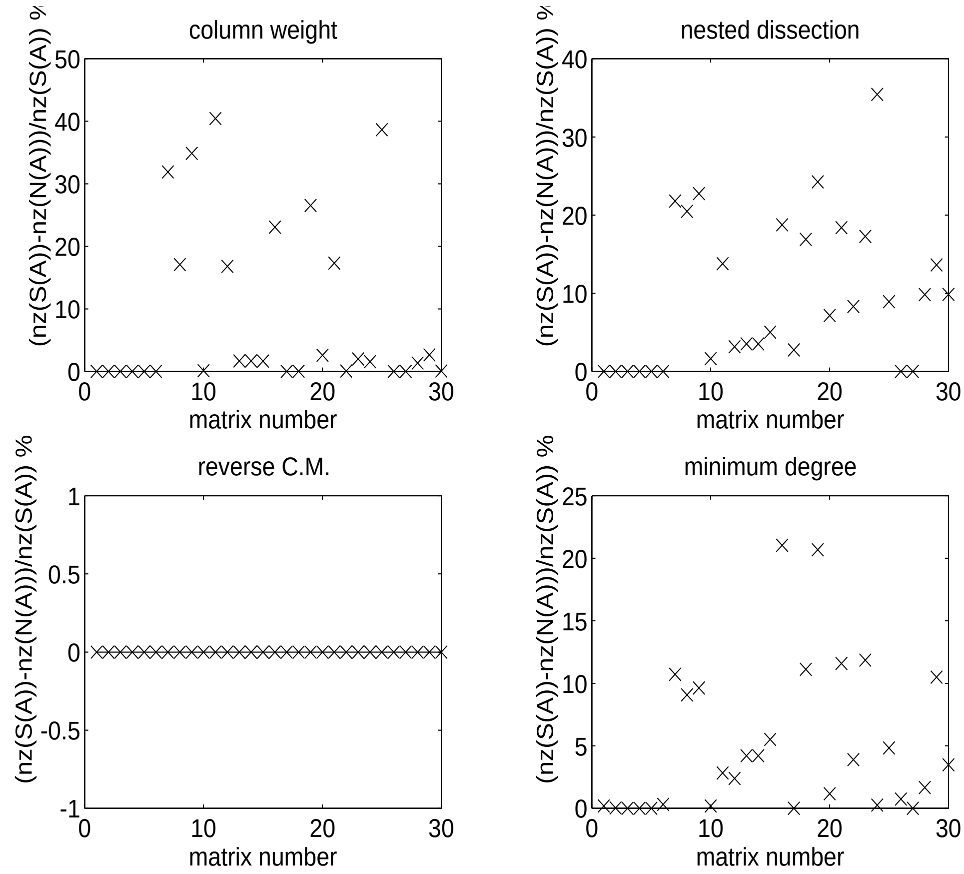


Figure 5.22 The percentage of the unnecessary positions in CIMGS with column ordering

The results also show that the column orderings often not only reduce $N(A)$ but also $S(A)$. For some problems the reduction of the set $S(A)$ is more significant than that of $N(A)$ so that the quantity $(nz(S)-nz(N(A)))/nz(S)$ is increased. Other than RCM ordering, the MD ordering is the most effective in reducing $nz(N(A))$.

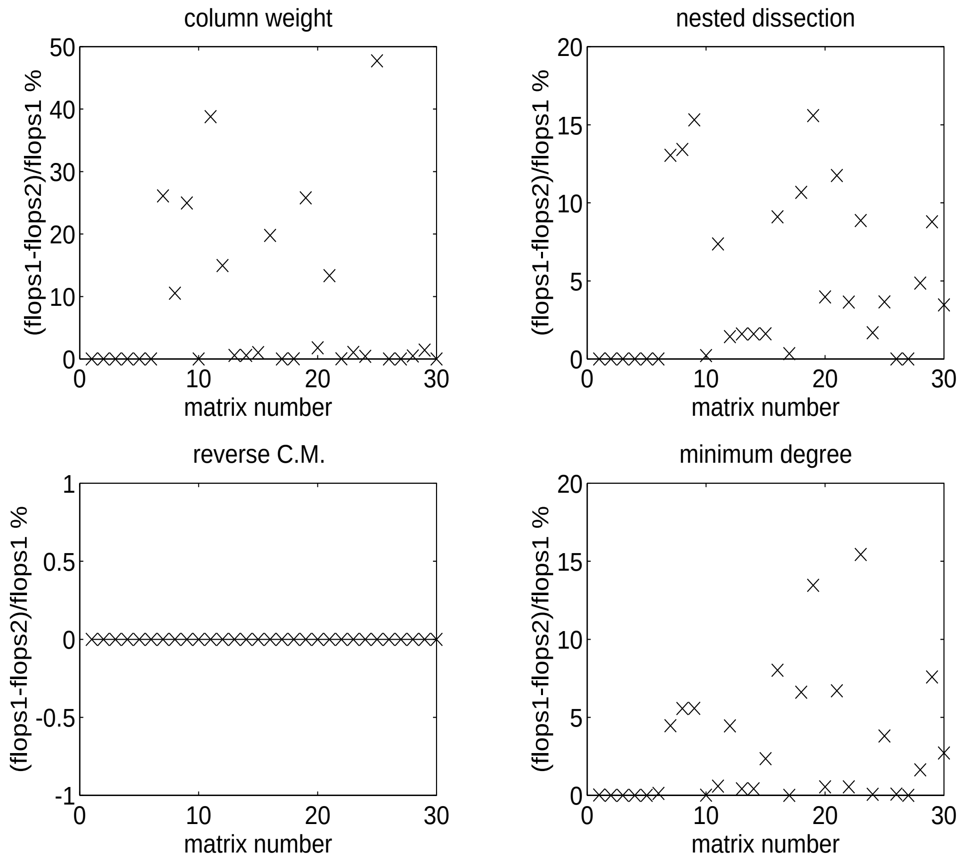


Figure 5.23 The percentage of the unnecessary operations in CIMGS with column ordering

CHAPTER 6

Incomplete Cholesky factorization with pattern modifications

6.1 Overview of problem

The incomplete Cholesky factorization is one of the most important preconditioners for iterative methods of solving symmetric positive definite linear systems [37]. Its major weakness is a lack of stability; to overcome this difficulty, several modifications have been suggested [47][44][61]. Those modification methods use information about the values of dropped elements and modify the process to assure the existence of the factorization. We refer to these modifications as numerical strategies. A completely different strategy is proposed here, which uses only information about the nonzero structure of the matrix to be factored and the structure of the target sparsity pattern for the IC factor R. This is referred to as a structural strategy, and is based on an understanding of the relationship between IMGS applied to A and IC applied to A^TA.

In this Chapter, we will use P as the set of non-zero positions for the incomplete factor, which differs from the usage in previous chapters. As shown in Chapter 3, if the non-zero position set $P \subset P_n$ is such that for any $1 \leq i \leq n$, $i < j$ and $i < k$, $(i,j) \in P$ and $(i,k) \notin P$ implies $(j,k) \notin P$, then IMGS applied to A and IC applied to $B = A^TA$ will generate the same triangular factors and the set P is said to have **property C**. This relationship has an important application: it leads to a method of modifying the sparsity pattern for IC, when IC with the original pattern breaks down, to guarantee the existence of the factorization. The modification can be done during the data structure allocation phase. This strategy is attractive for applications which need to solve multiple linear systems with the same nonzero structure but with different numerical values. It also provides a new direction of research for generally improving the quality of incomplete factorization preconditioning. Combining both numerical and structural information to compute higher quality preconditioners will be a topic for further research.

In the rest of this chapter, we will broaden our discussion to IC factorization on general symmetric positive definite matrices, and not restrict it to incomplete Cholesky factorization of normal equations.

As a result, we assume the matrix A refers to a symmetric positive definite matrix and not the full rank rectangular matrix of the previous chapters.

6.2 Property C revisited

We begin by stating the definition of property C in a different, but equivalent, form from the one above. Let $P_n = \{(i,j) | 1 \le i \le j \le n\}$ be the set of all the positions in a $n \times n$ upper triangular matrix. Let P be the set of non-zero positions in, R, the IC factor of a symmetric positive definite matrix A. In order for the IC factorization to complete the positions of the diagonal elements must be in P.

> A position set $P \subset P_n$ is said to have **property C** if for any $(j,k) \in P$, either of the following is true
>
> (1) for any $1 \le i < j$ $(i,j) \in P$ and $(i,k) \in P$;
>
> (2) for any $1 \le i < j$ $(i,j) \notin P$ and $(i,k) \notin P$.

A trivial example of a position set with property C is the position set of a dense block diagonal matrix. A small nontrivial example is shown in Figure 6.1 Note that informally the condition says that if you look at a non-zero position (j,k) then columns j and k have the same non-zero pattern from the j-th row to the 1-st row.

$$\begin{pmatrix} \times & & \times & \times & & \times & \\ & \times & & & \times & & \times \\ & & \times & \times & & \times & \\ & & & \times & & \times & \\ & & & & \times & & \times \\ & & & & & \times & \\ & & & & & & \times \end{pmatrix}$$

Figure 6.1 Position set P that has the property C

In Chapter 3, it was shown that if the set specifying the sparsity pattern of R has property C, the incomplete Cholesky factor of a symmetric positive definite matrix exists. The proof used the relationship between IMGS applied to a matrix and IC applied to the normal equations. Here another proof is given, based directly on property C and the relationship between the incomplete Cholesky factorization of a matrix and the Cholesky factorization of a principal submatrix.

Theorem 17 *Let the matrix $A \in \Re^{n \times n}$ be symmetric positive definite and $P \subset P_n$ be a position set. If P has property C then the incomplete Cholesky factorization of A using position set P exists.*

Proof: For any $1 \leq i \leq n$, let $S_i(j) = \{k|(k,i) \in P \quad and \quad k \leq j\}$ for $j < i$ be the set of row indices of positions in column i which have row index less than or equal to j. Let $P_i = \{(j,k) \in P_n | j \in S_i(i) and k \in S_i(i)\}$.

By property C, we know that $S_j(j) = S_i(j)$ for all $j \in S_i(i)$ and $P_i \subseteq P$. Now consider an element (k,j) where $j \in S_i(i)$ and $k \notin S_i(i)$. If $(k,j) \in P$ then property C is violated so we also have $(k,j) \notin P$ and position (k,j) must be a 0. This means that all of the non-zero positions in the principal submatrix defined by P_i are in P and therefore kept during the incomplete factorization. Therefore, the computation of the incomplete Cholesky factorization of the matrix A with the pattern P on positions in P_i is the same as the operation of Cholesky factorization on the principal submatrix of A defined by P_i. Since any principal submatrix of a symmetric positive definite matrix is also symmetric positive definite, the Cholesky factor of this principal submatrix exists. So the computation of the incomplete Cholesky factorization at any step i cannot break down. Therefore the whole process of incomplete Cholesky factorization can be completed. □

Since the purpose of this study is to provide an understanding of IC factorization, we are only interested in positions that will affect the computation of the IC factorization. For this purpose property C is too restrictive since it tends to create dense submatrices. We will define a new property which includes the notion of restricting our attention the non-zeros defined by the Cholesky factor of A.

For a given symmetric matrix A, let $U = (u_{ij})$ be the Cholesky factor of A, $A = U^T U$. Let $P^+(A)$ be the set of positions such that u_{ij} is not zero. We define the new property as follows

> The position set $P \subset P^+(A) \subseteq P_n$ has **property C+** if for any position $(j,k) \in P$ the following statement is true: for all pairs of positions in columns j and k, $[(i,j),(i,k)]$, such that $i < j$, $(i,j) \in P^+(A)$, and $(i,k) \in P^+(A)$, either of the following are true:
>
> (1) $(i,j) \in P$ and $(i,k) \in P$;
>
> (2) $(i,j) \notin P$ and $(i,k) \notin P$;

Theorem 18 *Let the matrix $A \in \Re^{n \times n}$ be symmetric positive definite, and suppose that position set $P \subset P_n$ has property C+. The incomplete Cholesky factorization of A using position set P exists.*

Proof: We will use the same technique as the proof of Theorem 17 – create a set of principal submatrices for which the pattern guarantees that the portion of the IC factorization on its elements is equivalent to a complete Cholesky on the submatrix. For any $1 \leq i \leq n$, let $V_i = \{j|(j,i) \in P\}$ be the set of row indices of elements of P in column i. Construct a set S_i in the following way: Let $S_i = V_i$, and then repeat the following computation until it converges: $S_i = S_i \bigcup_{j \in S_i} V_j$. This construction assures that for any $j \in S_i$, $V_j \subset S_i$. Let P_i be the set of positions (j,k) such that j and k are both in S_i.

We now show that for any $(j,k) \in P_i$, if $(j,k) \in P^+(A)$ then $(j,k) \in P$. Assume that $(j,k) \notin P$. Then there is $(j,l) \in P$ where $l \in S_i$; otherwise j would not be in S_i. Now consider the position

$(min\{l,k\}, max\{l,k\})$. Since $\{j,k,l\} \subset S_i$, by the definition of P_i we have $(min\{l,k\}, max\{l,k\}) \in P_i$. Since both (j,l) and (j,k) are in $P^+(A)$ it follows that $(min\{l,k\}, max\{l,k\}) \in P^+(A)$. Also because (j,l) is in P but (j,k) is not in P, and P has property C+, we have $(min\{l,k\}, max\{l,k\}) \notin P$. Now we have a position $(min\{l,k\}, max\{l,k\})$ which has the same status as (j,k): in P_i, in $P^+(A)$, but not in P. Note that $min\{l,k\} > j$ and $max\{l,k\} \geq k$. Apply this argument repeatedly, at each step we either get $(min\{l,k\}, max\{l,k\}) \in P$ and therefore prove by contradiction the assertion, or we repeat. Eventually, if we do not prove the assertion we reach a row in P_i that has only one off-diagonal element. This last element must be in P or the position would not be in P_i which again proves the assertion by contradiction. So the assumption that $(j,k) \notin P$ is not true and therefore $(j,k) \in P$.

On the other hand, we can show that any position (j,k) where $j \notin S_i$ and $k \in S_i$ will not affect the computation of the IC on position set P_i. For those $(j,k) \notin P^+(A)$, they would not affect the computation. Let $(j,k) \in P^+(A)$. If $(j,k) \in P$ then j will be in S_i by construction and this contradicts the assumption that $j \notin S_i$ and hence $(j,k) \notin P$. Therefore, (j,k) is either not in P or not in $P^+(A)$. Thus (j,k) will not affect the value of the elements in the position P_i during the computation of the incomplete Cholesky factorization.

Therefore, operations of the incomplete Cholesky factorization on A at positions P_i are equivalent to the operations of Cholesky factorization on the principal submatrix of A defined by P_i. Because any principal submatrix of a symmetric positive definite matrix is symmetric positive definite, Cholesky factorization of the principal submatrix of A defined by P_i can be completed, and hence the incomplete Cholesky factorization of A with pattern P exists. $\square$

This establishes a sufficient condition on P for the existence of an incomplete Cholesky factorization. In the next section, we discuss algorithms that modify a given position P so that the modified position set will have property C+. By using property C+ rather than the earlier property C we are able to work within the position set of the complete Cholesky factor rather than P_n – the position set of a dense upper triangular matrix

6.3 Algorithms for modifying the sparsity pattern for IC

There are many ways to modify a position set so that it will have property C+. Note that if there are any position patterns of the types shown in Figure 6.2 in P, property C+ is violated. To remove patterns that violate property C+, we can either add elements into P or remove elements from P, as shown in Figure 6.3. Lemma 8 provides a very simple way of modifying the position set so that it has property C+.

$$
\begin{array}{cc}
 & \begin{array}{ccc} i & j & k \end{array} \\
\begin{array}{c} i \\ \\ j \\ k \end{array} & \left(\begin{array}{ccc} & \times & \otimes \\ & & \\ & & \times \\ & & \end{array}\right)
\end{array}
\qquad
\begin{array}{cc}
 & \begin{array}{ccc} i & j & k \end{array} \\
\begin{array}{c} i \\ \\ j \\ k \end{array} & \left(\begin{array}{ccc} & \otimes & \times \\ & & \\ & & \times \\ & & \end{array}\right)
\end{array}
$$

(a) $\qquad\qquad$ (b)

$\times$ denotes the position that in P.

$\otimes$ denotes the position that in Cholesky factor R but not in P.

Figure 6.2 Patterns that do not have the property C+

$$
(a)\ \left(\begin{array}{cc} \times & \otimes \\ & \times \end{array}\right) \Rightarrow \left(\begin{array}{cc} \times & \times \\ & \times \end{array}\right) or \left(\begin{array}{cc} \otimes & \otimes \\ & \times \end{array}\right) or \left(\begin{array}{cc} \times & \otimes \\ & \otimes \end{array}\right)
$$

$$
(b)\ \left(\begin{array}{cc} \otimes & \times \\ & \times \end{array}\right) \Rightarrow \left(\begin{array}{cc} \times & \times \\ & \times \end{array}\right) or \left(\begin{array}{cc} \otimes & \otimes \\ & \times \end{array}\right) or \left(\begin{array}{cc} \otimes & \times \\ & \otimes \end{array}\right)
$$

$\times$ denotes the position in P.

$\otimes$ denotes the position in Cholesky factor R but not in P.

Figure 6.3 Modifications of the patterns

Lemma 8 *Let $A \in \Re^{n\times n}$, $P \subset P_n$ be a non-zero position set. Partition the set $\{1, 2, ..., n\}$ into subsets such that if $(i, j) \in P$ then i and j are in the same subset S_l. Define $\bar{P}$ in the following way. For any $(i, j) \in P^+(A)$, if $j \in S_k$ and $i \in S_k$, then $(i, j) \in \bar{P}$. Then $P \subset \bar{P}$, and $\bar{P}$ has property C+.*

Proof: Let $(i, j) \in P$; then i and j are in the same subset S_k. By the definition, $(i, j) \in \bar{P}$. For any $(i, j) \in \bar{P}$ and any $k < i$ such that (k, i) and (k, j) are in $P^+(A)$, if $(k, i) \in \bar{P}$ then we have k, i and j are in the same set S_l. By the definition of $\bar{P}$, $(k, j) \in \bar{P}$. Similarly we can show that if $(k, j) \in \bar{P}$ then we have $(k, i) \in \bar{P}$. Therefore $\bar{P}$ has property C+. $\square$

Figure 6.4 shows an example of A, P and $\bar{P}$ as defined in Lemma 8, where A is a full matrix.

The method described in Lemma 8 can add elements into the position set which are not necessary for property C+. This could unnecessarily increase the density of the preconditioner, which in turn may increase computation and storage requirements without improving the quality of the preconditioner.

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} &
\left(\begin{array}{ccccccc}
\times & & \times & & & & \\
 & \times & & & \times & & \\
 & & \times & \times & & & \\
 & & & \times & & \times & \\
 & & & & \times & & \times \\
 & & & & & \times & \\
 & & & & & & \times
\end{array}\right)
\end{array}
\qquad
\begin{array}{cc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} &
\left(\begin{array}{ccccccc}
\times & & \times & \times & & \times & \\
 & \times & & & \times & & \times \\
 & & \times & \times & & \times & \\
 & & & \times & & \times & \\
 & & & & \times & & \times \\
 & & & & & \times & \\
 & & & & & & \times
\end{array}\right)
\end{array}
$$

P $\qquad\qquad$ $\bar{P}$

$\times$ denotes the position that in the set. $S_1 = \{1, 3, 4, 6\}$, $S_2 = \{2, 5, 7\}$.

Figure 6.4 Example of P and $\bar{P}$

Figure 6.5 shows an example of a situation where positions are added into P unnecessarily when using the method of Lemma 8. For the given initial position set P, the partition of $S = \{1, 2, ..., n\}$ is itself. This makes the modified position set $\bar{P} = P_n$, but there are several positions shown in (b) which are not necessary to assure property C+.

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} &
\left(\begin{array}{ccccccc}
\times & & & \times & & & \times \\
 & \times & \times & \times & & & \\
 & & \times & & & & \\
 & & & \times & & & \\
 & & & & \times & \times & \times \\
 & & & & & \times & \\
 & & & & & & \times
\end{array}\right)
\end{array}
\qquad
\begin{array}{cc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} &
\left(\begin{array}{ccccccc}
\times & \star & \star & \times & \star & \star & \times \\
 & \times & \times & \times & \otimes & \otimes & \otimes \\
 & & \times & \otimes & \otimes & \otimes & \otimes \\
 & & & \times & \otimes & \otimes & \otimes \\
 & & & & \times & \times & \times \\
 & & & & & \times & \otimes \\
 & & & & & & \times
\end{array}\right)
\end{array}
$$

P $\qquad\qquad$ $MPADD(P)$

$\times$ denotes the original position that in P.
$\otimes$ and $\star$ denote the positions that added into P.
$\otimes$ denotes the nunecessary position that added into P.

Figure 6.5 Example of positions added unnecessarily for condition C+

To avoid these situations, consider more closely the relationship between columns that are in the same subset of the partition. In the above example, unnecessary added positions result because when $(i, j) \in P$, it tells us both that columns i and j share in part the same structure, and that the part is from 1 to i, while the remainder of column j is unrelated to column i. Using the partition set of Lemma 8 to store the information needed for the modification loses the second part of this information. For example, $(5, 6)$ in P says that columns 5 and 6 have the same structure from row 1 to row 5. $(5, 7)$ in P says that column 5 and 7 have the same structure from row 1 to row 5. So columns 6 and 7 have the same structure only from row 1 to row 5. $(6, 7)$ need not be added into P for property C+. Similar reasoning holds for the positions marked by $\otimes$ in Figure 6.5 (b). $\{2,3,4\}$ and $\{5,6,7\}$ are combined into one subset because of the elements $(1,4)$ and $(1,7)$, but these elements only restrict columns 2,3,4,5,6,7 to be of the same structure in the row 1. Simply putting the sets $\{2,3,4\}$ and $\{5,6,7\}$ together will only keep the information of 'the same structure' and lose the information of 'at row 1'. This is the reason why positions (2,5), (2,6), (2,7), (3,5), (3,6), (3,7), (4,5), (4,6), (4,7) are added into $\bar{P}$, but they are not necessary for property C+ to hold.

If we use trees instead of simple partitions we can maintain more complete information to guide the modifications. Given a position set P, there is a tree that is generated from P by the algorithm **CTREE** shown below. We call such a tree a c-tree of P, and define $tree(i)$ to be the subtree rooted at i.

Algorithm [T] = CTREE[P]

for $k = n, n-1, \ldots, 1$,

form a tree with one node k

for $j = k+1, k+2, \ldots, n$

if $(k,j) \in P$ *and* $j \in tree(l), l \neq k$ **then**

connect (k,l) *so that* $tree(l)$ *is a subtree*

of $tree(k)$

endif

endfor

endfor

For the position set P of the last example, **CTREE(P)** generates a tree in $n = 7$ steps as shown in Figure 6.6. This tree gives complete information about the relationships between columns.

Lemma 9 *Let $P \subset P_n$ be a position set, and let T be the c-tree of P. Then T has the following properties:*

(1) If $j \in tree(i)$ then $j > i$.

(2) If $(i,j) \in P$ then $j \in tree(i)$.

These properties follow immediately from the definition of the algorithm, so the proof is omitted. The tree provides more information than the partition set does. For the example P in Figure 6.5, **CTREE(P)** is shown in Figure 6.6. In general, after constructing the tree the positions to be added to guarantee property C+ are determined as a set of rows over which columns must possess the same structure. This set is given by the the index associated with the nodes on the path from the root to the common ancestor of the column pair that is being considered. For example, columns 6 and 7 are brought into the same tree by node 5 so they have to have the same structure from row 1 to row 5 in order to guarantee property C+. Columns 2 and 6 are brought into the same tree by node 1 so they

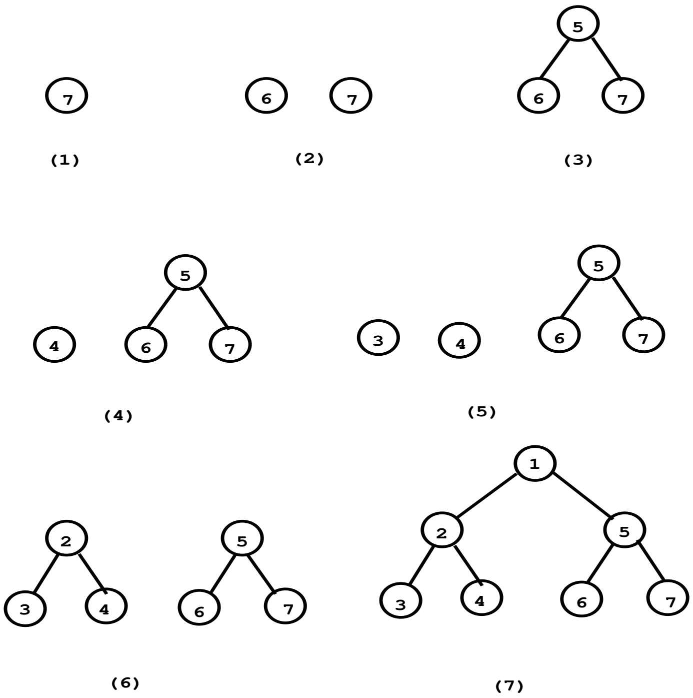


Figure 6.6 Example of TREE(P)

have to have the same structure at row 1 only. Using this information, we can make modifications that add less elements into the given original pattern P.

A position set modification algorithm **MPADD**, based on the c-tree generated by **CTREE(P)**, is now given.

Algorithm $[\bar{P}]$ = **MPADD[P, A]**

1. generate T *by* **CTREE(P)**.

2. generate $\bar{P}$ *in the following way:*

for $k = 1, 2, \ldots, n,$

  for $j = k+1, k+2, \ldots, n,$

    if $(k, j) \in P^{+}(A)$ *and* $j \in tree(k)$ **then**

      (k, j) *in* $\bar{P}$

    endif

  endfor

endfor

Theorem 19 *Let* $A \in \Re^{n\times n}$ *be a symmetric matrix,* $P \subset P_n$ *be a non-zero position set, and* $\bar{P}$ *be the position set generated by algorithm* **MPADD** *above. Then* $\bar{P} \supset P$ *and has property C+.*

Proof: First we show that $\bar{P} \supset P$. Let $(i,j) \in P$. by Lemma 9, $j \in tree(i)$. Algorithm **MPADD** assures that $(i,j) \in \bar{P}$, so $\bar{P} \supset P$.

Now we show that $\bar{P}$ has property C+. Let $(i,j) \in \bar{P}$. For any $k < i < j$, if (k,i) and (k,j) are in $P^+(A)$ and $(k,i) \in \bar{P}$, then we have $i \in tree(k)$ and $j \in tree(k)$, according to Lemma 9. Therefore $j \in tree(i)$. By the algorithm **MPADD** we have $(k,j) \in \bar{P}$. Similarly we can show that for any $k < i < j$, if (k,i) and (k,j) are in $P^+(A)$ and $(k,j) \in \bar{P}$, then $(k,i) \in \bar{P}$. Thus $\bar{P}$ has property C+. □

Algorithm **MPADD** can modify the position set so that IC is able to complete without breaking down. But experiments have shown that sometimes the resulting factor is too dense for practical use. Next, we present an algorithm based not on adding elements into P, but on eliminating some elements from P. Let $S_P(i,j) = \{k|(k,j) \in P \textit{ and } k < i\}$. $S_P(i,j)$ is the set of row indices of positions in P in column j that are smaller than i.

Algorithm [P] = MPDROP[A, P]

for $k = 1, 2, \ldots, n$,
  for $j = k+1, k+2, \ldots, n$
    if $(k,j) \in P$ *and* $S_P(k,k) \neq S_P(k,j)$ **then**
      remove (k,j) *from* P.
    endif
  endfor
endfor

Theorem 20 *Let $P \subset P_n$ be a non-zero position set, and let $\bar{P}$ be the position set generated by algorithm* **MPDROP** *above. Then $\bar{P}$ has property C+.*

Proof: From the description of the algorithm we see that for every $(i,j) \in \bar{P}$, $S_P(i,i) = S_P(i,j)$. This means that for any $k < i$, if $k \in S(i,i)$, that is $(k,i) \in \bar{P}$, we have also $k \in S_P(i,j)$, that is $(k,j) \in \bar{P}$. By the definition of property C+, we have $\bar{P}$ has property C+. □

The preceding two algorithms target arbitrary sparse symmetric positive definite systems. When additional structure is present in the matrix, the pattern modification algorithms can be specialized to take advantage of this structure. First some terms need to be defined. A symmetric matrix is called a bordered block diagonal matrix if it has the following structure.

$$\begin{pmatrix} B_1 & B_2 & B_3 & \cdots & B_k \\ B_2^T & D_2 & & & \\ B_3^T & & D_3 & & \\ \vdots & & & \ddots & \\ B_n^T & & & & D_k \end{pmatrix}$$

where B_i are called border blocks and D_i are called diagonal blocks. A matrix is said to be a nested bordered block diagonal matrix if the diagonal blocks are also bordered block diagonal matrices.

It is well known that if a position set P is the position set of a nested bordered block diagonal matrix, much of the factorization and iterative solution can be done in parallel. An important result here is that this structure will be preserved by the modification algorithms proposed above.

Theorem 21 *Let $A \in \Re^{n \times n}$ be a symmetric matrix and $P \subset P^{+}(A)$ be a position set which has nested bordered block diagonal structure. Let $\bar{P} =$ **MPADD**(P, A) and $\tilde{P} =$ **MPDROP**(P, A). Then $\bar{P}$ and $\tilde{P}$ are of the same structure as P.*

Proof: Because $\tilde{P} \subset P$, it is obvious that **MPDROP** preserves bordered block diagonal structure.

Now we will show that any position (i, j) that is outside a border block B_l or a diagonal block D_l, $1 \leq l \leq k$, will not be added into $\bar{P}$ by **MPADD**.

Let (i, j) be any position that is outside the border block B_l or the diagonal block D_l, $1 \leq l \leq k$. Because of the scan order used and connection rules used in the algorithm **CTREE**, there is no path from i to j, or from j to i. So in the algorithm **MPADD**, (i, j) will not be added into $\bar{P}$. $\Box$

Algorithm **MPBBD** is an application of **MPADD** to the pattern P, that orders the computations using nested bordered block diagonal form.

Algorithm $[\bar{P}]$ **= MPBBD[A, P]**

1. Permute P into nested bordered block diagonal form.

2. Permute A into the same form.

3. Apply **MPADD** *such that $\bar{P} =$* **MPADD**$(P,\ A)$.

Note that, this method of modification is rich in parallelism. In creating the c-tree, all the subtrees corresponding to diagonal blocks as well as all the subtrees that correspond to border blocks can be generated in parallel. In the step of generating the position set, we can first determine the positions in the border blocks and then the positions in all the diagonal blocks in parallel. We have not implemented a parallel version of this algorithm, but it is an area for future work.

The quality of a IC preconditioner is related to the ordering [21] [23],[24]. A good preconditioner must have not only potential for parallel processing but must also reduce the number of iterations so that overall computation time is reduced. How does this reordering and modification of the sparsity pattern affect the quality of the IC preconditioner? At present in the literature this question has only very specialized attempts at an answer and most discussions, including the specialized ones, rely mostly on numerical testing of model problems. In the next section, we present and discuss a set of such test results.

6.4 Experiments with pattern modification algorithms

In this section, we analyze the test results for the pattern modification methods presented above. The purpose of the testing is to examine the performance of IC with the pattern modifications proposed in the previous section. The initial pattern P to which the modifications are made is taken as the non-zero element pattern pattern of the matrix A.

First, the existence of IC after pattern modification is verified. Without pattern modification, IC factorization can only be completed for 12 of 30 problems. Table A.31 shows performance data for IC. The modification strategies are able to solve all but 4 problems: CONVE8, FFFFF800,NZFRI and STANDATA. The complete Cholesky factorization fails on these as well and, as mentioned earlier in the thesis, the problem CONVE8 comes from the constraint matrix for a Stokes problem in fluid dynamics, and has an exact rank deficiency of 1. Therefore, the failures are due to errors caused by finite precision computations and do not provide counterexamples to the earlier theorems. MPADD and MPBBD failed on all 4 of these problems while MPDROP only failed on FFFFF800. Ironically, this is because for the numerically sensitive problems the farther away we are from the complete Cholesky factorization the likelier we are to generate an incomplete factorization, e.g., in the limit we have diagonal preconditioning.

As mentioned in the previous section, MPADD modifies the pattern by adding positions into P while MPDROP does so by eliminating positions. So they can either make R too dense (MPADD) or too sparse (MPDROP). Let $nnz(R_{IC(0)})$ denote the number of elements in R using IC without pattern modification, and let $nnz(R_{MPADD})$ and $nnz(R_{MPDROP})$ be the number of non-zeros in R using IC with MPADD and MPDROP modifications. The quantities $nnz(R_{IC(0)})/nnz(R_{MPADD})$ and $nnz(R_{MPDROP})/nnz(R_{IC(0)})$ give a measure of the amount of modification. The closer they are to 1, the fewer modifications were made. These two quantities for the test problems are shown in Figure 6.7. The Figure shows that MPADD does add a large number of elements into the position set. For most of the test problems MPADD produces an R with a number of additional elements of more than 70% of the IC(0) count. Conversely, MPDROP eliminates 40% to 75% of the elements from the original given pattern. Since these added or eliminated element are selected based on the structure alone, they can degrade the quality or efficiency of IC preconditioning by severe information lose (MPDROP) or severe operation increase.

MPBBD is a MPADD with preordering. This preordering into bordered block diagonal form affects the density of modified P. Figure 6.8 shows the ratio $nnz(R_{MPBBD})/nnz(R_{MPADD})$ for the test problems. It shows that preordering reduces the density of resulting P for most of the test problems. But there are 6 exceptions, and they produce denser factors than the Cholesky factor R of the original matrix, i.e., without the BBD reordering.

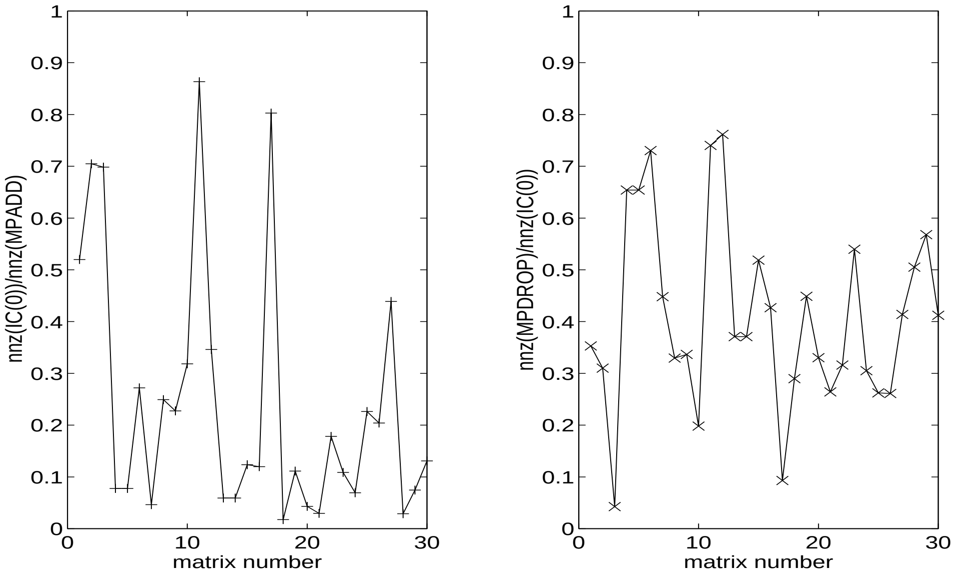


Figure 6.7 Changes in the number of non-zeros in R by the pattern modification

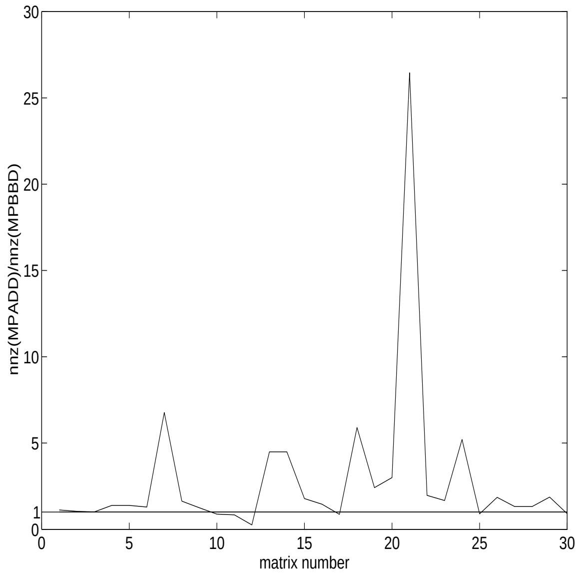


Figure 6.8 Comparison of the number of non-zeros in R with MPADD and MPBBD modifications

The effect of pattern modification on the quality of the IC preconditioner is measured by the number of operations in the iteration phase of solving the problems. The number of iterations is not a good measure, since the density of preconditioners, and hence the number of operations performed on each iteration, differ. Tables A.33 and A.34 in the Appendix show the results. The test shows that CGLS preconditioned by IC with MPADD modification converges in fewer than 10 iterations for most of the test problems. This is not unexpected because MPADD makes the density of P close to the density of Cholesky factor, so that IC is close to complete Cholesky factorization with an iterative method to improve the accuracy. CGLS preconditioned by IC with MPDROP modification works well for most of problems. There are 6 problems for which preconditioned CGLS does not converge within $2n$ iterations. Except for AMOCO1, all of those are square matrices. There are 3 problems for which CGLS preconditioned by IC with MPBBD does not converge. To compare the quality of these different patterns, we measure the ratios F(MPADD)/F(IC(0)), F(MPDROP)/F(IC(0)) and F(MPBBD)/F(IC(0)) for those problems on which IC without modification succeeds, where F(IC(0)), F(MPADD), F(MPDROP) and F(MPBBD) denote the number of floating point operations used by CGLS with IC preconditioning using no modification, MPADD, MPDROP, and MPBBD, respectively. From Figure 6.9 we see that IC without modification is the most effective preconditioner for the 60% of cases where it exists.

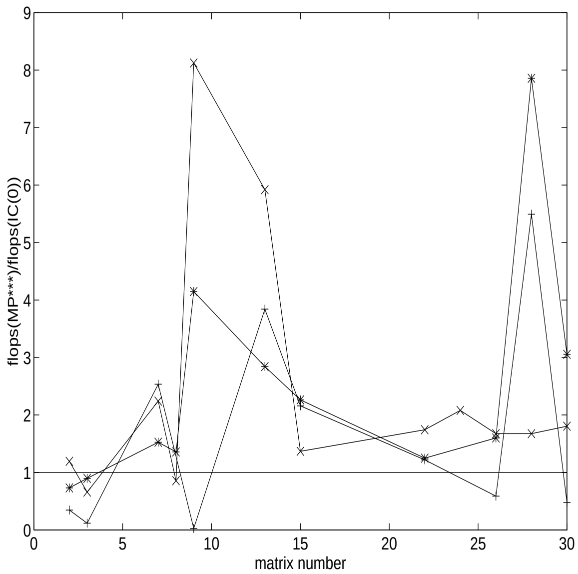


Figure 6.9 Changes in the number of non-zeros in R by the pattern modification

In order to provide a overview of IC with pattern modification methods, we set the total number of operations for those problems that fail to either compute the R or fail to converge at 10^5 million, which is much greater than the best method among the methods. Then we compare the total number of operations required by a method to the total number of operations for the best method for a given problem. If the method is the best, then this quantity is 1. If the method is much poorer than the best method, then the quantity is close to 0. Figure 6.10 shows this ratio for the four methods compared. From the Figure, IC with MPADD has the best solution most often. IC is next, while MPBBD has no best solution by comparison.

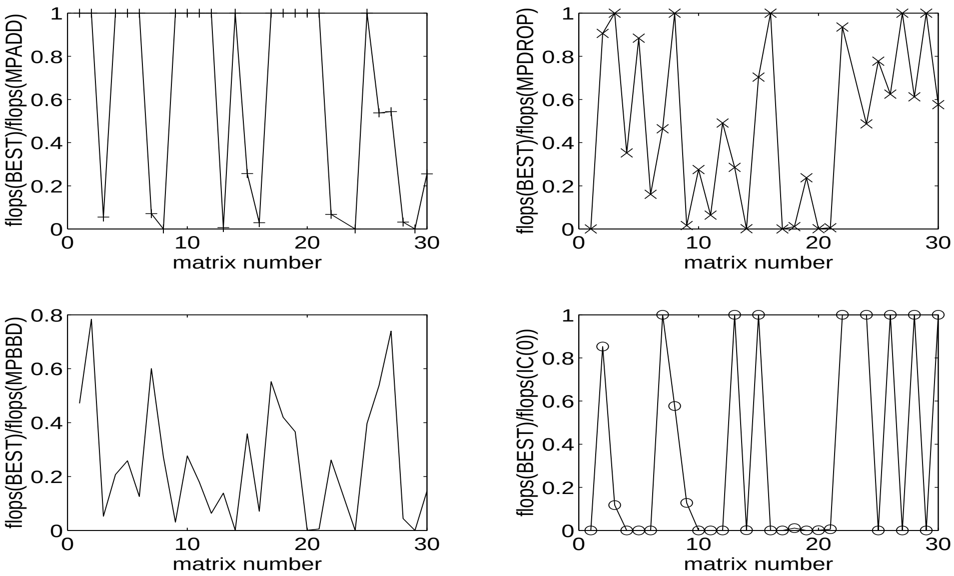


Figure 6.10 Total operations of solving the problems with different methods

In summary, experiments confirm that pattern modification greatly improves the robustness of IC preconditioning. The performance of IC with pattern modification works well when IC breaks down. Both IC with MPADD and IC with MPDROP are able to outperform IC without pattern modification for at least some problems. The drawback of these modification methods is that they only take one way of modification, either by adding or by eliminating, and so they can sometimes add too many elements or eliminate too many elements. An interesting topic for further research is to find a combination of these two basic modifications together for a better pattern modifications.

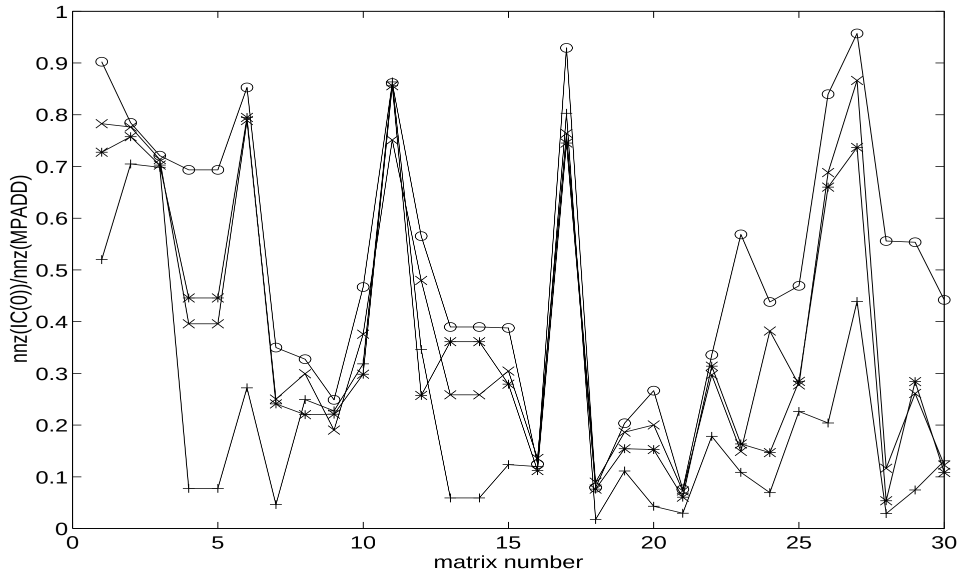
MPBBD, as shown in the Figure 6.10, tends to have performance that is not the best nor the worst on the suite of problems. The fact that it is not consistently the worst is encouraging since it has an important advantage over the other methods in its natural parallel form. This may lead to its superiority on the appropriate architectures.

6.5 Effects of matrix reordering

Experiments were conducted to study the effects of matrix ordering on the performance of pattern modifications. Of the four orderings that have been consistently considered in this thesis we apply here the three orderings motivated by reducing the fill-in and the operations for Cholesky factorization: ND, MD and RCM. MPADD and MPDROP are tested. MPBBD, since it implies a particular reordering, is not included.

Figure 6.11 shows the ratio of the number of non-zeros of R after modification of MPADD with and without ordering over the number of non-zeros in the original pattern P. We see that the ratio is reduced by the orderings. This means that the modification (adding or eliminating) on the original pattern (normal equations in this case) is reduced by the ordering. This is good provided that the given pattern is selected based on the knowledge of the applications. The reduction may not indicate that the MPADD does not add too many new elements into the P. It can still be close to the Cholesky factor R which is itself reduced. As usual, among these orderings, the MD ordering is the most effective.

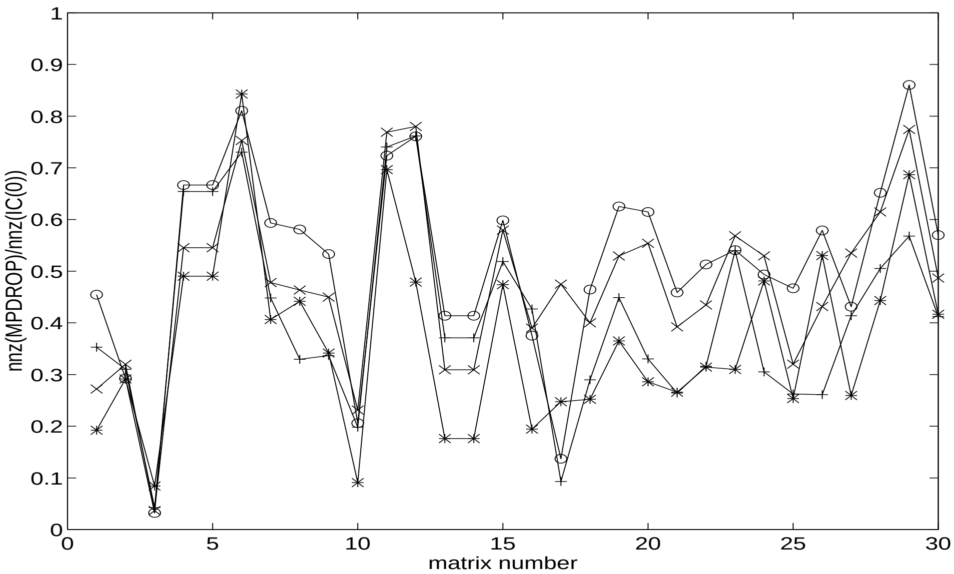
Figure 6.12 shows the ratio of the number of non-zeros of R after modification by MPDROP with and without ordering over the number of non-zeros of original pattern P. It shows that the P becomes denser after the modification of MPDROP with ordering than that after MPDROP without ordering. This indicates that MPDROP eliminates fewer elements from the original pattern after the reordering. As before, the MD ordering is the most effective.



o: minimum degree ordering $*$: reverse Cuthill McKee ordering
$\times$: nested dissection ordering $+$: without ordering

Figure 6.11 Ratio of the number of elements of P after MPADD modification over the number of elements of original P with and with out orderings

Table A.38 in the Appendix shows the number of operations for the incomplete Cholesky factorization phase. As expected, due to change in the density of R the computation count is also changed.



o: minimum degree ordering $*$: reverse Cuthill McKee ordering
$\times$: nested dissection ordering $+$: without ordering

Figure 6.12 Ratio of the number of elements of P after MPDROP modification over the number of elements of original P with and with out orderings

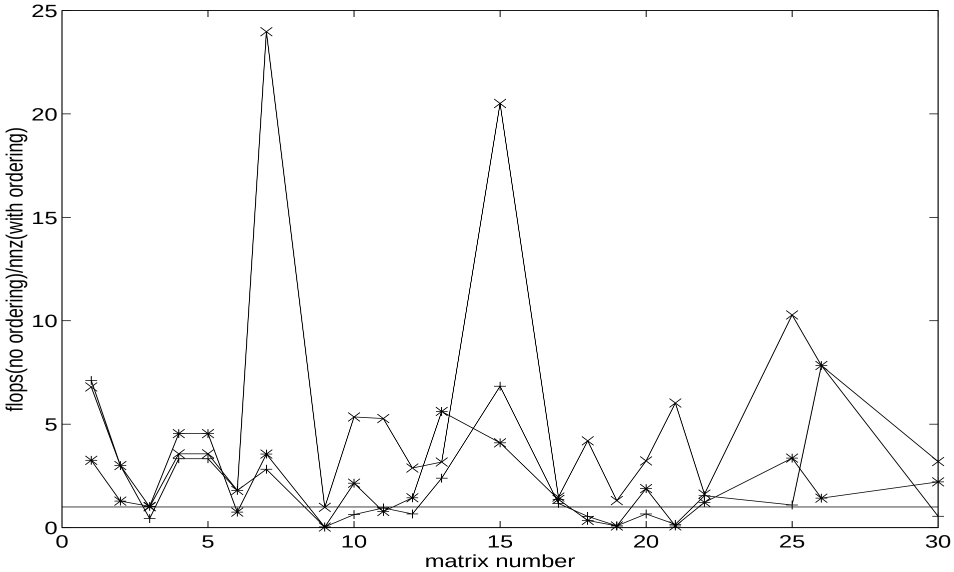
From the experiment we also see that these orderings affect the stability of IC factorization. The number of problems that failed for IC with the different combinations of modifications and orderings are listed in Table 6.1. Clearly, there is an improvement in the success rate of the IC factorization. The trend of MPDROP succeeding in the factorization but degrading in the iterative portion of CGLS due to a poor preconditioner is shown. As mentioned above the problems for which MPADD and MPBBD have trouble tend to be those which also cause numerical trouble for the complete Cholesky factorization.

	# break down				# not converge			
	none	ND	RCM	MD	none	ND	RCM	MD
IC(0)	18	17	17	16	0	0	0	0
MPADD	4	3	7	3	0	0	0	0
MPDROP	2	2	3	3	4	7	6	6

Table 6.1 The number of failures

Now consider the effects of these orderings on the quality of preconditioner. Since the numbers of non-zero elements in R produced by MPADD with these orderings are different, we measure the number of operations in the iteration phase for the comparison. Figure 6.13 shows the ratio of operations required by CGLS preconditioned by IC with MPADD without ordering to the operations required by CGLS preconditioned by IC with MPADD combined with ordering. It shows that for most combinations of

MPADD with orderings, the number of operations wasreduced. The RCM ordering is the most effective. It reduced the operation count for all the problems for which it successfully produced a preconditioner, while for the other two methods, there are examples showing that the operations in the iterative phase can be increased greatly. For example, the operations in the iterative phase for STRAT8 using IC with MPADD combined with the ND ordering and the MD ordering are increased from 1.12×10^6 (MPADD without ordering) to 32.53×10^6 and 40.21×10^6 respectively. It should be pointed out, however, that MPADD with these orderings still does better than IC without modification, in which case the number of operations needed in the iterative phase is 43.40×10^6.

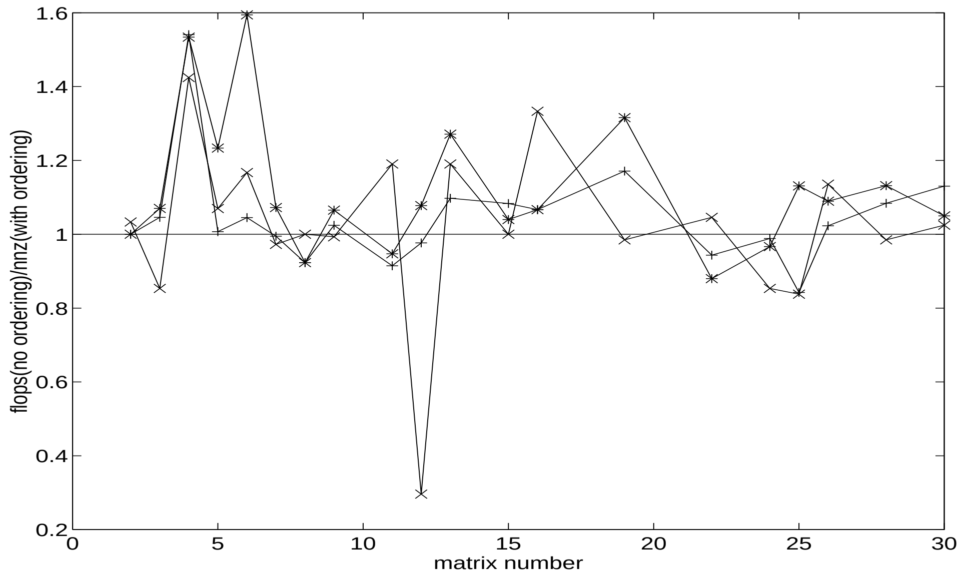


+: nested dissection ordering
×: reverse Cuthill McKee ordering
∗: minimum degree ordering

Figure 6.13 Ratio of the number of operations for the iterations after MPADD with ordering to the number of operations after MPADD without ordering

Figure 6.14 shows the ratio of the operations in iterative phase using IC with MPDROP combined with orderings to that by IC with MPDROP without ordering. We see that the operations are reduced by the orderings but by a very modest amount. For most of the test problems the ratio is within the interval (0.8, 1.6). This indicates that the preconditioning is not degraded severely by the dropping of the elements from P.

The overall effects of these orderings are examined by measuring the ratio of total operations for solving linear least squares problem using CGLS preconditioned by IC with MPADD without ordering to that for CGLS preconditioned by IC with the combination of MPADD and ordering. Figure 6.15 shows the result. Only 22 problems for which the IC with MPADD combined with all the ordering and



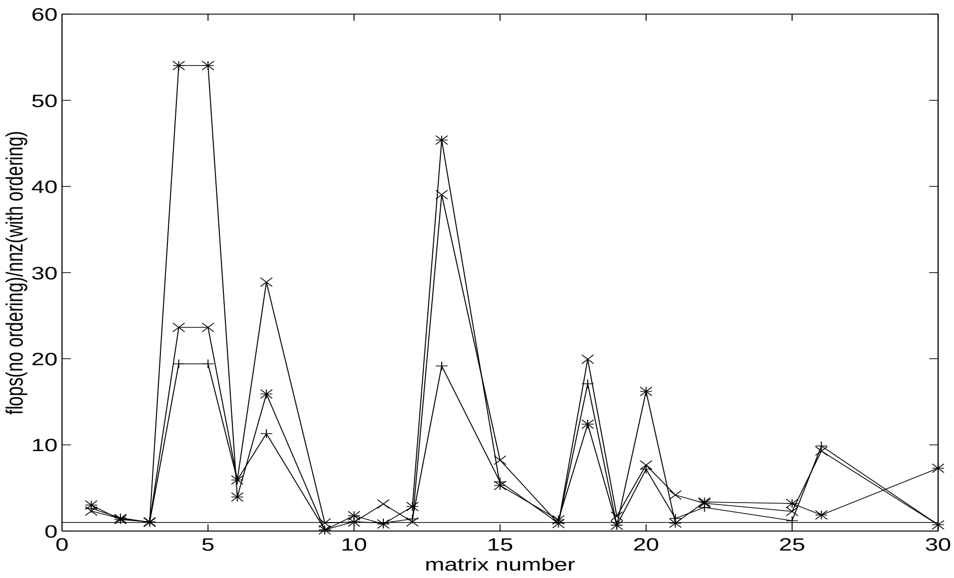
+: nested dissection ordering
×: reverse Cuthill McKee ordering
∗: minimum degree ordering

Figure 6.14 Ratio of the number of operations for the iterations after MPDROP with ordering to the number of operations after MPDROP without ordering

without ordering succeed are presented in the Figure. We see that the MD ordering is the most effective in reducing the total operations although the RCM ordering is the most effective one in the iterative phase.

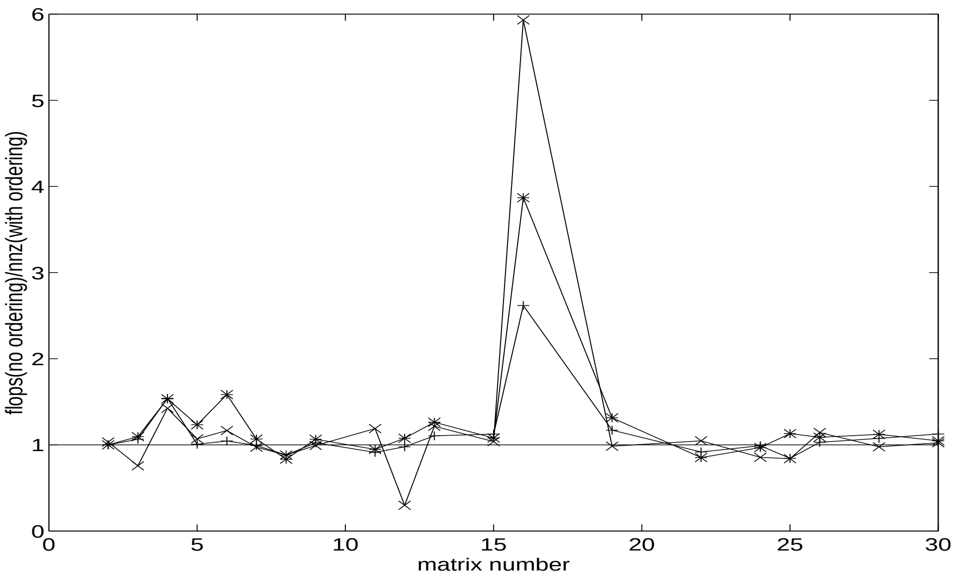
The same quantity was also measured for MPDROP. The results are presented in Figure 6.16. The effect of the ordering on MPDROP is much smaller than that observed for MPADD. The ratios are in (0.7,1.6) for most of the problems. Ordering may or may not reduce the total operations. This is not a complete surprise since the ordering methods are motivated in by reducing the fill-in which is not related to the dropping in MPDROP.

So far we have presented results on the effectiveness of the ordering of the pattern modification algorithm MPADD and MPDROP. We now consider the efficiency of the MPADD and MPDROP algorithms in computing the modifications. Figure 6.17 shows the ratio T(MPADD)/T(IC), where T(MPADD) is the time of doing MPADD modification and T(IC) is the time of IC factorization on the modified system. We see that for most of the test problems, this ratio is within 0.2. With orderings, the ratio increases slightly because the computation of IC is reduced. Figure 6.18 shows the ratio T(MPDROP)/T(IC), where T(MPDROP) is the time of doing MPDROP modification. From it we see that for most of the test problems, the ratio is greater than 1. this means that more time was spent on the pattern modification than on the IC factorization. Even though the total time of modification and factorization does not



+: nested dissection ordering
×: reverse Cuthill McKee ordering
∗: minimum degree ordering

Figure 6.15 Ratio of total operations using IC with MPADD without ordering to total operations using IC with the combination of MPADD and orderings



+: nested dissection ordering
×: reverse Cuthill McKee ordering
∗: minimum degree ordering

Figure 6.16 Ratio of the total operations using IC with MPDROP without ordering to the total operations using IC with the combination of MPDROP and ordering dominate the total time to solve the problem, due to the lower quality preconditioner produced, the algorithm MPDROP clearly needs to be improved.

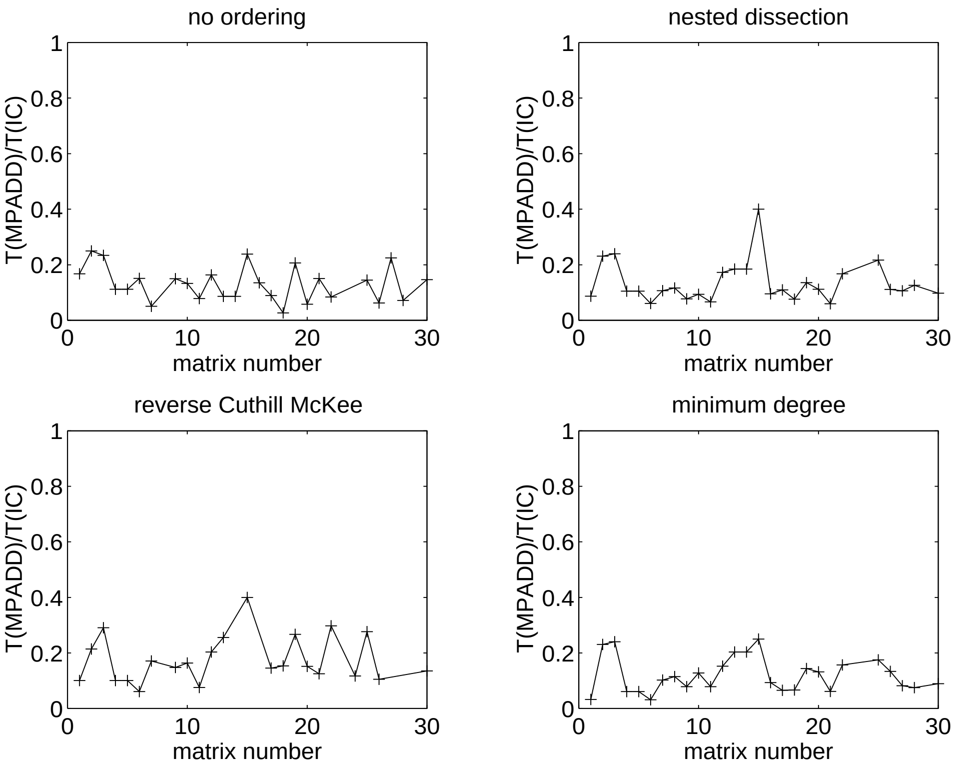


Figure 6.17 Ratio of the time for pattern modification and the time for IC factorization (MPADD)

The result of the experiments combining column orderings with pattern modifications can be summarized by saying that, in general, the orderings combined with MPADD do reduce the amount of elements added into modified sparsity pattern; do reduce the operations in iterative phase, which means they are effective in improving the quality of preconditioner; and do reduce the total operation counts which means they are effective in improving the efficiency of solving the problems relative to CGLS with IC preconditioning.

The result of the experiments also show that the orderings are much less effective for MPDROP as for MPADD. To improve the performance of MPDROP by ordering, new ordering methods must be found, which must take the motivation and method of dropping in MPDROP into consideration.

6.6 Summary

It was shown in Chapter 3 that if certain conditions are imposed on the sparsity pattern of the incomplete factor, the IC factorization of the symmetric positive definite matrix with will not suffer

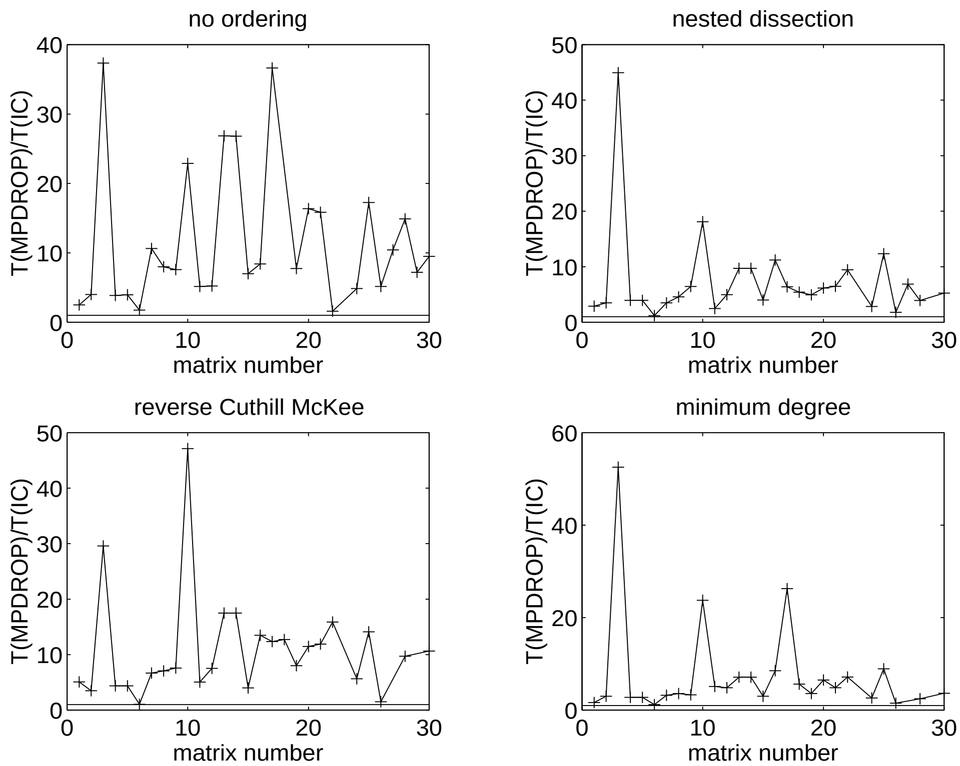


Figure 6.18 Ratio of the time for pattern modification and the time for IC factorization (MPDROP)

breakdown. In this chapter, we proved this again from a more conventional point of view. The condition gives us a new direction to improve the robustness of the IC factorization preconditioner. Three modification methods were proposed and tested on the test suite. The experiments show that with the pattern modification, we are able to greatly improve the stability of the IC factorization while preserving the necessary quality of IC preconditioner. Some problems are also identified by the experiments. The algorithms we proposed here are effective but not always efficient, especially MPDROP. A method that uses both adding and eliminating to enforce the property C+ is needed to avoid the problems of making R too dense (MPADD) or too sparse (MPDROP).

We conclude the chapter by mentioning the following open questions for the pattern modification for the incomplete Cholesky factorization preconditioning.

(1) As we mentioned above, we need more efficient algorithms or more efficient implementations of these algorithms to reduce the overhead of modifying the sparsity pattern. This work requires a better understanding of property C+ from both matrix and graph perspectives.

(2) Since it is shown in the tests that MPADD is likely to add too many elements into the non-zero set while MPDROP is likely to remove too many. It is desirable to have an algorithm that can enforce the property C+ without changing the density of R too much. Is there an algorithm

that guarantees conforming with property C+ with minimum modifications to P? Is there any heuristic or information on numerical value of elements of the matrix which could help to reduce the modification while preserve the quality of the preconditioning?

(3) Since property C+ is only a sufficient condition (will it be also necessary?) for IC factorization to be a success, can we find other ways of using property C+? For example, can we relax the condition a little so we will be able to make the pattern modification much more efficient and without losing too much stability? Can we enforce the condition on certain part of the matrix while leaving another part of the matrix unmodified, and still preserve the stability? Can we only do the modification on a small part of the original pattern when we have a stability problem with it at some step and not necessarily restart the factorization after modifying the pattern of the whole matrix?

CHAPTER 7

Comparisons of Methods

In previous chapters, the new incomplete factorization preconditioning methods ICGS, IMGS and CIMGS were proposed and tested individually. These methods are compared with each other in Chapter 4 and Chapter 5. In this chapter, we will summarize the comparisons of these preconditioners. Also, we will compare them with the standard IC factorization preconditioning and two direct methods: one based on QR factorization and the other on Cholesky factorization.

7.1 Comparison of the members of IGS family

In previous chapters, new incomplete factorization preconditioning methods: ICGS, IMGS and CIMGS with a variety of pattern selection and ordering schemes were proposed and tested individually as well as comparatively. Here we give a summary of the comparisons.

First, according to the experiments presented in Chapter 4, IMGS generates a better preconditioner with, unfortunately, much higher cost than ICGS. Therefore, the overall performance of IMGS may or may not be superior to ICGS. Both IMGS and ICGS suffer high computation costs if the sparsity of Q cannot be kept low. The effort of finding better preconditioners goes in two directions. One is to reduce the density of Q by dropping its elements as in Saad's ILQ method (essentially ICGS with semidynamic dropping on R and Q). It is effective in reducing the computation cost in the production of preconditioner but it also degrades the quality of preconditioner. It can also experience breakdown in exact arithmetic. We did not pursue this approach further in this thesis. But we believe that further study may lead to a method that can prevent the breakdown. The other direction is to eliminate Q from the computation of the preconditioner completely and also preserve the quality of the preconditioners. For ICGS preconditioners, we have not been able to find a clever way of doing this. For IMGS, we have developed a new algorithm CIMGS which can be viewed as a special implementation of IMGS without Q. The comparisons between CIMGS and IMGS and ICGS are presented in Chapter 5. The varieties of CIMGS are much more efficient than those of IMGS in the computation of the preconditioner and CIMGS preserves the quality of IMGS or, in some cases, is better than IMGS even though in exact arithmetic the two are identical. In addition, CIMGS was also found to be more efficient than ICGS. CIMGS did, however, experience more numerical breakdown than IMGS but on only a small number of ill conditioned problems. For most of those using a version of IMGS usually produced a high quality preconditioner.

Within each method of IMGS, ICGS and CIMGS, there are a variety of pattern selection strategies and ordering schemes we can choose to improve the performance of the method. In general, if the total time (or operations) to solve the problem is dominated by the computation of the preconditioner then the performance can be improved by adjusting the pattern selection method (parameters) to reduce the sparsity of the preconditioner R. By doing this, the cost of preconditioner generation will be reduced with the price that the computation in iterative phase increases. Usually the computation in preconditioning is more sensitive to the sparsity of R than is the iterative phase. So, if the quality of the computed preconditioner R is not good enough to accelerate the convergence, then we can adjust the pattern selection methods to increase the density of R. In the extreme, we can view the complete Cholesky factorization as a special case of CIMGS. Ideally, we then only need 1 iteration to get the solution, but, in practice, a few iterations of CGLS after computing the Cholesky factorization of the normal equations can improve the accuracy of the solution.

The application of column ordering schemes in combination with the preconditioners was, in general, useful in improving the performance of preconditioner generation by reducing the total operation counts. The MD ordering tended to be the overall best in almost all of the experiments.

7.2 Comparison of IMGS, ICGS and CIMGS with IC

We first define some notation. A given set P of positions in a matrix A can equivalently be defined by a 0–1 position matrix P with elements p_{ij} defined as:

$$p_{ij} = \begin{cases} 0, & (i,j) \notin P \\ 1, & (i,j) \in P \end{cases}$$

During an incomplete factorization, there are assignments such that when the component is not in the position set, the value zero will be assigned, while if the position is in position set, a computed value will be assigned. Let v and w be vectors of length n, and let p be a vector giving the non-zero positions allowed in the vector v. Then the assignment operation:

$$v_i = \begin{cases} 0, & p_i = 0 \\ w_i, & p_i \neq 0 \end{cases}$$

can be simply expressed as $v = w_{|p}$.

To establish the main theorem of Chapter 7, we first derive some results establishing certain monotonicity properties of lower triangular matrices and Cholesky factorization.

Lemma 10 . *Let $A, B \in R^{n\times n}$ be symmetric positive definite M-matrices with $A \geq B$. Then $R \geq T$, where $R^TR = A$ and $T^TT = B$ are Cholesky factorizations.*

Proof: By induction on the size of the problem.

If $A, B \in R^{1\times 1}$, then $R = \sqrt{A} \geq \sqrt{B} = T$. Now assume that when $A, B \in R^{(n-1)\times(n-1)}$, the assertion is true and let $A, B \in R^{n\times n}$. After one step of Cholesky factorization,

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} r_{11} & \\ r_{12}^T & I \end{pmatrix} \begin{pmatrix} I & \\ & A_{22} - r_{12}^T r_{12} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ & I \end{pmatrix},$$

and

$$B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} t_{11} & \\ t_{12}^T & I \end{pmatrix} \begin{pmatrix} I & \\ & B_{22} - t_{12}^T t_{12} \end{pmatrix} \begin{pmatrix} t_{11} & t_{12} \\ & I \end{pmatrix},$$

Because $A \geq B$, we then have

$$r_{11} = \sqrt{a_{11}} \geq \sqrt{b_{11}} = t_{11} > 0$$

and

$$0 \geq r_{12} = \frac{a_{12}}{r_{11}} \geq \frac{b_{12}}{r_{11}} \geq \frac{b_{12}}{t_{11}} = t_{12}.$$

Factor the reduced matrices $A_{22} - r_{12}^T r_{12}$ and $B_{22} - t_{12}^T t_{12}$, which are also symmetric positive definite M-matrices, to get

$$R_{22}^T R_{22} = A_{22} - r_{12}^T r_{12} \quad \textit{and} \quad T_{22}^T T_{22} = B_{22} - t_{12}^T t_{12}.$$

Then

$$R = \begin{pmatrix} r_{11} & r_{12} \\ & R_{22} \end{pmatrix} \quad \textit{and} \quad T = \begin{pmatrix} t_{11} & t_{12} \\ & T_{22} \end{pmatrix}$$

Since

$$A_{22} - r_{12}^T r_{12} \geq B_{22} - r_{12}^T r_{12} \geq B_{22} - t_{12}^T t_{12},$$

by the induction assumption, we have $R_{22} \geq T_{22}$, completing the proof. □.

Lemma 11 *Let $A, B \in R^{n\times n}$ be symmetric positive definite M-matrices, with $A \geq B$. Let P be a non-zero position matrix for incomplete Cholesky factorization. Then $R \geq T$, where R and T are the incomplete Cholesky factors of A and B, respectively, using the same P.*

Proof: By induction on the order of the matrices. Again the result holds trivially for $n = 1$. Assume that it is true for matrices of order $n - 1$. Let

$$R = \begin{pmatrix} r_{11} & r_{12} \\ & R_{22} \end{pmatrix},$$

and

$$T = \begin{pmatrix} t_{11} & t_{12} \\ & T_{22} \end{pmatrix}$$

be a partitioning of R and T such that r_{11} and t_{11} are scalars. Let A, B and P be conformally partitioned, so that A_{22}, B_{22}, P_{22}, R_{22} and T_{22} are all of order $n-1$. After one step of incomplete Cholesky factorization on A and B, we have

$$r_{11} = \sqrt{a_{11}} \geq \sqrt{b_{11}} = t_{11} > 0$$

and

$$\begin{aligned} 0 \geq r_{12} &= (a_{12}/r_{11})_{|p_{12}} \\ &\geq (b_{12}/r_{11})_{|p_{12}} \\ &\geq (b_{12}/t_{11})_{|p_{12}} \\ &= t_{12} \end{aligned}$$

The rest of R and T are computed by applying incomplete Cholesky factorization to $A^{(2)} = A_{22} - r_{12}^T r_{12}$ and $B^{(2)} = B_{22} - t_{12}^T t_{12}$, respectively. Since $A^{(2)}$ and $B^{(2)}$ are both M-matrices,

$$\begin{aligned} A^{(2)} &= A_{22} - r_{12}^T r_{12} \\ &\geq B_{22} - r_{12}^T r_{12} \\ &\geq B_{22} - t_{12}^T t_{12} \\ &= B^{(2)} \end{aligned}$$

By the induction hypothesis, $R_{22} \geq T_{22}$. Together, we have $R \geq T$. $\square$

The next Theorem establishes a partial ordering on the incomplete factors that result when using IMGS, complete Cholesky and incomplete Cholesky. This in turn provides a partial ordering on the corresponding error matrices.

Theorem 22 *Assume that $A \in R^{m \times n}$ is such that $B = A^T A$ is a symmetric positive definite M-matrix, where $B \in R^{n \times n}$. Let R_{CHOL} and R_{IC} be the upper triangular matrices from Cholesky factorization of B and incomplete Cholesky factorization of B with the pattern P, respectively. Let R_{IMGS} be the upper triangular matrix from IMGS on A with the same pattern P. The following relation is satisfied:*

$$R_{CHOL} \leq R_{IMGS} \leq R_{IC},$$

Furthermore, $E_1 \leq E_2$, where

$$\begin{aligned} E_1 &= R_{IMGS} - R_{CHOL}, \\ E_2 &= R_{IC} - R_{CHOL}. \end{aligned}$$

Proof: Since IMGS on A generates the same upper triangular matrix R as CIMGS on $A^T A$, in the proof we use the CIMGS algorithm. The proof is carried out in two parts.

(1) The proof of $R_{CHOL} \leq R_{IMGS}$. By induction on the size of the problem. (1) is trivial for $n = 1$. Assume that (1) is true for $n-1$. Let $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{12}^T & B_{22} \end{pmatrix}$ and $P = \begin{pmatrix} p_{11} & p_{12} \\ p_{12}^T & P_{22} \end{pmatrix}$ be the partition. Let $R_{CHOL} = \begin{pmatrix} r_{11} & r_{12} \\ & R_{22} \end{pmatrix}$, $R_{IMGS} = \begin{pmatrix} t_{11} & t_{12} \\ & T_{22} \end{pmatrix}$ denote the factors of Cholesky factorization and CIMGS factorization of $A^TA =$, respectively, where B_{22}, P_{22}, R_{22} and T_{22} are in $R^{(n-1)\times(n-1)}$.

Applying one step of Cholesky factorization and CIMGS factorization to A^TA, we get

$$r_{11} = \sqrt{b_{11}}, \quad r_{12} = \frac{b_{12}}{r_{11}}$$

and

$$t_{11} = \sqrt{b_{11}}, \quad t_{12} = \left(\frac{b_{12}}{t_{11}}\right)_{|p_{12}}.$$

It is easy to see that $r_{11} = t_{11}$ and $r_{12} \leq t_{12}$. Let g_{12} denote the dropped part from the first step of CIMGS, so that $r_{12} = t_{12} + g_{12}$ and $t_{12}^T g_{12} = 0$.

R_{22} is the Cholesky factor of $B^{(1)} = B_{22} - r_{12}^T r_{12}$, and T_{22} is the CIMGS factor of $\tilde{B}^{(1)} = B_{22} - t_{12}^T g_{12} - g_{12}^T t_{12} - t_{12}^T t_{12}$. Again both $B^{(1)}$ and $\tilde{B}^{(1)}$ are M-matrices (see the discussion in Chapter 3) . It is not difficult to see that $B^{(1)} \leq \tilde{B}^{(1)}$. Because

$$\begin{aligned} B^{(1)} &= B_{22} - r_{12}^T r_{12} \\ &= B_{22} - t_{12}^T t_{12} - t_{12}^T g_{12} - g_{12}^T t_{12} - g_{12}^T g_{12} \\ &= \tilde{B}^{(1)} - g_{12}^T g_{12} \end{aligned}$$

and $g_{12}^T g_{12} \geq 0$. Let $L^T L = \tilde{B}^{(1)}$ be the Cholesky factorization of $\tilde{B}^{(1)}$. By Lemma 10, $R_{22} \leq L$, and by the induction assumption, we have $L \leq T_{22}$. So $R_{22} \leq T_{22}$, which together imply $R_{CHOL} \leq R_{IMGS}$. By the induction, this is true for any n.

(2) The proof of $R_{IMGS} \leq R_{IC}$. This is similar to the induction proof of (1). It holds trivially for $n = 1$. Assume that it is true for $n-1$. Then for matrices of order n, let $R_{IC} = \begin{pmatrix} l_{11} & l_{12} \\ & L_{22} \end{pmatrix}$ be the incomplete Cholesky factor of A^TA. Applying one step of incomplete Cholesky factorization to A^TA, we have

$$l_{11} = \sqrt{b_{11}}, \quad and \quad l_{12} = \left(\frac{b_{12}}{l_{11}}\right)_{|p_{12}},$$

so $l_{11} = t_{11}$ and $l_{12} = t_{12}$. Now L_{22} is the incomplete Cholesky factor of $\hat{B}^{(1)} = B_{22} - l_{12}^T l_{12} \geq \tilde{B}^{(1)}$, and $\hat{B}^{(1)}$ is also a M-matrix. Let $\hat{L}$ be the incomplete Cholesky factor of $\hat{B}^{(1)}$. By the induction hypothesis, we have $T_{22} \leq \hat{L}$. From Lemma 11, $\hat{L} \leq L_{22}$. Thus $T_{22} \leq L_{22}$. Together, this implies $T \leq L$. By induction, it is true for matrices of any order $n \geq 1$.

Putting (1) and (2) together, the theorem is proved. □

This result indicates that R_{IMGS} better approximates R_{CHOL} than R_{IC} does, so we expect that R_{IMGS} will performs better than R_{IC} as preconditioners. What if the normal equations are not an M-matrix? Will R_{IMGS} performs better than R_{IC} for these more general cases? Since we do not yet have analytic results to answer this question, we next seek to experimentally determine which is better by numerical testing.

In previous chapters we tested the methods of IMGS, ICGS and CIMGS and IC, respectively. In general, IC is much more likely to fail than IMGS, ICGS and CIMGS. This is the primary advantage of incomplete orthogonal factorization preconditioning methods over IC preconditioning. As is noted in Chapter 5, IC failed so often that it is not practical to use it without modification for problem solving. For problems for which IC exists without modification, we now compare the performance of IC with IMGS, CIMGS (which is equivalent to IMGS, but more efficient) and ICGS.

When the pattern is chosen to be that of the normal equations, the number of iterations of CGLS preconditioned by CIMGS, IMGS, ICGS and IC(0) respectively are shown in Table 7.1.

name	cimgs	imgs	icgs	IC
belladit	14	14	15	15
bellmedt	16	16	16	17
dunes8	20	20	20	21
conev8	1	1	1	1
strat8	89	89	89	91
fs_760_1	4	4	4	4
gre_115	34	34	34	35
steam2	7	7	7	7
nzfri	57	57	57	57
scsd6	21	21	21	21
shell	23	23	21	22
woodw	34	34	34	35

Table 7.1 Comparison of CIMGS, IMGS, ICGS and IC in the performance of iteration reduction.

This table shows that for more general problems, with the same pattern the factors R generated by these methods perform similarly as preconditioners. IC(0) is likely to take one or two more iterations than the others. Although the difference is small, it seems to agree with the theoretical analysis for the case when A^TA is an M-matrix. This does not provide a firm basis to conjecture that the theoretical results can be extended to more general problems, because the testing is limited. We also do not yet know if this comparison will change with as we vary the non-zero pattern or how it changes as different aspects of the M-matrices change, e.g., the angle between the columns. More testing is needed to answer this question.

For problems for which IC needs the pattern modification to avoid the breakdown, the preconditioner R produced by IC using a modified pattern is the same as the R produced by IMGS (or CIMGS) using the same pattern ideally. The performance of R produced by IC as a preconditioner will be the same (or very close) as that of the R by IMGS (or CIMGS).

But keep in mind that the differences in the cost of computing these factorizations are quite significant. IC(0) is the cheapest one, while IMGS is the most expensive one. So in general, the IGS family does provide more robustness and does not degrade in convergence compared to IC when the IC factorization exists. The cost of the factorization and therefore the cost of solving the system, however, in those cases where both CIMGS and IC successfully use the same static pattern, not unexpectedly, heavily favors IC. CIMGS does have a significant advantage over IC and other preconditioners – CIMGS with dynamic or semidynamic dropping is the only robust preconditioner based on numerical dropping that does not have to rely on elaborate restart mechanisms to achieve robustness when the factorization fails to exist for a given drop tolerance [65]. This enables CIMGS to use its adjustability to improve its solution's efficiency by using a smaller and more problem dependent non-zero set to generate its preconditioner to offset the added cost of its factorization compared to IC and other similar methods.

This can be seen from the data in Figure 7.1. For the 12 problems where IC exists and converged the pattern selection was adjusted for CIMGS[1]. The CIMGS version is either comparable or appreciably better than the IC version indicating with careful adjustment the CIMGS preconditioner can be competitive with or superior to the standard and efficient IC preconditioner.

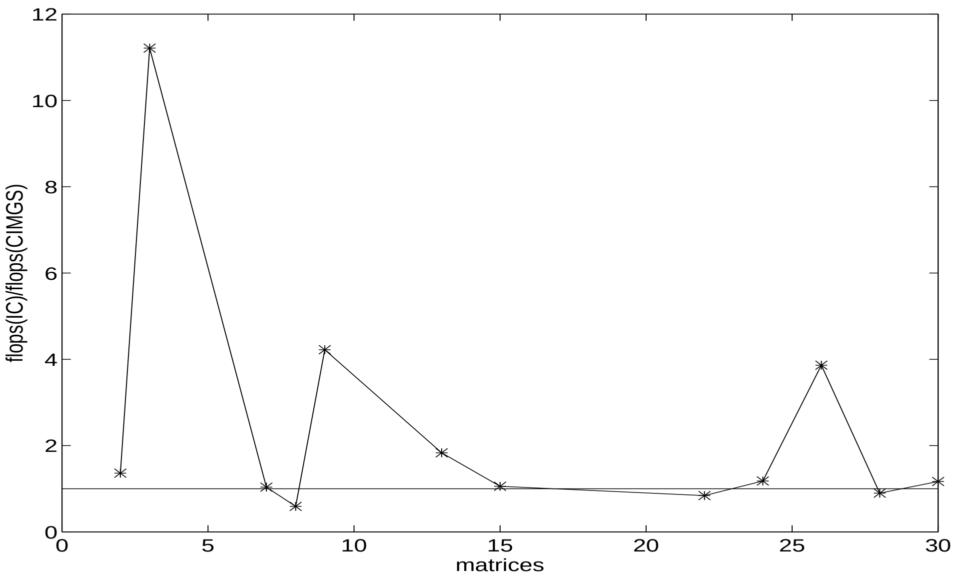


Figure 7.1 Ratio of total operations for CGLS/IC to total operations for CGLS/CIMGS with adjusted pattern selection

[1]A description of the adjustments is given in the discussion of the comparison to the normal equations.

7.3 Comparison with the direct methods

In this section, we discuss the comparison of IGS preconditioned iterative methods with the direct methods. More specifically, we compare the iterative methods proposed in the study with the two direct methods: the normal equations method and the QR factorization method. For the normal equations method, we first form the normal equations and then apply the algorithm implemented in the SPARSPAK-A package to solve the symmetric positive definite system. For the QR method, we use SPARSPAK-B which performs orthogonal reduction of A using Givens rotations. The results of the performance test are listed in Table A.44 for SPARSPAK-A and A.45 for SPARSPAK-B in Appendix.

To compare an iterative method with a direct method, it is necessary to specify the accuracy required for the iterative one. Here our comparison is based on the accuracy of $\| Ax - b \|_2 \leq 10^{-6}$.

From the Tables A.45 and 4.2 we see that SPARSPAK-B is able to solve all the problems except the two problems, MC_FE and STEAM2, for which $\| Ax - b \|_2 \leq 10^{-6}$ could not be reached because $\| b \|_2$ is very large, 3.955×10^{13} for MC_FE and 5.266×10^{10} for STEAM2. The relative residuals for the solution of these two problems are small.

From the tests we can see that the QR method is more robust than the preconditioned iterative methods. There are problems for which IMGS, ICGS, and CIMGS break down in the factorization with certain patterns on certain problems, and need adjustment of the pattern selection method. Comparing the number of floating point operations and the time used by the methods, we can see that QR using Givens rotations is more efficient than Gram-Schmidt type factorization in terms of the number of operations performed. We can see that even IMGS and ICGS, which are incomplete Gram-Schmidt type factorizations, may need more operations to produce the factor R. When using CIMGS, however, the CGLS preconditioned by CIMGS is more efficient than the direct method implemented in SPARSPAK-B.

The normal equations method implemented using SPARSPAK-A is much more efficient than the QR in SPARSPAK-B and is less robust. There are problems for which SPARSPAK-A fails to generate the Cholesky factor. The number of problems that SPARSPAK-A with different orderings fails to compute the factorization are listed in Table 7.2. In addition, the number of problems that the SPARSPAK-A with the correspondent orderings provide the most efficient solutions are also listed in the table.

method	reverse Cuthill-McKee	refine quotient tree	minimum degree
# failure	6	5	6
# best	9	1	15

Table 7.2 Number of problems that SPARSPAK-A fails and succeed most efficiently in factorization among the different ordering

Compared with CGLS preconditioned by CIMGS, which is the most efficient preconditioner proposed in this study, the normal equations method is less robust in the sense that it fails on more problems than the CGLS/CIMGS combination and there is almost no space for improvement. For CGLS preconditioned by CIMGS, there is great flexibility in pattern and ordering selection so that once CIMGS with a pattern fails to compute the factor R, we can either change the pattern or change to IMGS which is more robust than CIMGS according the error analysis and experiments presented in the previous chapters.

With a more careful selection of the sparsity pattern, the CGLS preconditioned by CIMGS can be as efficient as the normal equations method. For example, we compare the following two set of performance tests. The first set of data is the measure of time and operations using SPARSPAK-A with the RCM ordering to solve the normal equations for the test suite. The second set of data is the same measure but applying CGLS with CIMGS preconditioning. In this set, 50% of the problems use either the static pattern of the normal equations or a dynamic pattern with $\epsilon = 0.02$, whichever is better, and the remaining 50% of the problems use dynamic pattern with no more than 3 refinements of ϵ. The comparisons of these two sets of performance data are shown in Figures 7.3 and 7.2.

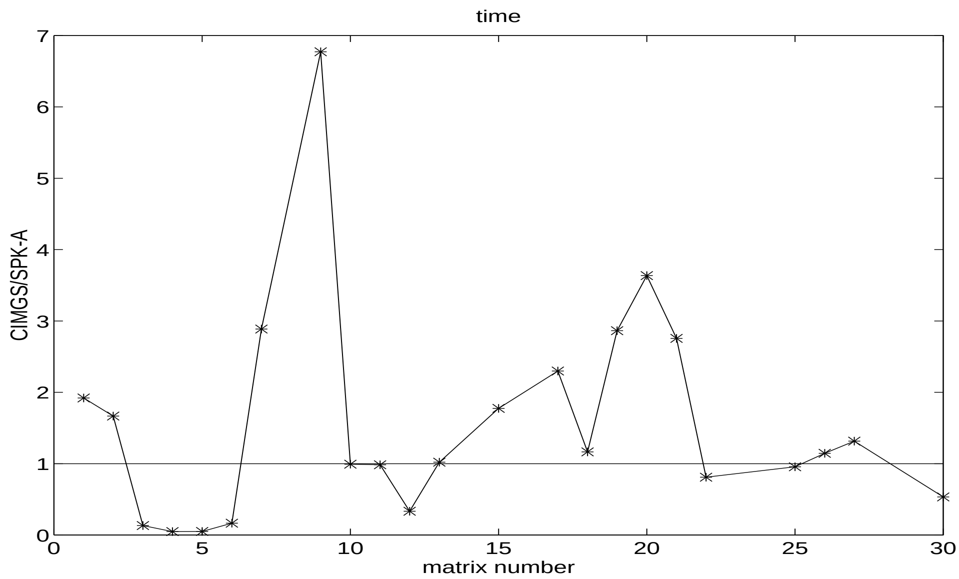


Figure 7.2 Comparison of normal equation method with CGLS preconditioned by CIMGS in terms of time

From Figure 7.3 we see that for some problems, such as BELLMEDT, ILLC1850, WELL1850 and WELL1033, the iterative method works much more efficiently than the normal equations method. For some problems, for example, STRAT8 and PROSE_2, the normal equations method works better.

Figure 7.2 shows the comparison on time. The result is different from the comparison of the number of operations. In terms of time, there are more problems for which the normal equations method is better than CGLS preconditioned by CIMGS. For the most part, this inconsistency is due to the difference in

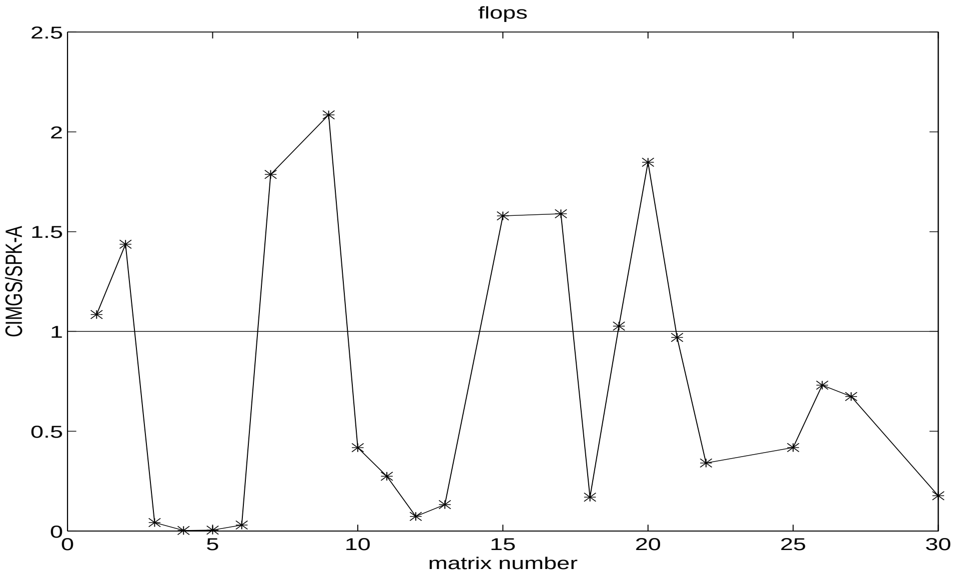


Figure 7.3 Comparison of normal equation method with CGLS preconditioned by CIMGS in term of operations

optimization levels afforded the two codes. The implementation of CIMGS is not optimized compared to SPARSPAK, which is a well optimized code. There are some parts of the implementation of CIMGS that still need to be optimized. For example, the symbolic analysis and data structure allocation could be improved by properly adapting the fast symbolic Cholesky factorization result. This will be the future work.

In summary, compared with the QR method, a robust direct method of solving linear least squares problems, we conclude that CGLS preconditioned by the methods proposed in the study can be as robust as QR method. The Modified Gram-Schmidt orthogonal factorization method is a limit case of this family of incomplete orthogonal factorization methods. For most test problems, using CGLS with CIMGS, is much more efficient than the QR method. On the other hand, the normal equations method, on the problems it can solve, is very efficient. If we are free to use the adjustability of the CIMGS preconditioner we find that CGLS with CIMGS preconditioning can be as efficient as the normal equations method.

A direct method which we did not compare but which, based on the last set of results discussed, merits study, is the augmented system method and possible iterative versions perhaps involving CIMGS preconditioning.

CHAPTER 8

Conclusion

In this chapter, we summarize the preceding chapters and offer suggestions on several subjects for future work.

8.1 Summary

The main contributions of this study can be summarized as follows:

(1) A generalization of two incomplete orthogonal factorization preconditioning methods (ICGS and IMGS) and their main variants.

(2) Theory assuring the existence of IMGS and ICGS.

(3) Conditions for the existence of IC based only on the sparsity pattern of the preconditioning matrix R.

(4) A new efficient preconditioning algorithm based on incomplete orthogonal factorization, CIMGS.

(5) Error analysis of IMGS and CIMGS.

(6) Establishment of several properties of IMGS and CIMGS when $A^T A$ is an M-matrix:

- Regular splitting: $R^{-T} A^T A R^{-1} = I - E$ is a regular splitting.
- Monotone property: if P_1 and P_2 are zero position sets such that $P_1 \subset P_2$, and R_i is the factor of IMGS using P_i, then $R_1 \leq R_2$.
- Relations between R_{CHOL}, R_{IMGS} and R_{IC}: $R_{CHOL} \leq R_{IMGS} \leq R_{IC}$ elementwise.

(8) Three new algorithms for pattern modification that ensure the existence of IC: MPADD, MPDROP and MPBBD.

8.2 Future work

The study in this thesis not only shows that the incomplete orthogonal factorization methods are a family of preconditioners that are robust and efficient, but also promises potential for future research. The interesting subjects that are closely related to the study presented in this thesis are listed in the following, some of them are being studied and some of them will be started in the near future:

(1) As pointed out in Chapter 4, the current implementation of IMGS does not aggressively exploit the sparsity of R in the QR factorization. Great benefit could be gained by careful use of a symbolic analysis of the QR or Cholesky factorizations, or a special purpose symbolic factorization algorithm which takes into account the denials of CIMGS. However, as pointed out in Chapter 5, the latter topic is very complex and it is not clear that a *fast* symbolic algorithm exists.

(2) As mentioned in chapter 4, dropping on Q can be an effective way to reduce the computation cost of the IMGS and ICGS factorizations. But, simply dropping some small elements can result in breakdown in both exact and inexact arithmetic. Hence, it would be very interesting to find modification techniques to act as a remedy for, or to prevent a priori, this drawback. It would be equally interesting to see the discussions of 'positional conditions' to enforce the existence of the IMGS and the ICGS factorizations with respect to dropping on Q like the property C and property C+ for the IC factorization. This may also enable the placement of IC into the IGS family (recall that complete Cholesky factorization is a trivial limit case). There is some indication that IC may be an ILQ-like member of the IGS family, i.e., to derive it you must consider dropping of some sort on R and Q.

(3) It should be pointed out that, while the IGS family is more robust than IC and the direct methods discussed earlier, the best performance of any member, but for CIMGS in particular, depends on the selection of an optimal sparsity pattern and careful choice of other parameters. The fact that in exact arithmetic the methods are breakdown-free and in inexact arithmetic much less prone to numerical breakdown than the other methods means that the choice of various patterns and orderings can be used to concentrate on improving performance. As was shown in Chapter 7 CIMGS can benefit substantially from this flexibility. It is precisely this combination of flexibility and freedom that makes the CIMGS a method with great potential. But, it is also this flexibility and freedom that, without clear guidelines for the parameter and pattern selection, will cause the difficulty to employ the method with the best performance or even satisfactory performance due to a wealth of apparently equally valuable options. Further research is needed to provide better understanding of these methods and the heuristics for the selection of the best sparsity pattern and method parameters.

(4) Parallel processing is a very important issue which has not been discussed in this study. It will certainly be one of the future research topics. All the techniques that associate with the sparse matrix computations will be studied to see if they can be applied to the implementation of the proposed preconditioning methods. Besides the parallel implementation of the algorithms proposed in this thesis, it will be interesting to see new algorithms combining the proposed methods with the proper sparsity pattern selection strategies and the matrix ordering techniques together for the better performance of the parallel processing.

(5) New ordering methods specifically designed for this class of the incomplete orthogonal factorization preconditioners are worthy of consideration. Four standard ordering methods were investigated in this study. The results show that these orderings are able to help improve the algorithms to a certain degree although they are originally aimed at improving the performance of the Cholesky factorization. For the incomplete orthogonal factorization algorithms, we may exploit their particular structure and design the orderings for them. Some possibilities are:

- IMGS and ICGS: The column weight ordering and the reverse Cuthill McKee ordering should take the fill-in in Q into consideration. As mentioned in Chapter 4, for the algorithm of minimum degree ordering and nested dissection ordering we do not use any information of the structure of A. It is expected that a method taking both the structure of A and A^TA into consideration and aimed to reduce the fill-ins in both A^TA and Q part will do better in improving the performance of the IMGS and the ICGS.
- CIMGS: It is noted that there is a difference in the mode of fill-in between the CIMGS and the Cholesky algorithms. A new algorithm with a structure similar to the minimum degree ordering but different at the fill-graph updating stage may be worth studying.

(6) IC with pattern modification: The key topic here is the development of pattern modification strategies that enforce property C+ via the minimum modification. This probably requires careful study of the interaction of the modification of P and the matrix ordering used.

(7) Solving nonsymmetric linear systems is a very important and interesting subject for research. Saad applied his ILQ preconditioner to the problem but found the cost too high. CIMGS may change this situation.

(8) All the preconditioning methods proposed here are applied to the CG type iterative methods. How will they perform when combined with other type of iterative methods, for example, row projection methods, GMRES or Lanczos-based methods? Of particular interest is adapting the preconditioner to the particular method. We are presently investigating the relationship of near-orthogonality of

the coefficient matrix of a system of linear equations to the convergence behavior of a collection of iterative methods and the implications for the use of CIMGS as a preconditioner.

APPENDIX A

Numerical testing results

The results of selected experiments are presented in tabular form in this Appendix. For time and operation count measurements the units of seconds and millions should be assumed unless otherwise noted. If a problem does not converge within the given iteration limit l, then the iteration count is recorded as $> l$ and the corresponding measures of the time and operation count reflect the activity of l iterations. In all tables, we use T($method$) to denote the time for the computation of $method$ and F($method$) to denote the number of operations. The field $method$ can be one of ICGS, IMGS, CIMGS, IC, CGLS or TOTAL. The method CGLS denotes in tables only the iteration phase of the preconditioned CGLS method, while TOTAL denotes the total cost (time or operation count). We use $iter$ as the number of iterations, and nnz(R) as the number of non-zero elements of R.

Table A.1 shows the results of ICGS using the dynamic pattern selection strategy. The drop tolerance parameter is set to $\epsilon = 0.02$.

name	nnz(R)	iter	T(ICGS)	T(CGLS)	F(ICGS)	F(CGLS)	F(TOTAL)
amoco1	37387	>660	76.26	82.63	110.96	196.36	307.32
belladit	2247	8	0.69	0.06	1.15	0.14	1.29
bellmedt	69861	19	454.66	4.66	533.46	10.02	543.48
illc1850	7670	401	12.58	16.19	13.13	30.49	43.62
well1850	7392	58	12.56	2.35	13.40	4.37	17.77
well1033	3080	49	3.12	0.96	3.92	1.81	5.73
dunes8	6900	16	21.37	1.17	27.86	2.53	30.39
conev8	3419	1	8.34	0.07	12.25	0.13	12.39
strat8	11187	100	274.90	43.01	187.10	43.51	230.61
bnl1	22616	>1264	25.29	72.22	37.92	171.75	209.67
ship12s	3817	29	2.82	0.99	0.63	1.88	2.51
gen	3816	36	1.22	0.75	0.26	1.36	1.63
fs_760_1	852	7	0.72	0.14	0.10	0.25	0.35
fs_760_3	8971	>1520	8.05	48.80	9.04	100.06	109.10
gre_115	1339	14	0.13	0.06	0.20	0.11	0.31
hwatt_2	21568	1	27.10	0.13	38.85	0.22	39.06
mc_fe	9917	895	9.46	57.50	10.08	128.27	138.35
orsreg_1	5729	2851	5.56	160.37	0.40	276.78	277.19
pde_9511	8812	26	5.93	0.84	8.39	1.63	10.03
pores_2	9572	>2448	3.60	106.68	3.27	211.80	215.08
saylr4	12295	2864	21.04	279.77	10.16	478.19	488.34
steam2	1178	28	0.59	0.89	0.26	1.84	2.10
fffff800	40916	> 1050	25.82	82.77	43.34	203.80	247.14
nzfri	17028	459	80.99	30.66	113.71	68.13	181.85
pilot4	3666	33	1.95	0.66	2.56	1.37	3.93
scsd6	2379	11	2.13	0.19	3.41	0.43	3.84
seba	34367	>1032	29.93	68.17	48.25	168.41	216.65
shell	3726	17	5.68	0.39	6.45	0.67	7.11
standata	3431	220	3.43	3.37	4.74	6.97	11.71
woodw	9912	34	30.60	3.74	30.09	7.83	37.92

Table A.1 ICGS with dynamic pattern selection strategy

Table A.2 shows the results of IMGS using the dynamic pattern selection strategy. The tolerance parameter is set to $\epsilon = 0.02$.

name	nnz(R)	iter	T(IMGS)	T(CGLS)	F(IMGS)	F(CGLS)	F(TOTAL)
amoco1	19365	173	90.45	16.92	194.41	39.07	233.48
belladit	2226	8	1.17	0.06	2.66	0.14	2.79
bellmedt	64186	18	2109.07	4.21	4251.12	9.09	4260.20
illc1850	6654	87	51.32	3.38	120.38	6.29	126.67
well1850	6971	37	65.25	1.47	149.75	2.74	152.49
well1033	2983	38	7.60	0.74	16.87	1.39	18.26
dunes8	7416	13	330.22	0.96	682.05	2.09	684.14
conev8	3656	1	121.44	0.07	264.47	0.13	264.61
bnl1	15325	103	55.92	4.74	129.18	11.04	140.22
ship12s	3885	26	7.91	0.88	11.19	1.69	12.89
gen	3817	32	4.36	0.66	6.61	1.21	7.83
fs_760_1	852	7	3.34	0.14	5.54	0.25	5.79
fs_760_3	8036	> 1520	71.30	45.79	174.80	94.38	269.18
gre_115	1363	13	0.38	0.05	0.92	0.11	1.03
hwatt_2	21221	1	965.04	0.13	2060.69	0.21	2060.90
mc_fe	11586	717	58.80	47.13	148.59	107.57	256.16
orsreg_1	5671	507	42.39	28.08	84.12	49.14	133.26
pde_9511	9644	17	119.86	0.58	295.14	1.13	296.28
pores_2	5565	> 2448	29.80	89.53	71.35	172.56	243.91
saylr4	12204	2279	1371.88	217.14	2976.69	379.70	3356.39
steam2	1177	28	2.54	0.85	4.80	1.84	6.63
fffff800	15337	> 1050	42.42	40.64	102.40	96.32	198.71
nzfri	13099	610	627.88	36.08	1294.32	80.93	1375.25
pilot4	3568	22	12.96	0.43	31.42	0.91	32.33
scsd6	2368	11	6.16	0.19	14.18	0.43	14.61
seba	13277	73	61.86	2.46	148.50	5.79	154.29
shell	4003	16	74.21	0.37	177.87	0.65	178.52
standata	3205	75	27.91	1.11	66.33	2.32	68.65
woodw	10073	27	471.27	2.99	972.28	6.26	978.54

Table A.2 IMGS with dynamic pattern selection strategy

Table A.3 shows the results of ICGS with the static pattern of the normal equations. There are 8 problems for which the iteration does not converge within the limit $l = 2n$.

name	nnz(R)	iter	T(ICGS)	T(CGLS)	F(ICGS)	F(CGLS)	F(TOTAL)
amoco1	27686	>660	56.65	72.29	78.72	170.73	249.45
belladit	2395	15	0.74	0.11	1.20	0.26	1.46
bellmedt	372255	16	1977.90	13.20	2775.98	28.44	2804.42
illc1850	5633	588	3.96	22.33	3.22	39.90	43.13
well1850	5633	197	3.90	7.50	3.22	13.39	16.61
well1033	2469	97	1.34	1.82	1.24	3.33	4.57
dunes8	6998	20	20.03	1.47	25.44	3.15	28.59
conev8	5135	1	10.60	0.08	16.60	0.14	16.74
strat8	21757	89	359.23	21.96	370.26	42.52	412.78
bnl1	28005	251	27.69	16.37	44.38	39.58	83.96
ship12s	6388	39	2.80	1.47	1.16	2.92	4.08
gen	5816	41	2.24	0.98	2.35	1.88	4.23
fs_760_1	13957	4	9.42	0.19	11.37	0.38	11.75
fs_760_3	13957	>1520	9.46	59.82	11.37	130.39	141.76
gre_115	692	34	0.05	0.10	0.06	0.18	0.24
hwatt_2	27445	1	24.59	0.15	38.11	0.25	38.36
mc_fe	73254	519	37.44	81.71	62.55	206.03	268.57
orsreg_1	24203	>4410	29.96	373.49	36.99	754.03	791.02
pde_9511	6420	39	3.87	1.11	5.39	2.05	7.45
pores_2	12723	>2448	11.45	116.94	15.57	242.66	258.23
saylr4	38793	>7128	70.92	989.77	98.09	1945.59	2043.68
steam2	20237	7	9.39	0.46	13.14	1.05	14.20
fffff800	10625	>1050	5.44	33.79	7.48	76.52	83.99
nzfri	8406	57	37.30	3.04	52.24	6.54	58.78
pilot4	6899	64	3.51	1.53	5.64	3.47	9.11
scsd6	2248	21	1.98	0.35	3.32	0.80	4.12
seba	52432	>1032	44.27	96.13	72.66	243.01	315.67
shell	2748	21	3.00	0.44	2.90	0.74	3.63
standata	1833	>720	1.86	9.27	2.17	18.16	20.34
woodw	21525	34	82.65	4.23	109.56	9.44	118.99

Table A.3 ICGS with static pattern of the normal equations

Table A.4 shows the results of IMGS with the static pattern of the normal equations. There are 5 problems for which the iteration does not converge within the limit $l = 2n$.

name	nnz(R)	iter	T(IMGS)	T(CGLS)	F(IMGS)	F(CGLS)	F(TOTAL)
amoco1	27685	140	71.17	15.31	145.55	36.31	181.86
belladit	2394	14	1.01	0.10	2.24	0.24	2.48
bellmedt	372254	16	2846.74	12.99	5475.47	28.44	5503.91
illc1850	5632	305	3.04	11.31	4.71	20.71	25.42
well1850	5632	180	3.06	6.69	4.71	12.23	16.94
well1033	2468	87	1.03	1.61	1.77	2.99	4.76
dunes8	6997	20	23.92	1.48	45.66	3.15	48.81
conev8	5134	1	15.34	0.08	30.61	0.14	30.75
strat8	21756	89	350.54	18.74	317.46	42.52	359.98
bnl1	28004	162	41.65	10.40	86.22	25.57	111.79
ship12s	6387	35	2.79	1.31	1.96	2.63	4.59
gen	5815	41	2.85	0.97	4.38	1.88	6.26
fs_760_1	13956	4	10.21	0.19	21.28	0.38	21.66
fs_760_3	13956	> 1520	10.22	59.41	21.28	130.38	151.66
gre_115	691	34	0.05	0.10	0.10	0.18	0.28
hwatt_2	27444	1	35.24	0.16	70.89	0.25	71.14
mc_fe	73253	234	54.58	36.79	118.32	93.00	211.32
orsreg_1	24202	312	35.51	26.34	68.50	53.42	121.92
pde_9511	6419	26	4.93	0.75	9.58	1.38	10.95
pores_2	12722	> 2448	14.33	116.03	28.69	242.65	271.35
saylr4	38792	> 7128	95.64	972.22	182.24	1945.56	2127.79
steam2	20236	7	10.62	0.45	24.64	1.05	25.70
fffff800	10624	> 1050	6.61	33.56	13.83	76.51	90.34
nzfri	8405	57	48.38	2.97	98.27	6.54	104.81
pilot4	6898	61	5.16	1.45	10.59	3.31	13.89
scsd6	2247	21	2.82	0.35	6.23	0.80	7.03
seba	52431	70	67.74	6.53	142.32	16.59	158.91
shell	2747	23	2.72	0.48	4.65	0.81	5.45
standata	1832	> 720	1.87	9.15	3.55	18.16	21.71
woodw	21524	34	108.18	4.20	207.19	9.44	216.63

Table A.4 IMGS with static pattern of the normal equations

Table A.5 shows the results of ICGS with semidynamic pattern selection where the maximum number of elements in a row is $max(10, n/100)$. There are 5 problems for which the iteration does not converge within the limit $l = 2n$.

name	nnz(R)	iter	T(ICGS)	T(CGLS)	F(ICGS)	f(CGLS)	F(TOTAL)
amoco1	3573	>660	21.78	48.83	17.86	107.02	124.89
belladit	848	19	0.31	0.10	0.41	0.20	0.61
bellmedt	11301	34	157.61	4.81	106.70	9.66	116.36
illc1850	4562	781	10.13	28.16	8.63	49.64	58.27
well1850	4552	123	10.24	4.45	8.72	7.84	16.56
well1033	1555	98	1.83	1.68	1.80	3.01	4.81
dunes8	6768	21	26.70	1.51	32.37	3.29	35.66
conev8	5125	1	11.83	0.08	18.39	0.14	18.53
bnl1	5201	462	5.24	14.58	5.21	30.60	35.80
strat8	50279	20	553.81	5.26	860.33	12.05	872.39
ship12s	3793	41	2.86	1.38	0.76	2.64	3.40
gen	3911	73	1.31	1.51	0.39	2.78	3.17
fs_760_1	7556	2	8.43	0.09	7.96	0.15	8.10
fs_760_3	7560	>1520	9.74	45.34	9.50	91.48	100.99
gre_115	1208	23	0.14	0.09	0.18	0.17	0.35
hwatt_2	35094	1	43.98	0.18	68.91	0.30	69.21
mc_fe	8236	1182	9.75	72.67	9.67	161.43	171.10
orsreg_1	50250	196	71.62	24.66	97.15	54.06	151.22
pde_9511	10318	20	6.81	0.70	10.03	1.38	11.41
pores_2	15505	>2448	14.62	124.62	19.07	269.91	288.98
saylr4	126626	468	320.31	127.75	489.56	292.46	782.02
steam2	6111	11	8.12	0.45	7.43	0.97	8.40
fffff800	5160	>1050	5.02	25.43	5.03	53.55	58.58
nzfri	6750	329	36.45	16.57	47.24	35.31	82.55
pilot4	3600	52	2.45	0.99	3.18	2.13	5.31
scsd6	1527	24	1.43	0.37	2.22	0.85	3.07
seba	5622	>1032	7.95	23.42	8.78	49.69	58.47
shell	4813	20	8.18	0.49	9.99	0.87	10.86
standata	3330	251	3.55	3.81	4.91	7.84	12.76
woodw	8957	60	41.45	6.74	44.14	13.52	57.65

Table A.5 ICGS with semidynamic pattern

Table A.6 shows the results of IMGS with semidynamic pattern selection where the maximum number of elements in a row is $max(10, n/100)$. There are 9 problems for which iteration does not converge within the limit $l = 2n$.

name	nnz(R)	iter	T(IMGS)	T(CGLS)	F(IMGS)	F(CGLS)	F(TOTAL)
amoco1	3576	$>$ 660	147.96	49.48	294.37	107.03	401.40
belladit	848	19	1.51	0.10	3.29	0.20	3.50
bellmedt	11309	33	4099.38	4.70	7805.27	9.38	7814.65
illc1850	7789	$>$ 1426	232.56	57.74	488.45	109.03	597.48
well1850	7789	201	231.67	8.24	487.15	15.40	502.54
well1033	3477	257	14.46	5.15	29.87	9.85	39.72
dunes8	8427	26	1228.40	1.97	2784.92	4.23	2789.15
conev8	5270	1	173.49	0.08	564.43	0.14	564.57
bnl1	6898	517	140.64	17.67	309.21	37.75	346.96
ship12s	11408	39	12.47	1.80	29.60	3.72	33.31
gen	8526	181	34.14	5.03	71.16	10.22	81.38
fs_760_1	8306	2	312.08	0.09	656.58	0.16	656.73
fs_760_3	8306	$>$ 1520	311.47	47.31	656.58	96.02	752.60
gre_115	1211	24	0.79	0.09	1.82	0.18	2.01
hwatt_2	35094	2	1375.77	0.27	4707.26	0.50	4707.76
mc_fe	8361	$>$ 1530	97.53	94.74	296.32	209.71	506.03
orsreg_1	50463	6	4775.32	0.87	10657.67	1.79	10659.46
pde_9511	10517	17	167.07	0.61	648.67	1.20	649.86
pores_2	15835	$>$ 2448	651.79	127.70	1741.98	273.14	2015.13
saylr4	127675	2538	11435.43	707.45	35002.63	1595.33	36597.97
steam2	6546	11	108.81	0.45	265.52	0.99	266.51
fffff800	5721	$>$ 1050	102.28	26.48	232.66	55.91	288.57
nzfri	6810	$>$ 1248	1315.84	62.56	2518.32	134.11	2652.43
pilot4	4467	59	37.83	1.21	101.35	2.62	103.97
scsd6	1574	27	6.68	0.42	22.05	0.96	23.01
seba	5622	$>$ 1032	190.29	23.56	391.63	49.69	441.32
shell	5853	178	192.47	4.47	457.71	8.36	466.08
standata	3906	$>$ 720	72.96	11.46	163.95	24.14	188.09
woodw	12035	84	2183.60	9.08	5677.39	19.93	5697.32

Table A.6 IMGS with semidynamic pattern

Table A.7 shows the results of ILQ. Here the maximum number of elements in each column of R is $max(10, n/100)$ and the maximum number of elements each column of Q is $max(20, 2m/100)$. There are problems for which the iteration does not converge within the limit $l = 2n$.

name	nnz(R)	iter	T(ICGS)	T(CGLS)	F(ICGS)	F(CGLS)	F(TOTAL)
amoco1	3464	>660	1.19	48.66	0.39	106.74	107.13
belladit	826	22	0.11	0.11	0.04	0.23	0.28
bellmedt	11281	33	27.68	4.62	5.83	9.38	15.20
illc1850	4331	1171	1.89	41.61	0.39	73.34	73.72
well1850	4317	174	1.87	6.21	0.39	10.91	11.30
well1033	1168	182	0.39	3.01	0.06	5.29	5.35
dunes8	6657	21	9.72	1.50	1.77	3.28	5.04
conev8	4467	1	3.42	0.07	0.69	0.14	0.83
strat8	50279	19	203.29	5.42	35.70	11.46	47.16
bnl1	5094	604	1.84	19.05	0.53	39.73	40.27
ship12s	3788	41	3.40	1.38	0.26	2.64	2.90
gen	3878	74	1.38	1.52	0.12	2.81	2.93
fs_760_1	7433	2	1.25	0.08	0.42	0.15	0.57
fs_760_3	7400	>1520	1.32	44.96	0.42	90.51	90.93
gre_115	1122	24	0.09	0.09	0.05	0.17	0.23
hwatt_2	34755	1	8.74	0.17	2.88	0.29	3.17
mc_fe	8191	>1530	2.06	94.04	0.93	208.67	209.59
orsreg_1	45954	1349	12.23	157.37	4.50	348.12	352.62
pde_9511	10055	23	1.73	0.81	0.47	1.56	2.03
pores_2	13370	>2448	2.87	117.90	0.81	249.00	249.81
saylr4	114797	3056	43.45	768.35	17.05	1763.43	1780.48
steam2	5768	11	1.10	0.44	0.50	0.95	1.45
fffff800	4865	>1050	0.93	25.27	0.28	52.31	52.59
nzfri	5443	78	5.09	3.80	0.87	8.00	8.87
pilot4	3288	56	0.66	1.06	0.16	2.22	2.38
scsd6	1402	33	0.40	0.54	0.09	1.14	1.23
seba	4430	>1032	1.29	21.50	0.56	44.77	45.33
shell	3706	26	1.53	0.61	0.32	1.01	1.33
standata	2160	293	0.51	3.93	0.10	7.78	7.88
woodw	8856	69	28.57	7.15	3.02	15.50	18.52

Table A.7 ICGS with semidynamic pattern plus dropping on Q

Table A.8 shows the results of IMGS (column version) with semidynamic pattern selection and dropping on Q. Here the maximum number or elements each column of R is $max(10, n/100)$ and the maximum number of elements each column of Q is $max(20, 2m/100)$. There are 5 problems for which the iteration does not converge within the limit $l = 2n$.

name	nnz(R)	iter	T(IMGS)	T(CGLS)	F(IMGS)	F(CGLS)	F(TOTAL)
amoco1	3576	>660	4.39	48.86	6.11	107.03	113.14
belladit	848	19	0.24	0.10	0.27	0.20	0.48
bellmedt	11309	34	151.68	5.01	247.15	9.66	256.80
illc1850	7789	508	10.38	20.85	19.38	38.86	58.24
well1850	7789	142	10.25	5.81	19.32	10.89	30.21
well1033	3477	185	1.14	3.74	1.57	7.09	8.67
dunes8	8427	22	52.07	1.64	122.29	3.59	125.88
conev8	5270	1	11.71	0.08	30.87	0.14	31.01
strat8	50463	18	948.20	4.85	3163.88	10.89	3174.77
bnl1	6898	454	9.26	15.64	16.39	33.15	49.54
ship12s	11408	37	9.57	1.69	12.54	3.53	16.07
gen	8526	215	6.02	5.87	10.10	12.13	22.23
fs_760_1	8306	2	11.18	0.09	21.90	0.16	22.05
fs_760_3	8306	>1520	11.74	47.38	21.95	96.02	117.97
gre_115	1211	23	0.31	0.09	0.52	0.17	0.69
hwatt_2	35094	2	77.09	0.27	254.98	0.50	255.48
mc_fe	8361	>1530	9.23	94.37	23.07	209.71	232.78
orsreg_1	50463	26	165.40	3.31	425.43	7.31	432.74
pde_9511	10517	22	12.36	0.76	36.65	1.54	38.19
pores_2	15835	>2448	27.90	129.58	71.48	273.14	344.62
saylr4	127675	2784	509.98	764.33	1778.96	1749.94	3528.89
steam2	6546	12	6.80	0.50	14.33	1.07	15.40
fffff800	5721	>1050	5.76	26.80	10.00	55.91	65.91
nzfri	6810	87	30.56	4.40	54.36	9.39	63.75
pilot4	4467	61	2.88	1.27	5.29	2.71	8.00
scsd6	1574	30	0.77	0.47	1.18	1.06	2.24
seba	5622	384	5.16	8.65	9.49	18.50	28.00
shell	5853	25	7.83	0.65	19.60	1.19	20.79
standata	3906	461	2.30	7.48	4.77	15.46	20.23
woodw	12035	94	138.96	10.13	250.42	22.29	272.71

Table A.8 IMGS with semidynamic pattern (column version) plus dropping on Q

Table A.9 shows the results of IMGS (row version) with semidynamic pattern selection and dropping on Q. The maximum number of elements in each column of R and Q are $max(10, n/100)$ and $max(20, 2m/100)$ respectively. The problem HWATT_2 breaks down during the factorization. There are 19 other problems for which the iteration does not converge within the limit $l = 2n$. There are only 10 problems successfully solved.

name	nnz(R)	iter	T(IMGS)	T(CGLS)	F(IMGS)	F(CGLS)	F(TOTAL)
amoco1	3576	>660	5.81	48.89	0.22	107.03	107.25
belladit	848	38	0.38	0.19	0.04	0.41	0.44
bellmedt	11309	92	131.72	12.78	2.46	25.92	28.38
illc1850	7789	> 1426	14.96	58.09	0.38	109.03	109.41
well1850	7789	>1426	15.09	57.83	0.38	109.03	109.41
well1033	3477	585	2.92	11.72	0.06	22.39	22.46
dunes8	8427	111	55.52	8.04	1.26	17.83	19.09
conev8	5270	1	17.83	0.07	0.55	0.14	0.70
strat8	50463	1012	872.17	253.95	13.04	597.70	610.74
bnl1	6898	>1264	11.79	42.81	0.40	92.24	92.64
ship12s	11408	536	25.82	24.12	0.21	50.56	50.76
gen	8526	>1560	10.76	42.37	0.11	87.86	87.97
fs_760_1	8306	>1520	8.50	46.52	0.30	96.02	96.32
fs_760_3	8306	>1520	8.60	46.58	0.31	96.02	96.33
gre_115	1211	227	0.20	0.82	0.02	1.69	1.71
hwatt_2	—-	—-	—-	—-	—-	—-	—-
mc_fe	8361	>1530	9.44	94.05	0.37	209.71	210.08
orsreg_1	50463	>4410	100.42	526.56	1.44	1217.31	1218.75
pde_9511	10517	1150	13.37	38.60	0.41	78.76	79.17
pores_2	15835	>2448	23.44	122.90	0.61	273.14	273.75
saylr4	127675	>7128	700.07	1888.25	17.26	4479.97	4497.23
steam2	6546	>1200	6.17	46.75	0.27	103.19	103.46
ffffff800	5721	> 1050	5.59	26.33	0.20	55.91	56.11
nzfri	6810	>1248	28.67	62.57	0.81	134.11	134.92
pilot4	4467	>822	4.24	16.60	0.14	36.25	36.39
scsd6	1574	51	1.85	0.78	0.09	1.79	1.88
seba	5622	>1032	5.89	23.45	0.28	49.69	49.97
shell	5853	>1074	10.35	26.64	0.29	50.37	50.65
standata	3906	> 720	3.55	11.46	0.11	24.14	24.24
woodw	12035	90	92.24	9.66	0.52	21.35	21.87

Table A.9 IMGS with semidynamic patter (row version) plus dropping on Q

Table A.10 shows the comparison results of column weight ordering applied to IMGS with static pattern selection. The problem **standata** breaks down during the factorization.

	no reorder			column weight ordering		
name	iter	F(IMGS)	F(TOTAL)	iter	F(IMGS)	F(TOTAL)
amoco1	140	145.55	181.86	88	76.25	99.12
belladit	14	2.24	2.48	12	1.71	1.92
bellmedt	16	5475.47	5503.91	15	4951.00	4977.72
illc1850	305	4.71	25.42	194	0.63	13.81
well1850	180	4.71	16.94	159	0.62	11.43
well1033	87	1.77	4.76	53	0.18	2.01
dunes8	20	45.66	48.81	19	21.24	24.24
conev8	1	30.61	30.75	1	20.57	20.71
strat8	89	317.46	359.98	296	131.78	272.74
bnl1	162	86.22	111.79	200	92.49	124.04
ship12s	35	1.96	4.59	30	0.85	3.10
gen	41	4.38	6.26	32	3.21	4.68
fs_760_1	4	21.28	21.66	5	19.97	20.44
fs_760_3	> 1520	21.28	151.66	> 1520	20.06	150.45
gre_115	34	0.10	0.28	27	0.07	0.21
hwatt_2	1	70.89	71.14	1	73.90	74.15
mc_fe	234	118.32	211.32	26	118.10	128.61
orsreg_1	312	68.50	121.92	443	102.11	177.93
pde_9511	26	9.58	10.95	31	10.29	11.93
pores_2	> 2448	28.69	271.35	> 2448	15.23	257.89
saylr4	> 7128	182.24	2127.79	> 7128	180.02	2125.60
steam2	7	24.64	25.70	6	23.81	24.73
fffff800	> 1050	13.83	90.34	> 1050	5.12	81.64
nzfri	57	98.27	104.81	46	3.85	9.13
pilot4	61	10.59	13.89	30	2.99	4.63
scsd6	21	6.23	7.03	22	1.69	2.53
seba	70	142.32	158.91	> 1032	68.44	311.46
shell	23	4.65	5.45	24	0.30	1.14
woodw	34	207.19	216.63	37	43.22	53.47

Table A.10 Performance of column weight ordering on IMGS with static pattern

Table A.11 shows the results of nested dissection ordering applied to IMGS with static pattern selection of the normal equations. The problem FFFFF800 breaks down during the factorization. There are 4 problems for which iteration does not converge within the limit $l = 2n$.

	no reorder			nested disection		
name	iter	F(IMGS)	F(TOTAL)	iter	F(IMGS)	F(TOTAL)
amoco1	140	145.55	181.86	111	66.76	95.58
belladit	14	2.24	2.48	12	1.89	2.10
bellmedt	16	5475.47	5503.91	15	5432.49	5459.20
illc1850	305	4.71	25.42	223	0.98	16.13
well1850	180	4.71	16.94	177	0.97	13.01
well1033	87	1.77	4.76	73	0.30	2.81
dunes8	20	45.66	48.81	20	7.37	10.52
conev8	1	30.61	30.75	1	5.44	5.58
strat8	89	317.46	359.98	286	30.55	166.75
bnl1	162	86.22	111.79	236	99.33	136.55
ship12s	35	1.96	4.59	34	1.94	4.49
gen	41	4.38	6.26	60	4.48	7.23
fs_760_1	4	21.28	21.66	5	7.03	7.50
fs_760_3	> 1520	21.28	151.66	> 1520	7.04	137.42
gre_115	34	0.10	0.28	31	0.07	0.23
hwatt_2	1	70.89	71.14	1	14.23	14.48
mc_fe	234	118.32	211.32	35	67.72	81.80
orsreg_1	312	68.50	121.92	4200	15.15	733.28
pde_9511	26	9.58	10.95	85	0.92	5.37
pores_2	> 2448	28.69	271.35	> 2448	6.94	249.60
saylr4	> 7128	182.24	2127.79	> 7128	29.32	1974.90
steam2	7	24.64	25.70	7	10.95	12.00
nzfri	57	98.27	104.81	51	13.57	19.43
pilot4	61	10.59	13.89	47	5.38	7.94
scsd6	21	6.23	7.03	23	1.60	2.48
seba	70	142.32	158.91	30	71.70	78.88
shell	23	4.65	5.45	27	2.20	3.14
standata	> 720	3.55	21.71	> 720	0.59	18.76
woodw	34	207.19	216.63	43	235.92	247.82

Table A.11 Performance of nested disection ordering: IMGS with static pattern

Table A.12 shows the results of reverse Cutlhill McKee ordering applied to IMGS with static pattern selection of the normal equations. The problem STANDATA breaks down during the factorization. There are 5 problems for which the iteration does not converge within the limit $l = 2n$.

	no reorder			RCM ordering		
name	iter	F(IMGS)	F(TOTAL)	iter	F(IMGS)	F(TOTAL)
amoco1	140	145.55	181.86	107	80.63	108.41
belladit	14	2.24	2.48	12	1.78	1.99
bellmedt	16	5475.47	5503.91	15	5037.21	5063.92
illc1850	305	4.71	25.42	194	0.67	13.85
well1850	180	4.71	16.94	160	0.67	11.55
well1033	87	1.77	4.76	55	0.20	2.09
dunes8	20	45.66	48.81	13	58.54	60.61
conev8	1	30.61	30.75	1	28.26	28.40
strat8	89	317.46	359.98	172	334.89	416.88
bnl1	162	86.22	111.79	189	108.80	138.62
ship12s	35	1.96	4.59	26	1.38	3.34
gen	41	4.38	6.26	50	5.32	7.61
fs_760_1	4	21.28	21.66	6	19.09	19.64
fs_760_3	> 1520	21.28	151.66	> 1520	19.09	149.48
gre_115	34	0.10	0.28	26	0.10	0.24
hwatt_2	1	70.89	71.14	1	51.09	51.34
mc_fe	234	118.32	211.32	24	93.15	102.87
orsreg_1	312	68.50	121.92	360	68.70	130.33
pde_9511	26	9.58	10.95	39	9.47	11.52
pores_2	> 2448	28.69	271.35	> 2448	16.80	259.47
saylr4	> 7128	182.24	2127.79	> 7128	242.08	2187.66
steam2	7	24.64	25.70	7	20.50	21.55
fffff800	> 1050	13.83	90.34	> 1050	9.73	86.25
nzfri	57	98.27	104.81	452	5.56	57.05
pilot4	61	10.59	13.89	30	11.76	13.40
scsd6	21	6.23	7.03	22	2.21	3.05
seba	70	142.32	158.91	> 1032	65.88	308.89
shell	23	4.65	5.45	24	2.18	3.02
woodw	34	207.19	216.63	36	238.62	248.60

Table A.12 Performance of reverse Cuthill McKee (RCM) ordering: IMGS with static pattern

Table A.13 shows the results of minimum degree ordering applied to IMGS with static pattern selection of the normal equations. The problem **standata** breaks down during the factorization. There are 5 problems for which the iteration does not converge within the limit $l = 2n$.

	no reorder			minimum degree ordering		
name	iter	F(IMGS)	F(TOTAL)	iter	F(IMGS)	F(TOTAL)
amoco1	140	145.55	181.86	66	84.25	101.44
belladit	14	2.24	2.48	12	1.86	2.07
bellmedt	16	5475.47	5503.91	15	5345.51	5372.22
illc1850	305	4.71	25.42	224	0.71	15.92
well1850	180	4.71	16.94	161	0.70	11.65
well1033	87	1.77	4.76	54	0.33	2.20
dunes8	20	45.66	48.81	19	14.60	17.59
conev8	1	30.61	30.75	1	7.31	7.45
strat8	89	317.46	359.98	278	88.34	220.74
bnl1	162	86.22	111.79	191	93.71	123.85
ship12s	35	1.96	4.59	35	1.96	4.59
gen	41	4.38	6.26	35	4.91	6.52
fs_760_1	4	21.28	21.66	4	10.65	11.03
fs_760_3	> 1520	21.28	151.66	> 1520	10.67	141.06
gre_115	34	0.10	0.28	29	0.08	0.24
hwatt_2	1	70.89	71.14	1	22.03	22.28
mc_fe	234	118.32	211.32	18	100.95	108.29
orsreg_1	312	68.50	121.92	4179	22.54	737.06
pde_9511	26	9.58	10.95	89	1.02	5.68
pores_2	> 2448	28.69	271.35	> 2448	7.50	250.15
saylr4	> 7128	182.24	2127.79	> 7128	48.09	1993.65
steam2	7	24.64	25.70	7	14.04	15.10
fffff800	> 1050	13.83	90.34	> 1050	4.93	81.44
nzfri	57	98.27	104.81	45	6.69	11.86
pilot4	61	10.59	13.89	33	6.86	8.66
scsd6	21	6.23	7.03	21	1.78	2.58
seba	70	142.32	158.91	> 1032	83.10	326.11
shell	23	4.65	5.45	24	0.49	1.33
woodw	34	207.19	216.63	40	39.97	51.05

Table A.13 Performance of minimum degree ordering: IMGS with static pattern

Table A.14 shows comparison results of column weight ordering applied to IMGS with dynamic pattern selection ($\epsilon = 0.02$). The problem STANDATA breaks down during the factorization. In the table, 'n.o.' is for 'no ordering' while 'c.w.o.' is for 'column weight ordering'. There are 4 problems for which the iteration does not converge within the limit $l = 2n$.

Table A.15 shows the results of nested dissection ordering applied to IMGS with dynamic pattern selection ($\epsilon = 0.02$). The problem FFFFF800 breaks down during the factorization. There are 4 problems for which the iteration does not converge within the limit $l = 2n$. Here 'nestd' is for 'nested dissection'.

Table A.16 shows the results of reverse Cutlhill McKee ordering applied to IMGS with dynamic pattern selection ($\epsilon = 0.02$). The problem STANDATA breaks down during the factorization. There are 5 problems for which the iteration does not converge. within the limit $l = 2n$. In the table, 'rcm' is for reverse Cuthill McKee ordering.

Table A.17 shows the results of minimum degree ordering applied to IMGS with dynamic pattern selection ($\epsilon = 0.02$). The problem STANDATA breaks down during the factorization. There are 4 problems for which the iteration does not converge within the limit $l = 2n$. In the table, 'mmd' is for minimum degree ordering.

Table A.14 Performance of column weight ordering: IMGS with dynamic pattern

	nnz(R)		F(IMGS)		iter		F(CGLS)		F(TOTAL)	
name	n.o.	c.w.o.	n.o.	c.w.o.	n.o.	c.w.o.	n.o.	c.w.o.	n.o.	c.w.o.
amoco1	19365	16279	194.41	73.10	173	150	39.07	32.03	233.48	105.14
belladit	2226	2119	2.66	1.85	8	8	0.14	0.13	2.79	1.99
bellmedt	64186	62257	4251.12	3289.04	18	19	9.09	9.43	4260.20	3298.47
illc1850	6654	4957	120.38	10.68	87	60	6.29	3.93	126.67	14.61
well1850	6971	5573	149.75	12.67	37	30	2.74	2.06	152.49	14.72
well1033	2983	2097	16.87	1.08	38	27	1.39	0.90	18.26	1.98
dunes8	7416	7245	682.05	323.26	13	11	2.09	1.77	684.14	325.04
conev8	3656	3472	264.47	102.68	1	1	0.13	0.13	264.61	102.81
bnl1	15325	11597	129.18	99.52	103	104	11.04	9.58	140.22	109.10
ship12s	3885	3747	11.19	9.88	26	27	1.69	1.74	12.89	11.62
gen	3817	3436	6.61	4.46	32	34	1.21	1.24	7.83	5.69
fs_760_1	852	886	5.54	3.19	7	8	0.25	0.28	5.79	3.47
fs_760_3	8036	9238	174.80	117.14	> 1520	> 1520	94.38	101.69	269.18	218.83
gre_115	1363	846	0.92	0.25	13	11	0.11	0.07	1.03	0.32
hwatt_2	21221	21206	2060.69	1710.88	1	1	0.21	0.21	2060.90	1711.09
mc_fe	11586	7661	148.59	49.91	717	931	107.57	125.02	256.16	174.94
orsreg_1	5671	5605	84.12	55.71	507	694	49.14	67.06	133.26	122.77
pde_9511	9644	9479	295.14	339.57	17	19	1.13	1.25	296.28	340.82
pores_2	5565	3335	71.35	16.44	> 2448	> 2448	172.56	150.72	243.91	167.15
saylr4	12204	12498	2976.69	656.54	2279	4108	379.70	689.20	3356.39	1345.73
steam2	1177	1308	4.80	6.23	28	26	1.84	1.72	6.63	7.95
fffff800	15337	5278	102.40	6.66	> 1050	> 1050	96.32	54.05	198.71	60.70
nzfri	13099	7121	1294.32	34.40	610	31	80.93	3.42	1375.25	37.82
pilot4	3568	2971	31.42	6.38	22	24	0.91	0.93	32.33	7.31
scsd6	2368	2288	14.18	5.05	11	11	0.43	0.43	14.61	5.48
seba	13277	8579	148.50	16.74	73	> 1032	5.79	61.90	154.29	78.65
shell	4003	3034	177.87	5.44	16	11	0.65	0.41	178.52	5.85
standata										
woodw	10073	9843	972.28	102.49	27	27	6.26	6.23	978.54	108.73

Table A.15 Performance of nested disection ordering: IMGS with dynamic pattern

	nnz(R)		F(IMGS)		iter		F(CGLS)		F(TOTAL)	
name	n.o.	nestd	n.o.	nestd	n.o.	nestd	n.o.	nestd	n.o.	nestd
amoco1	19365	17023	194.41	69.53	173	154	39.07	33.34	233.48	102.88
belladit	2226	2134	2.66	2.06	8	8	0.14	0.13	2.79	2.19
bellmedt	64186	63340	4251.12	4045.33	18	18	9.09	9.02	4260.20	4054.36
illc1850	6654	4950	120.38	33.93	87	61	6.29	4.00	126.67	37.93
well1850	6971	5818	149.75	47.12	37	33	2.74	2.29	152.49	49.41
well1033	2983	2190	16.87	2.10	38	29	1.39	0.97	18.26	3.07
dunes8	7416	7560	682.05	190.05	13	12	2.09	1.95	684.14	192.00
conev8	3656	3866	264.47	63.64	1	1	0.13	0.13	264.61	63.78
bnl1	15325	15935	129.18	150.11	103	85	11.04	9.32	140.22	159.43
ship12s	3885	3803	11.19	33.25	26	27	1.69	1.75	12.89	35.00
gen	3817	3942	6.61	7.48	32	33	1.21	1.27	7.83	8.74
fs_760_1	852	878	5.54	4.34	7	8	0.25	0.28	5.79	4.62
fs_760_3	8036	8780	174.80	51.95	> 1520	> 1520	94.38	98.90	269.18	150.85
gre_115	1363	953	0.92	0.37	13	11	0.11	0.07	1.03	0.45
hwatt_2	21221	22822	2060.69	497.10	1	1	0.21	0.22	2060.90	497.33
mc_fe	11586	7364	148.59	67.25	717	932	107.57	124.05	256.16	191.30
orsreg_1	5671	6293	84.12	72.79	507	607	49.14	60.33	133.26	133.12
pde_9511	9644	11218	295.14	45.78	17	24	1.13	1.74	296.28	47.52
pores_2	5565	4176	71.35	19.27	> 2448	> 2448	172.56	158.95	243.91	178.23
saylr4	12204	14471	2976.69	297.72	2279	3663	379.70	643.46	3356.39	941.18
steam2	1177	1364	4.80	7.69	28	22	1.84	1.47	6.63	9.16
fffff800										
nzfri	13099	8237	1294.32	106.39	610	105	80.93	11.93	1375.25	118.32
pilot4	3568	3777	31.42	15.99	22	25	0.91	1.05	32.33	17.04
scsd6	2368	2331	14.18	5.57	11	11	0.43	0.43	14.61	6.01
seba	13277	8300	148.50	11.59	73	> 1032	5.79	60.75	154.29	72.34
shell	4003	3639	177.87	41.29	16	13	0.65	0.51	178.52	41.80
standata	3205	2352	66.33	3.75	75	> 720	2.32	19.66	68.65	23.40
woodw	10073	10289	972.28	1673.35	27	26	6.26	6.05	978.54	1679.40

Table A.16 Performance of reverse Cuthill McKee (RCM) ordering: IMGS with dynamic pattern

	nnz(R)		F(IMGS)		iter		F(CGLS)		F(TOTAL)	
name	n.o.	rcm	n.o.	rcm	n.o.	rcm	n.o.	rcm	n.o.	rcm
amoco1	19365	16218	194.41	83.43	173	153	39.07	32.63	233.48	116.06
belladit	2226	2112	2.66	1.95	8	8	0.14	0.13	2.79	2.08
bellmedt	64186	62291	4251.12	3476.81	18	18	9.09	8.95	4260.20	3485.76
illc1850	6654	4434	120.38	9.76	87	59	6.29	3.75	126.67	13.50
well1850	6971	5577	149.75	13.18	37	31	2.74	2.12	152.49	15.30
well1033	2983	2091	16.87	1.18	38	27	1.39	0.90	18.26	2.07
dunes8	7416	6890	682.05	882.97	13	10	2.09	1.60	684.14	884.57
conev8	3656	3416	264.47	216.70	1	1	0.13	0.13	264.61	216.83
bnl1	15325	12461	129.18	175.87	103	96	11.04	9.18	140.22	185.06
ship12s	3885	3765	11.19	19.66	26	29	1.69	1.87	12.89	21.53
gen	3817	4590	6.61	35.93	32	33	1.21	1.36	7.83	37.29
fs_760_1	852	886	5.54	4.73	7	8	0.25	0.28	5.79	5.01
fs_760_3	8036	7585	174.80	133.54	> 1520	> 1520	94.38	91.64	269.18	225.18
gre_115	1363	811	0.92	0.49	13	10	0.11	0.06	1.03	0.55
hwatt_2	21221	21626	2060.69	1823.19	1	1	0.21	0.22	2060.90	1823.40
mc_fe	11586	7491	148.59	66.73	717	945	107.57	126.26	256.16	192.99
orsreg_1	5671	6465	84.12	510.91	507	461	49.14	46.15	133.26	557.06
pde_9511	9644	9517	295.14	302.83	17	13	1.13	0.87	296.28	303.70
pores_2	5565	3711	71.35	33.51	> 2448	> 2448	172.56	154.40	243.91	187.91
saylr4	12204	11654	2976.69	1566.47	2279	1732	379.70	284.77	3356.39	1851.25
steam2	1177	1385	4.80	7.94	28	21	1.84	1.40	6.63	9.34
fffff800	15337	8356	102.40	42.80	> 1050	> 1050	96.32	66.98	198.71	109.78
nzfri	13099	7086	1294.32	46.47	610	> 1248	80.93	135.49	1375.25	181.96
pilot4	3568	3226	31.42	27.55	22	24	0.91	0.96	32.33	28.50
scsd6	2368	2358	14.18	6.14	11	11	0.43	0.43	14.61	6.58
seba	13277	8595	148.50	16.11	73	> 1032	5.79	61.97	154.29	78.08
shell	4003	2963	177.87	46.02	16	11	0.65	0.40	178.52	46.42
standata										
woodw	10073	10113	972.28	1266.57	27	22	6.26	5.12	978.54	1271.69

	nnz(R)		F(IMGS)		iter		F(CGLS)		F(TOTAL)	
name	n.o.	mmd	n.o.	mmd	n.o.	mmd	n.o.	mmd	n.o.	mmd
amoco1	19365	16380	194.41	86.48	173	160	39.07	34.23	233.48	120.70
belladit	2226	2110	2.66	1.99	8	8	0.14	0.13	2.79	2.12
bellmedt	64186	63642	4251.12	3946.35	18	18	9.09	9.05	4260.20	3955.40
illc1850	6654	4873	120.38	10.17	87	52	6.29	3.40	126.67	13.56
well1850	6971	5413	149.75	11.06	37	29	2.74	1.97	152.49	13.03
well1033	2983	2070	16.87	1.25	38	23	1.39	0.76	18.26	2.02
dunes8	7416	7394	682.05	202.45	13	11	2.09	1.78	684.14	204.23
conev8	3656	3748	264.47	53.09	1	1	0.13	0.13	264.61	53.23
strat8		15862		2923.63		55		25.05		2948.67
bnl1	15325	12189	129.18	104.97	103	92	11.04	8.70	140.22	113.68
ship12s	3885	3905	11.19	10.85	26	27	1.69	1.76	12.89	12.61
gen	3817	3816	6.61	5.78	32	33	1.21	1.25	7.83	7.04
fs_760_1	852	887	5.54	4.05	7	8	0.25	0.28	5.79	4.34
fs_760_3	8036	8046	174.80	57.58	> 1520	> 1520	94.38	94.44	269.18	152.02
gre_115	1363	968	0.92	0.31	13	10	0.11	0.07	1.03	0.38
hwatt_2	21221	22882	2060.69	594.77	1	1	0.21	0.22	2060.90	594.99
mc_fe	11586	8976	148.59	75.52	717	1047	107.57	146.10	256.16	221.62
orsreg_1	5671	6339	84.12	74.13	507	659	49.14	65.62	133.26	139.75
pde_9511	9644	10748	295.14	37.81	17	24	1.13	1.70	296.28	39.51
pores_2	5565	4194	71.35	15.63	> 2448	> 2448	172.56	159.13	243.91	174.76
saylr4	12204	14114	2976.69	282.89	2279	3803	379.70	662.62	3356.39	945.51
steam2	1177	1204	4.80	5.36	28	28	1.84	1.84	6.63	7.20
fffff800	15337	5334	102.40	5.38	> 1050	> 1050	96.32	54.28	198.71	59.67
nzfri	13099	7220	1294.32	39.69	610	31	80.93	3.43	1375.25	43.12
pilot4	3568	2994	31.42	9.38	22	24	0.91	0.93	32.33	10.31
scsd6	2368	2242	14.18	4.07	11	10	0.43	0.39	14.61	4.46
seba	13277	8282	148.50	17.08	73	> 1032	5.79	60.68	154.29	77.76
shell	4003	3085	177.87	6.31	16	11	0.65	0.41	178.52	6.72
standata										
woodw	10073	10104	972.28	116.66	27	27	6.26	6.26	978.54	122.92

Table A.17 Performance of minimum degree ordering: IMGS with dynamic pattern

Table A.18 shows the test results of CIMGS using the static pattern of the normal equations.

CIMGS with static pattern:normal equations

name	iter	T(CIMGS)	T(CGLS)	T(total)	NE	CIMGS	CGLS	Total
amoco1	141	3.24	15.39	18.63	0.93	8.77	36.57	46.27
belladit	14	0.07	0.10	0.17	0.01	0.17	0.24	0.42
bellmedt	16	129.03	12.70	141.73	1.77	324.30	28.44	354.51
illc1850	296	0.80	10.95	11.76	0.05	0.47	20.10	20.63
well1850	180	0.80	6.66	7.46	0.05	0.47	12.24	12.76
well1033	88	0.15	1.63	1.78	0.03	0.12	3.03	3.17
dunes8	20	1.79	1.44	3.23	0.10	1.89	3.15	5.14
conev8	1	0.48	0.07	0.55	0.07	0.38	0.14	0.59
strat8	89	7.28	18.81	26.09	0.32	1.61	42.52	44.45
bnl1	162	3.95	10.42	14.37	0.36	9.54	25.57	35.47
ship12s	35	0.94	1.32	2.26	0.02	0.08	2.63	2.73
gen	41	0.69	0.96	1.65	0.01	0.23	1.88	2.12
fs_760_1	4	4.59	0.19	4.77	0.06	8.40	0.38	8.84
fs_760_3	> 1520	4.63	59.47	64.10	0.06	8.40	130.39	138.85
gre_115	34	0.05	0.10	0.15	0.00	0.07	0.18	0.26
hwatt_2	1	7.44	0.14	7.58	0.08	5.86	0.25	6.19
mc_fe	235	5.26	36.45	41.71	0.97	12.60	93.40	106.97
orsreg_1	312	25.89	26.09	51.97	0.08	30.94	53.42	84.44
pde_9511	26	1.53	0.74	2.27	0.02	0.69	1.38	2.08
pores_2	> 2448	4.94	116.56	121.50	0.07	5.46	242.66	248.20
saylr4	> 7128	34.82	963.76	998.58	0.12	27.26	1945.59	1972.96
steam2	7	3.12	0.45	3.58	0.31	7.02	1.05	8.38
nzfri	57	3.75	2.97	6.72	0.06	7.52	6.54	14.12
pilot4	61	0.60	1.45	2.05	0.06	0.97	3.31	4.34
scsd6	21	0.05	0.35	0.40	0.02	0.08	0.80	0.90
seba	70	9.24	6.47	15.71	0.62	24.92	16.59	42.13
shell	23	0.94	0.48	1.42	0.01	0.87	0.81	1.68
standata	151	0.33	1.93	2.26	0.01	0.29	3.82	4.11
woodw	34	4.28	4.14	8.42	0.18	6.61	9.44	16.23

Table A.18 Performance of CIMGS with static pattern of normal equations

Table A.19 shows the test results of an example of applying the numerical strategy of density reduction on a static pattern.

Table A.20 shows the test results of an example of applying the structural strategy of density reduction on a static pattern.

ϵ	nnz(R)	iter	T(CIMGS)	T(CGLS)	T(total)	NE+CIMGS	CGLS	Total
0.05	34168	24	19.55	4.32	23.87	43.27	9.10	52.37
0.10	11156	30	8.64	4.22	12.86	14.16	8.52	22.68
0.15	5623	37	5.68	4.83	10.51	7.36	9.65	17.00
0.20	3413	41	4.44	5.19	9.63	4.74	10.31	15.05
0.25	2307	46	3.78	5.73	9.51	3.50	11.35	14.85
0.30	1741	50	3.38	6.17	9.55	2.89	12.21	15.10
0.35	1420	52	3.16	6.36	9.52	2.54	12.63	15.17
0.40	1241	53	3.03	6.48	9.51	2.35	12.83	15.18
0.45	1150	56	2.95	6.85	9.79	2.25	13.53	15.78
0.50	1094	56	2.90	6.81	9.71	2.20	13.52	15.72
0.55	1058	57	2.88	6.99	9.87	2.16	13.75	15.91
0.60	1049	59	2.88	7.17	10.05	2.16	14.23	16.38
0.65	1040	59	2.86	7.19	10.05	2.15	14.22	16.37
0.70	1038	61	2.88	7.46	10.34	2.15	14.70	16.85
0.75	1037	61	2.88	7.49	10.37	2.15	14.70	16.85
0.80	1037	61	2.92	7.42	10.34	2.15	14.70	16.85
0.85	1035	67	2.99	8.16	11.15	2.14	16.14	18.28
0.90	1035	67	3.00	8.19	11.18	2.14	16.14	18.28
0.95	1035	67	2.92	8.15	11.08	2.14	16.14	18.28
1.00	1034	79	2.91	9.56	12.47	2.14	19.01	21.15

Table A.19 Example of numerical strategy for the reduction of the density of R for CIMGS with static pattern (BELLMEDT)

ϵ	nnz(R)	iter	T(CIMGS)	T(CGLS)	T(total)	NE+CIMGS	CGLS	Total
0.05	23428	35	14.66	5.64	20.30	30.41	11.66	42.07
0.10	2235	49	3.71	6.08	9.80	3.39	12.07	15.46
0.15	1198	56	2.98	6.84	9.82	2.29	13.54	15.83
0.20	1064	58	2.89	7.09	9.98	2.17	13.99	16.16
0.25	1045	61	2.87	7.40	10.28	2.15	14.70	16.86
0.30	1040	62	2.86	7.56	10.42	2.15	14.94	17.09
0.35	1038	62	2.86	7.56	10.42	2.15	14.94	17.09
0.40	1037	63	2.93	7.73	10.65	2.15	15.18	17.33
0.45	1036	64	2.89	7.76	10.65	2.15	15.42	17.57
0.50	1036	64	2.88	7.75	10.63	2.15	15.42	17.57
0.55	1035	67	2.89	8.16	11.06	2.14	16.14	18.28
0.60	1035	67	2.86	8.19	11.05	2.14	16.14	18.28
0.65	1035	67	2.86	8.11	10.98	2.14	16.14	18.28
0.70	1035	67	2.88	8.12	11.00	2.14	16.14	18.28
0.75	1035	67	2.87	8.12	10.99	2.14	16.14	18.28
0.80	1035	67	2.87	8.13	11.00	2.14	16.14	18.28
0.85	1035	67	2.88	8.12	11.00	2.14	16.14	18.28
0.90	1035	67	2.90	8.13	11.03	2.14	16.14	18.28
0.95	1034	79	2.90	9.62	12.52	2.14	19.01	21.15
1.00	1034	79	2.93	9.57	12.50	2.14	19.01	21.15

Table A.20 Example of structural strategy for reduction the density of R for CIMGS with static pattern (BELLMEDT)

Table A.21 shows the test results of CIMGS with dynamic pattern of $\epsilon = 0.02$.

CIMGS with dynamic pattern: $\epsilon = 0.02$

name	nz(R)	iter	T(CIMGS)	T(CGLS)	T(total)	NE	CIMGS	CGLS	Total
amoco1	19366	174	2.85	16.86	19.71	0.93	6.78	39.30	47.01
belladit	2227	8	0.08	0.06	0.14	0.01	0.17	0.14	0.31
bellmedt	64187	18	38.72	4.25	42.97	1.77	84.01	9.09	94.87
illc1850	6655	87	1.30	3.37	4.67	0.05	0.86	6.29	7.20
well1850	6972	37	1.43	1.47	2.90	0.05	1.14	2.74	3.93
well1033	2984	38	0.26	0.74	1.00	0.03	0.24	1.39	1.66
dunes8	7417	13	1.78	0.96	2.73	0.10	1.38	2.09	3.57
conev8	3657	1	0.54	0.07	0.61	0.07	0.27	0.13	0.47
strat8	11207	99	7.91	19.38	27.29	0.32	0.74	43.08	44.15
bnl1	15326	103	2.98	4.77	7.75	0.36	5.66	11.04	17.06
ship12s	3886	26	1.38	0.89	2.26	0.02	0.07	1.69	1.78
gen	3818	32	0.85	0.66	1.51	0.01	0.07	1.21	1.29
fs_760_1	853	7	0.94	0.14	1.08	0.06	0.06	0.25	0.36
fs_760_3	8037	> 1520	3.55	45.62	49.17	0.06	4.51	94.38	98.95
gre_115	1364	13	0.09	0.05	0.14	0.00	0.13	0.11	0.24
hwatt_2	21222	1	8.25	0.13	8.38	0.08	4.50	0.21	4.79
mc_fe	11587	709	2.37	46.55	48.91	0.97	3.14	106.37	110.48
orsreg_1	5672	507	7.89	28.22	36.11	0.08	0.35	49.14	49.56
pde_9511	9645	17	2.15	0.57	2.72	0.02	1.06	1.13	2.22
pores_2	5566	> 2448	2.82	89.47	92.29	0.07	0.64	172.57	173.27
saylr4	12205	2277	21.15	213.97	235.12	0.12	1.92	379.38	381.41
steam2	1178	28	0.63	0.87	1.50	0.31	0.10	1.84	2.25
nzfri	13100	46	5.31	2.76	8.06	0.06	10.03	6.16	16.25
pilot4	3569	22	0.56	0.43	0.99	0.06	0.53	0.91	1.50
scsd6	2369	11	0.06	0.19	0.25	0.02	0.06	0.43	0.52
seba	13278	73	4.27	2.48	6.74	0.62	8.94	5.79	15.35
shell	4004	16	1.38	0.37	1.74	0.01	1.37	0.65	2.03
standata	3206	76	0.56	1.13	1.70	0.01	0.65	2.35	3.00
woodw	10074	27	3.42	2.83	6.26	0.18	2.27	6.26	8.71

Table A.21 CIMGS with dynamic pattern of $\epsilon = 0.02$

Table A.22 shows the test results of CIMGS using the static pattern of normal equations and column weight ordering.

CIMGS with static pattern:normal equations

name	nz(R)	iter	T(CIMGS)	T(CGLS)	T(total)	CIMGS	CGLS	Total
amoco1	27686	88	2.34	9.66	12.00	6.14	22.87	29.01
belladit	2395	12	0.06	0.09	0.15	0.15	0.21	0.36
bellmedt	372255	15	122.05	11.97	134.02	303.82	26.71	330.53
illc1850	5633	197	0.56	7.34	7.90	0.09	13.39	13.48
well1850	5633	159	0.56	5.91	6.47	0.09	10.81	10.90
well1033	2469	53	0.10	0.97	1.07	0.03	1.83	1.86
dunes8	6998	19	1.39	1.43	2.82	1.16	3.00	4.16
conev8	5135	1	0.90	0.08	0.97	1.26	0.14	1.40
strat8	21757	296	12.25	69.61	81.86	8.85	140.96	149.82
bnl1	28005	200	3.31	13.07	16.38	7.79	31.55	39.34
ship12s	6388	30	0.96	1.15	2.11	0.12	2.26	2.38
gen	5816	32	0.71	0.76	1.46	0.23	1.47	1.71
fs_760_1	13957	5	3.56	0.23	3.79	6.49	0.47	6.96
fs_760_3	13957	> 1520	3.61	59.46	63.07	6.55	130.39	136.94
gre_115	692	27	0.02	0.08	0.10	0.02	0.15	0.16
hwatt_2	27445	1	18.34	0.15	18.49	23.71	0.25	23.96
mc_fe	73254	26	7.20	4.24	11.44	18.12	10.51	28.62
orsreg_1	24203	443	30.65	37.97	68.62	38.27	75.82	114.09
pde_9511	6420	31	2.46	0.89	3.35	2.16	1.64	3.79
pores_2	12723	> 2448	3.12	123.72	126.84	2.92	242.66	245.58
steam2	20237	6	3.41	0.40	3.81	7.55	0.91	8.47
nzfri	8406	46	0.55	2.44	2.99	0.31	5.28	5.60
pilot4	6899	30	0.42	0.74	1.16	0.56	1.64	2.20
scsd6	2248	22	0.05	0.38	0.43	0.06	0.84	0.90
shell	2748	24	0.26	0.51	0.77	0.03	0.84	0.87
woodw	21525	37	5.16	4.72	9.88	8.37	10.26	18.63

Table A.22 CIMGS with static pattern of normal equations and column weight ordering

Table A.23 shows the test results of CIMGS using the static pattern of normal equations and nested dissection ordering.

CIMGS with static pattern:normal equations

name	nz(R)	#iter	T(CIMGS)	T(CGLS)	T(total)	CIMGS	CGLS	Total
amoco1	27686	111	1.60	11.94	13.54	4.10	28.82	32.92
belladit	2395	12	0.06	0.09	0.15	0.14	0.21	0.35
bellmedt	372255	15	124.25	12.05	136.30	313.71	26.71	340.43
illc1850	5633	224	0.61	8.19	8.80	0.15	15.22	15.37
well1850	5633	177	0.61	6.44	7.05	0.16	12.03	12.19
well1033	2469	74	0.10	1.34	1.44	0.03	2.55	2.58
dunes8	6998	20	0.89	1.47	2.36	0.33	3.15	3.49
conev8	5135	1	0.42	0.07	0.50	0.25	0.14	0.39
strat8	21757	286	6.76	61.38	68.14	1.33	136.21	137.53
bnl1	28005	236	3.30	15.24	18.54	7.75	37.22	44.97
ship12s	6388	34	1.06	1.27	2.32	0.10	2.55	2.66
gen	5816	60	0.66	1.42	2.08	0.15	2.75	2.89
fs_760_1	13957	5	1.43	0.23	1.66	1.64	0.47	2.11
fs_760_3	13957	> 1520	1.42	60.19	61.61	1.65	130.39	132.04
gre_115	692	31	0.02	0.09	0.11	0.02	0.17	0.19
hwatt_2	27445	1	6.61	0.15	6.76	4.71	0.25	4.96
mc_fe	73254	32	5.36	5.11	10.47	13.07	12.89	25.95
orsreg_1	24203	4190	8.42	373.82	382.24	3.87	716.42	720.29
pde_9511	6420	85	1.23	2.44	3.67	0.24	4.45	4.68
pores_2	12723	> 2448	2.24	121.62	123.86	1.15	242.66	243.82
saylr4	38793	> 7128	21.07	1024.45	1045.53	7.20	1945.59	1952.79
steam2	20237	7	1.76	0.45	2.21	3.40	1.05	4.46
nzfri	8406	51	0.73	2.71	3.43	0.66	5.85	6.52
pilot4	6899	47	0.53	1.15	1.68	0.83	2.55	3.39
scsd6	2248	23	0.05	0.39	0.44	0.07	0.88	0.95
seba	52432	30	4.17	2.80	6.97	11.16	7.18	18.34
shell	2748	27	0.39	0.57	0.96	0.23	0.94	1.18
woodw	21525	43	4.57	5.27	9.83	7.13	11.90	19.03

Table A.23 CIMGS with static pattern of normal equations and nested dissection ordering

Table A.24 shows the test results of CIMGS using the static pattern of normal equations and reverse Cuthill-McKee ordering.

CIMGS with static pattern:normal equations

name	nz(R)	iter	T(CIMGS)	T(CGLS)	T(total)	CIMGS	CGLS	Total
amoco1	27686	107	1.84	11.55	13.39	4.71	27.78	32.50
belladit	2395	12	0.06	0.09	0.15	0.14	0.21	0.35
bellmedt	372255	15	125.04	12.05	137.08	309.70	26.71	336.41
illc1850	5633	190	0.56	7.06	7.63	0.10	12.91	13.01
well1850	5633	160	0.56	5.95	6.51	0.10	10.88	10.97
well1033	2469	53	0.10	0.97	1.07	0.03	1.83	1.86
dunes8	6998	13	1.00	0.97	1.97	0.44	2.07	2.51
conev8	5135	1	0.51	0.08	0.58	0.44	0.14	0.58
strat8	21757	172	7.35	36.48	43.83	1.66	81.99	83.65
bnl1	28005	189	3.41	12.25	15.66	7.73	29.82	37.55
ship12s	6388	26	0.96	0.98	1.94	0.08	1.96	2.04
gen	5816	50	0.75	1.19	1.93	0.23	2.29	2.52
fs_760_1	13957	6	1.23	0.26	1.49	1.11	0.56	1.67
fs_760_3	13957	> 1520	1.23	59.05	60.28	1.11	130.39	131.50
gre_115	692	26	0.02	0.08	0.10	0.02	0.14	0.16
hwatt_2	27445	1	8.04	0.15	8.19	7.37	0.25	7.62
mc_fe	73254	24	6.01	3.87	9.88	14.81	9.71	24.52
orsreg_1	24203	360	10.00	31.32	41.32	6.52	61.63	68.14
pde_9511	6420	39	1.43	1.13	2.56	0.48	2.05	2.54
pores_2	12723	> 2448	2.71	120.64	123.35	1.83	242.66	244.49
saylr4	38793	> 7128	25.13	1014.82	1039.94	12.97	1945.59	1958.56
steam2	20237	7	1.88	0.45	2.33	3.79	1.05	4.85
nzfri	8406	44	0.92	2.35	3.27	0.99	5.06	6.05
pilot4	6899	30	0.53	0.73	1.27	0.81	1.64	2.45
scsd6	2248	22	0.05	0.38	0.43	0.07	0.84	0.91
shell	2748	24	0.58	0.51	1.09	0.41	0.84	1.25
woodw	21525	36	4.93	4.47	9.40	8.05	9.98	18.03

Table A.24 CIMGS with static pattern of normal equations and reverse Cuthill McKee ordering

Table A.25 shows the test results of CIMGS using the static pattern of normal equations and minimum degree ordering.

CIMGS with static pattern:normal equations

name	iter	T(CIMGS)	T(CGLS)	T(total)	NE	CIMGS	CGLS	Total
amoco1	66	1.29	7.18	8.47	0.93	3.18	17.19	21.29
belladit	12	0.06	0.09	0.15	0.01	0.14	0.21	0.35
bellmedt	15	119.51	12.22	131.73	1.77	307.15	26.71	335.63
illc1850	232	0.53	8.61	9.14	0.05	0.08	15.76	15.89
well1850	161	0.53	6.00	6.53	0.05	0.08	10.95	11.07
well1033	56	0.10	1.03	1.13	0.03	0.03	1.93	1.98
dunes8	19	0.82	1.44	2.26	0.10	0.25	3.00	3.35
conev8	1	0.38	0.08	0.45	0.07	0.23	0.14	0.44
strat8	278	6.46	63.80	70.25	0.32	1.05	132.40	133.77
bnl1	191	2.89	12.37	15.25	0.36	6.78	30.13	37.27
ship12s	35	0.94	1.32	2.26	0.02	0.08	2.63	2.73
gen	35	0.63	0.83	1.46	0.01	0.10	1.61	1.72
fs_760_1	4	1.14	0.19	1.34	0.06	1.01	0.38	1.45
fs_760_3	>1520	1.15	60.59	61.74	0.06	1.01	130.39	131.46
gre_115	29	0.02	0.08	0.11	0.00	0.02	0.16	0.18
hwatt_2	1	6.88	0.16	7.03	0.08	4.89	0.25	5.22
mc_fe	18	4.32	2.95	7.27	0.97	10.09	7.33	18.40
orsreg_1	4185	8.64	384.74	393.38	0.08	4.33	715.56	719.97
pde_9511	89	1.20	2.63	3.83	0.02	0.23	4.66	4.91
pores_2	>2448	2.00	124.00	126.00	0.07	0.83	242.66	243.56
saylr4	>7128	20.57	1070.60	1091.17	0.12	7.04	1945.59	1952.75
steam2	7	1.59	0.45	2.04	0.31	2.93	1.05	4.29
nzfri	45	0.61	2.39	3.00	0.06	0.41	5.17	5.64
pilot4	33	0.37	0.81	1.18	0.06	0.50	1.80	2.36
scsd6	21	0.04	0.36	0.40	0.02	0.05	0.80	0.87
shell	24	0.25	0.51	0.76	0.01	0.03	0.84	0.88
woodw	40	2.04	5.07	7.11	0.18	1.47	11.08	12.73

Table A.25 CIMGS with static pattern of normal equations and minimum degree ordering

Table A.26 shows the test results of symbolic factorization of CIMGS without ordering.

name	nz(R)	nz(S)	nz(SN)	flops1	flops2	x	y	z
amoco1	27686	53291	53127	8.77	8.77	1.92	0.31	0.01
belladit	2395	3401	3395	0.17	0.17	1.42	0.18	0.01
bellmedt	372255	533070	533070	324.30	324.30	1.43	0.00	0.00
illc1850	5633	35202	35202	0.47	0.47	6.25	0.00	0.00
well1850	5633	35202	35202	0.47	0.47	6.25	0.00	0.00
well1033	2469	8355	8355	0.12	0.12	3.38	0.00	0.00
dunes8	6998	119523	93035	1.89	1.59	13.29	22.16	15.71
conev8	5135	20591	20591	0.38	0.38	4.01	0.00	0.00
strat8	21757	95589	95589	1.61	1.61	4.39	0.00	0.00
bnl1	28005	88526	87688	9.54	9.52	3.13	0.95	0.19
ship12s	6388	7507	7273	0.08	0.08	1.14	3.12	0.62
gen	5816	12612	12557	0.23	0.23	2.16	0.44	0.05
fs_760_1	13957	232165	232165	8.70	8.70	16.63	0.00	0.00
fs_760_3	13957	232165	232165	8.70	8.70	16.63	0.00	0.00
gre_115	692	5749	4414	0.07	0.06	6.38	23.22	14.03
hwatt_2	27445	231168	225839	5.86	5.80	8.23	2.31	1.10
mc_fe	73254	91277	91202	12.60	12.60	1.25	0.08	0.00
orsreg_1	24203	1389670	1389670	30.94	30.94	57.42	0.00	0.00
pde_9511	6420	57749	57749	0.69	0.69	9.00	0.00	0.00
pores_2	12723	287512	287512	5.46	5.46	22.60	0.00	0.00
saylr4	38793	1302805	1296344	27.26	27.16	33.42	0.50	0.36
steam2	20237	113548	113548	7.32	7.32	5.61	0.00	0.00
fffff800	10625	98097	96178	4.74	4.69	9.05	1.96	0.85
nzfri	8406	189907	23368	7.52	3.60	2.78	87.70	52.11
pilot4	6899	30482	28778	0.97	0.97	4.17	5.59	0.37
scsd6	2248	11026	11017	0.32	0.32	4.90	0.08	0.00
seba	52432	133386	100104	24.92	21.00	1.91	24.95	15.70
shell	2748	95739	91515	0.87	0.84	33.30	4.41	2.84
standata	1833	35464	5422	0.29	0.11	2.96	84.71	62.14
woodw	21525	156426	153974	6.61	6.56	7.15	1.57	0.69

flops1: number of floating point operations using set S.
flops2: number of floating point operations using set SN.
x : $\frac{nz(SN)}{nz(R)}$. y : $\frac{nz(S)-nz(N)}{nz(S)}(\%)$. z : $\frac{flops1-flops2}{flops1}(\%)$.

Table A.26 Statistic information of symbolic analysis of CIMGS without ordering

Table A.27 shows the test results of symbolic factorization of CIMGS with column weight ordering.

name	nz(R)	nz(S)	nz(SN)	flops1	flops2	x	y	z
amoco1	27686	45509	45509	6.14	6.14	1.64	0.00	0.00
belladit	2395	3251	3251	0.15	0.15	1.36	0.00	0.00
bellmedt	372255	524164	524164	303.82	303.82	1.41	0.00	0.00
illc1850	5633	8660	8660	0.09	0.09	1.54	0.00	0.00
well1850	5633	8660	8660	0.09	0.09	1.54	0.00	0.00
well1033	2469	3004	3004	0.03	0.03	1.22	0.00	0.00
dunes8	6998	70148	47773	1.16	0.86	6.83	31.90	26.11
conev8	5135	55345	45897	1.26	1.12	8.94	17.07	10.52
strat8	21757	502688	327413	8.85	6.64	15.05	34.87	24.98
bnl1	28005	73855	73774	7.79	7.79	2.63	0.11	0.00
ship12s	6388	11410	6795	0.12	0.07	1.06	40.45	38.80
gen	5816	15529	12922	0.23	0.20	2.22	16.79	14.96
fs_760_1	13957	167498	164678	6.96	6.92	11.80	1.68	0.58
fs_760_3	13957	167498	164678	6.96	6.92	11.80	1.68	0.58
gre_115	692	1816	1786	0.02	0.02	2.58	1.65	1.01
hwatt_2	27445	975681	750668	23.71	19.02	27.35	23.06	19.78
mc_fe	73254	115457	115457	18.12	18.12	1.58	0.00	0.00
orsreg_1	24203	1689469	1689063	38.27	38.27	69.79	0.02	0.02
pde_9511	6420	169997	124897	2.16	1.60	19.45	26.53	25.79
pores_2	12723	130986	127625	2.92	2.86	10.03	2.57	1.79
steam2	20237	118870	118782	7.90	7.90	5.87	0.07	0.04
fffff800	10625	53069	52008	2.11	2.09	4.89	2.00	1.05
nzfri	8406	17974	17696	0.31	0.31	2.11	1.55	0.42
pilot4	6899	17511	10745	0.56	0.29	1.56	38.64	47.73
scsd6	2248	3212	3212	0.06	0.06	1.43	0.00	0.00
seba	52432	70993	70993	13.65	13.65	1.35	0.00	0.00
shell	2748	4534	4474	0.03	0.03	1.63	1.32	0.49
standata	1833	3648	3552	0.03	0.03	1.94	2.63	1.41
woodw	21525	201978	201875	8.37	8.37	9.38	0.05	0.01

flops1: number of floating point operations using set S.
flops2: number of floating point operations using set SN.
x : $\frac{nz(SN)}{nz(R)}$. y : $\frac{nz(S)-nz(N)}{nz(S)}(\%)$. z : $\frac{flops1-flops2}{flops1}(\%)$.

Table A.27 Statistic information of symbolic analysis of CIMGS with column weight ordering

Table A.28 shows the test results of symbolic factorization of CIMGS with nested dissection ordering.

name	nz(R)	nz(S)	nz(SN)	flops1	flops2	x	y	z
amoco1	27686	35379	35379	4.10	4.10	1.28	0.00	0.00
belladit	2395	3085	3085	0.14	0.14	1.29	0.00	0.00
bellmedt	372255	521722	521721	313.71	313.71	1.40	0.00	0.00
illc1850	5633	12626	12626	0.16	0.16	2.24	0.00	0.00
well1850	5633	12626	12626	0.16	0.16	2.24	0.00	0.00
well1033	2469	3098	3098	0.03	0.03	1.25	0.00	0.00
dunes8	6998	25219	19721	0.33	0.29	2.82	21.80	13.04
conev8	5135	15967	12699	0.25	0.22	2.47	20.47	13.42
strat8	21757	94600	73077	1.33	1.12	3.36	22.75	15.31
bnl1	28005	72664	71470	7.75	7.73	2.55	1.64	0.22
ship12s	6388	8601	7417	0.10	0.10	1.16	13.77	7.37
gen	5816	11050	10701	0.15	0.15	1.84	3.16	1.43
fs_760_1	13957	53924	52021	1.68	1.66	3.73	3.53	1.60
fs_760_3	13957	53924	52021	1.68	1.66	3.73	3.53	1.60
gre_115	692	2135	2028	0.02	0.02	2.93	5.01	1.61
hwatt_2	27445	191646	155714	4.71	4.28	5.67	18.75	9.10
mc_fe	73254	96040	93412	13.07	13.02	1.28	2.74	0.34
orsreg_1	24203	246362	204718	3.87	3.46	8.46	16.90	10.67
pde_9511	6420	27011	20463	0.24	0.20	3.19	24.24	15.58
pores_2	12723	61681	57264	1.15	1.11	4.50	7.16	3.97
saylr4	38793	468929	382712	7.20	6.35	9.87	18.39	11.75
steam2	20237	68220	62556	3.73	3.59	3.09	8.30	3.64
fffff800	10625	69768	57709	2.63	2.40	5.43	17.28	8.87
nzfri	8406	31534	20359	0.66	0.65	2.42	35.44	1.68
pilot4	6899	23777	21655	0.83	0.80	3.14	8.92	3.65
scsd6	2248	3247	3247	0.07	0.07	1.44	0.00	0.00
seba	52432	60514	60514	11.16	11.16	1.15	0.00	0.00
shell	2748	23267	20982	0.23	0.22	7.64	9.82	4.86
standata	1833	7141	6169	0.06	0.06	3.37	13.61	8.79
woodw	21525	176419	159040	7.13	6.88	7.39	9.85	3.46

flops1: number of floating point operations using set S.
flops2: number of floating point operations using set SN.
x : $\frac{nz(SN)}{nz(R)}$. y : $\frac{nz(S)-nz(N)}{nz(S)}(\%)$. z : $\frac{flops1-flops2}{flops1}(\%)$.

Table A.28 Statistic information of symbolic analysis of CIMGS with nested dissection ordering

Table A.29 shows the test results of symbolic factorization of CIMGS with reverse Cuthill-McKee ordering.

name	nz(R)	nz(S)	nz(SN)	flops1	flops2	x	y	z
amoco1	27686	38041	38041	4.71	4.71	1.37	0.00	0.00
belladit	2395	3161	3161	0.14	0.14	1.32	0.00	0.00
bellmedt	372255	529745	529745	309.70	309.70	1.42	0.00	0.00
illc1850	5633	8799	8799	0.10	0.10	1.56	0.00	0.00
well1850	5633	8799	8799	0.10	0.10	1.56	0.00	0.00
well1033	2469	3025	3025	0.03	0.03	1.23	0.00	0.00
dunes8	6998	28577	28577	0.44	0.44	4.08	0.00	0.00
conev8	5135	23284	23284	0.44	0.44	4.53	0.00	0.00
strat8	21757	98403	98403	1.66	1.66	4.52	0.00	0.00
bnl1	28005	90676	90676	7.73	7.73	3.24	0.00	0.00
ship12s	6388	7399	7399	0.08	0.08	1.16	0.00	0.00
gen	5816	16563	16563	0.23	0.23	2.85	0.00	0.00
fs_760_1	13957	38622	38622	1.14	1.14	2.77	0.00	0.00
fs_760_3	13957	38622	38622	1.14	1.14	2.77	0.00	0.00
gre_115	692	2260	2260	0.02	0.02	3.27	0.00	0.00
hwatt_2	27445	245233	245233	7.37	7.37	8.94	0.00	0.00
mc_fe	73254	98277	98277	14.81	14.81	1.34	0.00	0.00
orsreg_1	24203	317744	317744	6.52	6.52	13.13	0.00	0.00
pde_9511	6420	41618	41618	0.48	0.48	6.48	0.00	0.00
pores_2	12723	81833	81833	1.83	1.83	6.43	0.00	0.00
saylr4	38793	636653	636653	12.97	12.97	16.41	0.00	0.00
steam2	20237	64444	64444	3.80	3.80	3.18	0.00	0.00
fffff800	10625	62598	62598	2.40	2.40	5.89	0.00	0.00
nzfri	8406	50800	50800	0.99	0.99	6.04	0.00	0.00
pilot4	6899	23830	23830	0.81	0.81	3.45	0.00	0.00
scsd6	2248	3404	3404	0.07	0.07	1.51	0.00	0.00
seba	52432	70631	70631	13.76	13.76	1.35	0.00	0.00
shell	2748	47272	47272	0.41	0.41	17.20	0.00	0.00
standata	1833	6136	6136	0.06	0.06	3.35	0.00	0.00
woodw	21525	185401	185401	8.05	8.05	8.61	0.00	0.00

flops1: number of floating point operations using set S.
flops2: number of floating point operations using set SN.
x : $\frac{nz(SN)}{nz(R)}$. y : $\frac{nz(S)-nz(N)}{nz(S)}(\%)$. z : $\frac{flops1-flops2}{flops1}(\%)$.

Table A.29 Statistic information of symbolic analysis of CIMGS with reverse Cuthill McKee ordering

Table A.30 shows the test results of symbolic factorization of CIMGS with minimum degree ordering.

name	nz(R)	nz(S)	nz(SN)	flops1	flops2	x	y	z
amoco1	27686	30678	30623	3.18	3.18	1.11	0.18	0.02
belladit	2395	3055	3054	0.14	0.14	1.28	0.03	0.00
bellmedt	372255	516381	516381	307.15	307.15	1.39	0.00	0.00
illc1850	5633	7579	7579	0.08	0.08	1.35	0.00	0.00
well1850	5633	7579	7579	0.08	0.08	1.35	0.00	0.00
well1033	2469	2896	2887	0.03	0.03	1.17	0.31	0.13
dunes8	6998	19568	17471	0.25	0.24	2.50	10.72	4.46
conev8	5135	15114	13743	0.23	0.22	2.68	9.07	5.57
strat8	21757	78034	70529	1.05	0.99	3.24	9.62	5.58
bnl1	28005	59060	58948	6.78	6.78	2.10	0.19	0.01
ship12s	6388	7507	7295	0.08	0.08	1.14	2.82	0.59
gen	5816	8454	8252	0.10	0.10	1.42	2.39	4.45
fs_760_1	13957	36124	34605	1.03	1.03	2.48	4.20	0.42
fs_760_3	13957	36124	34605	1.03	1.03	2.48	4.20	0.42
gre_115	692	1762	1665	0.02	0.02	2.41	5.51	2.35
hwatt_2	27445	207623	163940	4.89	4.50	5.97	21.04	8.02
mc_fe	73254	78833	78833	10.09	10.09	1.08	0.00	0.00
orsreg_1	24203	277416	246554	4.33	4.05	10.19	11.12	6.61
pde_9511	6420	26616	21111	0.23	0.20	3.29	20.68	13.46
pores_2	12723	46587	46047	0.83	0.82	3.62	1.16	0.55
saylr4	38793	459936	406695	7.04	6.57	10.48	11.58	6.70
steam2	20237	60448	58096	3.14	3.12	2.87	3.89	0.55
fffff800	10625	18487	16295	0.56	0.47	1.53	11.86	15.43
nzfri	8406	18632	18583	0.41	0.41	2.21	0.26	0.07
pilot4	6899	14541	13838	0.50	0.48	2.01	4.83	3.80
scsd6	2248	2693	2673	0.05	0.05	1.19	0.74	0.08
seba	52432	54779	54779	10.66	10.66	1.04	0.00	0.00
shell	2748	4391	4318	0.03	0.03	1.57	1.66	1.63
standata	1833	3128	2800	0.02	0.02	1.53	10.49	7.59
woodw	21525	46797	45168	1.47	1.43	2.10	3.48	2.71

flops1: number of floating point operations using set S.
flops2: number of floating point operations using set SN.
x : $\frac{nz(SN)}{nz(R)}$. y : $\frac{nz(S)-nz(N)}{nz(S)}(\%)$. z : $\frac{flops1-flops2}{flops1}(\%)$.

Table A.30 Statistic information of symbolic analysis of CIMGS with minimum degree ordering

Table A.27 shows the test results of incomplete Cholesky factorization using the pattern of the normal equations. The blank rows are the problems that IC does not complete successfully.

name	iter	T(IC(0))	T(CGLS)	T(total)	NE	IC(0)	CGLS	Total
amoco1								
belladit	15	0.09	0.11	0.20	0.01	0.08	0.26	0.34
bellmedt	17	109.30	13.42	122.72	1.77	142.32	30.16	174.25
illc1850								
well1850								
well1033								
dunes8	21	0.51	1.50	2.01	0.10	0.06	3.31	3.47
conev8	1	0.28	0.07	0.35	0.07	0.05	0.14	0.26
strat8	91	3.08	19.16	22.24	0.32	0.18	43.48	43.98
bnl1								
ship12s								
gen								
fs_760_1	4	0.62	0.19	0.81	0.06	0.22	0.38	0.66
fs_760_3								
gre_115	35	0.01	0.10	0.11	0.00	0.00	0.19	0.19
hwatt_2								
mc_fe								
orsreg_1								
pde_9511								
pores_2								
saylr4								
steam2	7	0.97	0.45	1.42	0.31	0.53	1.05	1.89
fffff800								
nzfri	57	1.38	3.00	4.39	0.06	0.06	6.54	6.65
pilot4								
scsd6	21	0.12	0.35	0.46	0.02	0.03	0.80	0.85
seba								
shell	22	0.32	0.45	0.77	0.01	0.01	0.77	0.79
standata								
woodw	35	3.89	4.27	8.16	0.18	0.29	9.71	10.18

Table A.31 Performance of IC(0) with static pattern of normal equations

Table A.32 shows the number of non-zero elements in R using the modified pattern produced by MPADD, MPDROP and MPBBD.

name	IC(0)	MPADD	MPDROP	MPBBD
amoco1	27686	53258	9775	47468
belladit	2395	3398	743	3256
bellmedt	372255	533070	15829	529055
illc1850	5633	72562	3685	52349
well1850	5633	72562	3685	52349
well1033	2469	9077	1803	7039
dunes8	6998	150970	3136	22302
conev8	5135	20591	1692	12587
strat8	21757	95589	7326	76632
bnl1	28005	87941	5549	99938
ship12s	6388	7399	4729	8804
gen	5816	16807	4429	64851
fs_760_1	13957	235707	5178	52496
fs_760_3	13957	235707	5178	52496
gre_115	692	5596	359	3139
hwatt_2	27445	229152	11712	157169
mc_fe	73254	91252	6823	104890
orsreg_1	24203	1389670	7012	235504
pde_9511	6420	57749	2882	23902
pores_2	12723	294534	4203	98161
saylr4	38793	1309753	10249	49504
steam2	20237	113548	6396	57676
fffff800	10625	97791	5734	58549
nzfri	8406	121273	2565	23303
pilot4	6899	30486	1810	33968
scsd6	2248	11017	587	5945
seba	52432	119477	21698	90403
shell	2748	94911	1388	71755
standata	1833	24579	1041	13148
woodw	21525	164469	8874	181408

Table A.32 Number of non zeros of R

Table A.33 shows the number of iterations needed by CGLS preconditioned by IC with and without pattern modification.

name	IC(0)	MPADD	MPDROP	MPBBD
amoco1		8	> 660	67
belladit	15	4	30	9
bellmedt	17	1	66	11
illc1850		1	714	267
well1850		1	284	210
well1033		1	139	103
dunes8	21	11	53	23
conev8	1		1	1
strat8	91	1	845	259
bnl1		8	1241	142
ship12s		7	158	44
gen		5	83	71
fs_760_1	4	1	44	4
fs_760_3		5	>1520	>1520
gre_115	35	16	65	28
hwatt_2		1	1	1
mc_fe		5	>1530	22
orsreg_1		1		2475
pde_9511		1	442	85
pores_2		3	>2448	>2448
saylr4		7		
steam2	7	2	21	4
fffff800				
nzfri	57		150	
pilot4		12	188	53
scsd6	21	6	43	24
seba		17	196	31
shell	22	10	44	19
standata			255	
woodw	35	5	78	32

Table A.33 Number of iterations of CGLS preconditioned by incomplete Cholesky factorization with different pattern modifications

Table A.34 shows the number of operations in the iterative phase of the CGLS preconditioned by IC with and without pattern modification.

name	IC(0)	MPADD	MPDROP	MPBBD
amoco1		3.06	123.41	22.79
belladit	0.26	0.09	0.31	0.19
bellmedt	30.16	3.54	19.83	27.03
illc1850		0.50	42.88	68.12
well1850		0.50	17.07	53.60
well1033		0.09	4.40	5.43
dunes8	3.31	8.39	7.42	5.05
conev8	0.14		0.12	0.19
strat8	43.48	1.12	353.22	180.33
bnl1		3.37	83.87	63.42
ship12s		0.58	10.70	3.72
gen		0.49	3.33	20.13
fs_760_1	0.38	1.46	2.25	1.08
fs_760_3		5.35	77.00	364.78
gre_115	0.19	0.41	0.26	0.43
hwatt_2		1.46	0.16	1.03
mc_fe		2.57	200.29	11.77
orsreg_1		8.44		2515.52
pde_9511		0.38	16.78	10.43
pores_2		4.29	159.22	1079.44
saylr4		40.17		
steam2	1.05	1.28	1.83	1.31
fffff800				
nzfri	6.54		13.60	
pilot4		1.85	6.30	8.67
scsd6	0.80	0.47	1.34	1.28
seba		8.81	22.09	12.20
shell	0.77	4.23	1.29	6.05
standata			5.63	
woodw	9.71	4.63	17.52	29.67

Table A.34 Number of flops of CGLS preconditioned by incomplete Cholesky factorization with different pattern modifications

Table A.35 shows the number of operations in the computation of the preconditioner R via IC using the pattern with and without modifications.

name	IC(0)	MPADD	MPDROP	MPBBD
amoco1		11.32	0.44	8.74
belladit	0.08	0.19	0.01	0.17
bellmedt	142.32	366.25	0.50	360.36
illc1850		14.58	0.01	4.91
well1850		14.58	0.01	4.91
well1033		0.59	0.01	0.18
dunes8	0.06	40.35	0.01	0.62
conev8	0.05		0.01	0.29
strat8	0.18	4.18	0.03	2.20
bnl1		19.41	0.13	19.81
ship12s		0.09	0.02	0.11
gen		1.15	0.02	5.79
fs_760_1	0.22	101.95	0.03	3.64
fs_760_3		101.95	0.03	3.64
gre_115	0.00	0.33	0.00	0.09
hwatt_2		29.03	0.72	11.39
mc_fe		13.34	0.08	17.85
orsreg_1		1056.18		19.53
pde_9511		3.58	0.01	0.42
pores_2		89.78	0.02	7.44
saylr4		505.89		
steam2	0.53	26.50	0.08	5.62
fffff800				
nzfri	0.06		0.01	
pilot4		3.02	0.01	3.68
scsd6	0.03	1.09	0.00	0.28
seba		36.35	2.43	20.83
shell	0.01	20.83	0.00	11.87
standata			0.00	
woodw	0.29	35.03	0.07	39.83

Table A.35 Number of flops of incomplete Cholesky factorization with different pattern modifications

Table A.36 shows the total number of operations in solving the linear least squares problems using CGLS preconditioned by IC with and without pattern modification.

name	IC(0)	MPADD	MPDROP	MPBBD
amoco1		15.31	124.26	32.46
belladit	0.34	0.29	0.32	0.37
bellmedt	174.25	371.56	20.44	389.16
illc1850		15.13	42.92	73.08
well1850		15.13	17.11	58.56
well1033		0.71	4.42	5.64
dunes8	3.47	48.83	7.47	5.78
conev8	0.26		0.15	0.55
strat8	43.98	5.63	353.32	182.84
bnl1		23.14	84.06	83.60
ship12s		0.69	10.73	3.85
gen		1.65	3.36	25.94
fs_760_1	0.66	103.46	2.31	4.77
fs_760_3		107.35	77.05	368.48
gre_115	0.19	0.74	0.27	0.53
hwatt_2		30.57	0.89	12.49
mc_fe		16.89	200.50	30.59
orsreg_1		1064.70		2535.12
pde_9511		3.98	16.79	10.87
pores_2		94.13	159.25	1086.95
saylr4		546.18		53
steam2	1.89	28.09	2.02	7.23
fffff800			51	51
nzfri	6.65		13.65	51
pilot4		4.93	6.34	12.41
scsd6	0.85	1.58	1.36	1.58
seba		45.78	24.88	33.64
shell	0.79	25.07	1.29	17.93
standata			5.64	51
woodw	10.18	39.84	17.67	69.68

Table A.36 Number of total flops of solve the least squares problem using CGLS preconditioned by incomplete Cholesky factorization with different pattern modifications

Table A.37 shows the number of non-zero elements in R of IC with and without pattern modification and orderings.

		MPADD				MPDROP			
name	IC(0)	none	nestd	rcm	mind	none	nestd	rcm	mind
amoco1	27686	53258	35379	38062	30677	9775	7530	5319	12583
belladit	2395	3398	3085	3161	3054	743	766	698	700
bellmedt	372255	533070	521721	529767	516445	15829	13645	31369	11985
illc1850	5633	72562	14233	12633	8124	3685	3073	2762	3756
well1850	5633	72562	14233	12633	8124	3685	3073	2762	3756
well1033	2469	9077	3130	3107	2895	1803	1859	2081	2001
dunes8	6998	150970	27956	29043	19994	3136	3345	2845	4151
conev8	5135	205011	17160	23315	15682	1692	2380	2268	2982
strat8	21757	95589	114319	98433	87465	7326	9789	7433	11602
bnl1	28005	87941	74550	93905	59975	5549	6480	2543	5767
ship12s	6388	7399	8503	7462	7417	4729	4912	4449	4621
gen	5816	16807	12123	22603	10287	4429	4537	2786	4426
fs_760_1	13957	235707	53972	38624	35826	5178	4316	2457	5775
fs_760_3	13957	235707	53972	38624	35826	5178	4316	2457	5775
gre_115	692	5596	2273	2479	1784	359	401	328	414
hwatt_2	27445	229152	202326	245233	220521	11712	10700	5336	10300
mc_fe	73254	91252	95894	98277	78833	6823	34789	18120	10044
orsreg_1	24203	1389670	269881	317744	306405	7012	9684	6102	11240
pde_9511	6420	57749	34466	41618	31535	2882	3399	2344	4014
pores_2	12723	294534	63562	83281	47708	4203	7047	3639	7818
saylr4	38793	1309753	534122	639127	511966	10249	15216	10310	17802
steam2	20237	113548	68132	64444	60296	6396	8796	6364	10384
fffff800	10625	97791	71238	64986	18686	5734	6040	3293	i5746
nzfri	8406	121272	22007	57241	19192	2565	4452	4044	4150
pilot4	6899	30486	24852	24234	14703	1810	2208	1749	3221
scsd6	2248	11017	3267	3406	2677	587	970	1192	1301
seba	52432	119477	60536	71163	54779	21698	28049	13600	22613
shell	2748	94911	23586	51124	4943	1388	1689	1218	1791
standata	1833	24579	7022	6450	3311	1041	1419	1259	1577
woodw	21525	164469	175511	198501	48703	8874	10478	8968	12267

Table A.37 Comparison of the number of non zeros of R in different orderings

Table $A.38$ shows the number of operations in the computation of R via IC with and without pattern modification and orderings.

		MPADD				MPDROP			
name	IC(0)	none	nestd	rcm	mind	none	nestd	rcm	mind
amoco1		11.32	4.46	5.24	3.22	0.44	0.20	0.11	0.64
belladit	0.08	0.19	0.15	0.16	0.15	0.01	0.01	0.01	0.01
bellmedt	142.32	366.25	350.96	362.02	343.19	0.50	0.17	3.58	0.09
illc1850		14.58	0.58	0.45	0.12	0.01	0.01	0.01	0.02
well1850		14.58	0.58	0.45	0.12	0.01	0.01	0.01	0.02
well1033		0.59	0.04	0.04	0.03	0.01	0.01	0.01	0.01
dunes8	0.06	40.35	1.26	1.23	0.61	0.01	0.01	0.01	0.02
conev8	0.05		0.74		0.67	0.01	0.01	0.01	0.02
strat8	0.18	4.18	7.67	4.52	4.23	0.03	0.05	0.03	0.05
bnl1		19.41	15.65	21.36	11.16	0.13	0.10	0.01	0.07
ship12s		0.09	0.12	0.09	0.09	0.02	0.03	0.02	0.02
gen		1.15	0.42	1.39	0.22	0.02	0.02	0.01	0.02
fs_760_1	0.22	101.95	4.73	2.13	1.97	0.03	0.02	0.01	0.03
fs_760_3		101.95	4.73		1.97	0.03	0.02	0.01	0.03
gre_115	0.00	0.33	0.06	0.07	0.04	0.00	0.00	0.00	0.00
hwatt_2		29.03	28.65		34.82	0.72	0.17	0.02	0.06
mc_fe		13.34	15.41	16.82	10.20	0.08	2.19	0.57	0.18
orsreg_1		1056.18	46.68	51.31	61.70		0.04	0.02	0.04
pde_9511		3.58	1.70	2.01	1.39	0.01	0.01	0.01	0.02
pores_2		89.78	6.55	10.89	3.47	0.02	0.03	0.01	0.04
saylr4		505.89	116.63	123.92	103.14		0.06	0.04	0.07
steam2	0.53	26.50	9.11	7.60	6.99	0.08	0.12	0.08	0.13
fffff800									
nzfri	0.06			8.27		0.01	0.01	0.01	0.01
pilot4		3.02	2.40	1.93	0.93	0.01	0.02	0.01	0.02
scsd6	0.03	1.09	0.08	0.09	0.05	0.00	0.00	0.01	0.01
seba		36.35	12.44		10.72	2.43	2.87		
shell	0.01	20.83	2.78		0.08	0.00	0.00	0.00	0.01
standata						0.00			
woodw	0.29	35.03	46.66	53.95	3.21	0.07	0.09	0.07	0.09

Table A.38 Number of flops of incomplete Cholesky factorization with different ordering

	IC(0)				MPADD				MPDROP			
name	none	nestd	rcm	mind	none	nestd	rcm	mind	none	nestd	rcm	mind
amoco1					8	1	1	3	> 660	> 660	> 660	> 660
belladit	15	14	13	13	4	1	1	3	30	30	30	31
bellmedt	17	17	17	17	1	3	1	1	66	65	64	65
illc1850		264	225	243		1	1	1	714	483	534	463
well1850			214	192	1	1	1	1	284	294	283	229
well1033				58	1	1	1	3	139	132	115	85
dunes8	21	21	15	20	11	12	1	11	53	53	55	48
conev8	1	1	1	1		1		1	1	1	1	1
strat8	91	351	200	319	1	38	1	54	845	806	850	762
bnl1					8	15	1	5	1241	> 1264	1231	918
ship12s					7	7	1	9	158	171	135	168
gen		62			5	10	1	5	83	84	337	77
fs_760_1	4	5	6	4	1	2	2	1	44	43	47	33
fs_760_3					5	14		75	>1520	> 1520	> 1520	> 1520
gre_115	35	32	27	32	16	5	1	10	65	57	67	59
hwatt_2					1	1		1	1	1	1	1
mc_fe					5	4	3	4	>1530	> 1530	> 1530	> 1530
orsreg_1					1	13	1	18		> 4410	> 4410	> 4410
pde_9511					1	25	1	29	442	358	476	300
pores_2					3	21	3	9	>2448	> 2448	> 2448	> 2448
saylr4					7	113	2	230		> 7128	> 7128	> 7128
steam2	7	7	7	7	2	2	2	3	21	20	20	20
fffff800												
nzfri	57						1		150	140	165	145
pilot4					12	13	1	6	188	213	226	142
scsd6	21	22	22	21	6	1	1	8	43	40	35	36
seba					17	1		1	196	72		
shell	22	26	21	25	10	14		9	44	39	46	37
standata									255			
woodw	35	43	36	39	5	9	1	5	78	67	76	70

Table A.39 Number of iterations of CGLS preconditioned by incomplete Cholesky factorization with different pattern modifications and orderings

Table A.40 Number of operations of CGLS preconditioned by IC with different orderings

	IC(0)				MPADD				MPDROP			
name	none	nestd	rcm	mind	none	nestd	rcm	mind	none	nestd	rcm	mi nd
amoco1					3.06	0.43	0.45	0.94	123.41	117.48	111.64	130.83
belladit	0.26	0.24	0.23	0.23	0.09	0.03	0.03	0.07	0.31	0.31	0.30	0.31
bellmedt	30.16	30.16	30.16	30.16	3.54	8.11	3.52	3.44	19.83	18.96	23.24	18.53
illc1850		17.93	15.29	16.51	0.50	0.15	0.14	0.11	42.88	27.83	30.10	27.95
well1850			14.54	13.05	0.50	0.15	0.14	0.11	17.07	16.95	15.96	13.84
well1033				2.00	0.09	0.05	0.05	0.12	4.40	4.21	3.77	2.76
dunes8	3.31	3.31	2.38	3.15	8.39	2.97	0.35	2.36	7.42	7.46	7.63	6.92
conev8	0.14	0.14	0.14	0.14		0.21		0.21	0.12	0.13	0.12	0.13
strat8	43.48	167.12	95.31	151.90	1.12	32.53	1.14	40.21	353.22	344.88	355.68	331.59
bnl1					3.37	5.32	0.63	1.57	83.87	90.13	68.38	62.85
ship12s					0.58	0.61	0.11	0.74	10.70	11.70	8.99	11.30
gen		2.84			0.49	0.74	0.17	0.34	3.33	3.41	11.25	3.09
fs_760_1	0.38	0.47	0.56	0.38	1.46	0.61	0.46	0.26	2.25	2.05	1.89	1.77
fs_760_3					5.35	3.56		13.08	77.00	71.75	60.45	80.63
gre_115	0.19	0.17	0.15	0.17	0.41	0.06	0.02	0.10	0.26	0.24	0.26	0.25
hwatt_2					1.46	1.30		1.41	0.16	0.15	0.12	0.15
mc_fe					2.57	2.19	1.74	1.88	200.29	371.50	269.45	220.01
orsreg_1					8.44	15.57	2.01	24.04		497.89	434.69	525.34
pde_9511					0.38	4.19	0.29	4.50	16.78	14.33	17.04	12.75
pores_2					4.29	6.50	1.33	2.27	159.22	187.07	153.69	194.62
saylr4					40.17	255.85	6.67	499.17		1273.31	1133.42	1347.05
steam2	1.05	1.05	1.05	1.05	1.28	0.83	0.79	1.05	1.83	1.94	1.75	2.08
fffff800												
nzfri	6.54						0.46		13.60	13.76	15.94	14.07
pilot4					1.85	1.69	0.18	0.55	6.30	7.48	7.52	5.57
scsd6	0.80	0.84	0.84	0.80	0.47	0.06	0.06	0.33	1.34	1.31	1.18	1.23
seba					8.81	0.40		0.36	22.09	9.99		
shell	0.77	0.91	0.74	0.87	4.23	1.70		0.41	1.29	1.19	1.31	1.14
standata									5.63			
woodw	9.71	11.90	9.98	10.81	4.63	8.44	1.45	2.09	17.52	15.50	17.10	16.69

	IC(0)				MPADD				MPDROP			
name	none	nestd	rcm	mind	none	nestd	rcm	mind	none	nestd	rcm	mind
amoco1					15.31	5.82	6.61	5.09	124.26	117.97	111.97	131.92
belladit	0.34	0.33	0.31	0.31	0.29	0.19	0.20	0.22	0.32	0.32	0.31	0.32
bellmedt	174.25	174.25	174.25	174.25	371.56	360.85	367.31	348.40	20.44	19.19	27.02	18.66
illc1850		18.02	15.37	16.59	15.13	0.78	0.64	0.28	42.92	27.87	30.13	27.99
well1850			14.63	13.14	15.13	0.78	0.64	0.28	17.11	16.99	15.99	13.88
well1033				2.04	0.71	0.12	0.12	0.18	4.42	4.23	3.79	2.79
dunes8	3.47	3.47	2.54	3.31	48.83	4.32	1.69	3.07	7.47	7.52	7.68	6.99
conev8	0.26	0.26	0.26	0.26		1.03		0.94	0.15	0.17	0.17	0.18
strat8	43.98	167.62	95.81	152.40	5.63	40.52	5.98	44.76	353.32	345.05	355.78	331.79
bnl1					23.14	21.34	22.35	13.08	84.06	90.31	68.42	62.97
ship12s					0.69	0.75	0.22	0.85	10.73	11.74	9.02	11.33
gen		2.91			1.65	1.17	1.57	0.58	3.36	3.44	11.27	3.12
fs_760_1	0.66	0.74	0.83	0.66	103.46	5.40	2.65	2.28	2.31	2.09	1.90	1.83
fs_760_3					107.35	8.35		15.10	77.05	71.79	60.46	80.68
gre_115	0.19	0.18	0.15	0.18	0.74	0.13	0.09	0.14	0.27	0.24	0.26	0.25
hwatt_2					30.57	30.03		36.31	0.89	0.34	0.15	0.23
mc_fe					16.89	18.57	19.53	13.06	200.50	374.18	270.31	220.34
orsreg_1					1064.70	62.32	53.40	85.82		497.95	434.72	525.41
pde_9511					3.98	5.90	2.32	5.91	16.79	14.35	17.05	12.78
pores_2					94.13	13.12	12.29	5.81	159.25	187.14	153.72	194.69
saylr4					546.18	372.60	130.71	602.42		1273.41	1133.48	1347.16
steam2	1.89	1.89	1.89	1.89	28.09	10.25	8.70	8.35	2.02	2.20	1.93	2.37
fffff800									51			
nzfri	6.65						8.78		13.65	13.79	15.96	14.10
pilot4					4.93	4.15	2.17	1.54	6.34	7.52	7.55	5.61
scsd6	0.85	0.89	0.89	0.40	1.58	0.16	0.17	0.85	1.36	1.32	1.19	1.25
seba					45.78	13.46		11.70	24.88	13.24		
shell	0.79	0.93	0.76	0.90	25.07	4.49		0.49	1.29	1.20	1.32	1.15
standata									5.64			
woodw	10.18	12.37	10.46	11.28	39.84	55.28	55.59	5.47	17.67	15.68	17.25	16.87

Table A.41 Number of total operations of solve the least squares problem using CGLS preconditioned by IC with different pattern modifications and orderings

Table A.42 shows the time of patterm modification and the time of IC factorization without orderings.

	MPADD			MPDROP		
name	nz(R)	T(mpadd)	T(ic)	nz(R)	T(mpdrop)	T(ic)
amoco1	53258	1.82	10.87	9775	4.05	1.62
belladit	3398	0.04	0.16	743	0.08	0.02
bellmedt	533070	53.68	229.42	15829	78.04	2.09
illc1850	72562	1.02	9.13	3685	2.05	0.53
well1850	72562	1.02	9.13	3685	2.05	0.52
well1033	9077	0.11	0.73	1803	0.35	0.20
dunes8	150970	1.33	26.28	3136	2.87	0.27
conev8				1692	0.88	0.11
strat8	95589	1.14	7.60	7326	13.26	1.75
bnl1	87941	1.56	11.73	5549	6.41	0.28
ship12s	7399	0.04	0.51	4729	2.06	0.40
gen	16807	0.16	0.98	4429	1.41	0.27
fs_760_1	235707	4.76	55.24	5178	7.52	0.28
fs_760_3	235707	4.76	55.28	5178	7.51	0.28
gre_115	5596	0.05	0.21	359	0.07	0.01
hwatt_2	229152	2.50	18.54	11712	12.01	1.43
mc_fe	91252	0.86	9.68	6823	16.85	0.46
orsreg_1	1389670	15.98	604.12			
pde_9511	57749	0.56	2.71	2882	2.01	0.26
pores_2	294534	2.93	50.45	4203	7.86	0.48
steam2	113548	2.44	16.20	6396	5.07	0.32
fffff800						
nzfri				2565	4.62	0.95
pilot4	30486	0.30	2.07	1810	1.38	0.08
scsd6	11017	0.05	0.80	587	0.36	0.07
seba	119477	4.76	21.19	21698	19.16	1.84
shell	94911	0.85	11.88	1388	1.49	0.10
standata				1041	0.36	0.05
woodw	164469	3.97	27.15	8874	12.04	1.27

Table A.42 Time for the modification and factorization without preordering

Table A.42 shows the time of patterm modification and the time of IC factorization with reverse Cuthill-McKee degree ordering.

	MPADD			MPDROP		
name	nz(R)	T(mpadd)	T(ic)	nz(R)	T(mpdrop)	T(ic)
amoco1	38062	0.71	7.07	5319	3.45	0.68
belladit	3161	0.03	0.14	698	0.07	0.02
bellmedt	529767	66.38	228.34	31369	128.69	4.35
illc1850	12633	0.13	1.29	2762	1.09	0.25
well1850	12633	0.13	1.29	2762	1.09	0.25
well1033	3107	0.02	0.33	2081	0.25	0.24
dunes8	29043	0.32	1.87	2845	1.67	0.25
conev8				2268	0.99	0.14
strat8	98433	1.17	7.91	7433	13.34	1.76
bnl1	93905	2.08	12.71	2543	7.07	0.15
ship12s	7462	0.04	0.53	4449	1.97	0.39
gen	22603	0.24	1.18	2786	1.43	0.19
fs_760_1	38624	0.46	1.80	2457	3.15	0.18
fs_760_3				2457	3.15	0.18
gre_115	2479	0.02	0.05	328	0.04	0.01
hwatt_2				5336	14.31	1.06
mc_fe	98277	1.73	11.91	18120	15.69	1.27
orsreg_1	317744	4.85	31.68	6102	18.97	1.49
pde_9511	41618	0.47	1.76	2344	1.92	0.24
pores_2	83281	1.11	7.32	3639	5.16	0.45
saylr4	639127	9.66	77.15	10310	47.71	4.01
steam2	64444	1.63	5.48	6364	5.08	0.32
fffff800						
nzfri	57241	0.83	7.10	4044	2.15	0.38
pilot4	24234	0.39	1.41	1749	1.13	0.08
scsd6	3406	0.02	0.19	1192	0.09	0.06
seba						
shell				1218	0.97	0.10
standata						
woodw	198501	4.98	36.78	8968	13.54	1.27

Table A.43 Time for the modification and factorization with the reverse Cuthill McKee ordering

Table A.44 shows the performance of SPARSPAK-A with three orderings.

	reverse Cuthill-McKee			refined quotient tree			minimum degree		
name	nflops	nflops	time	nflops	nflops	time	nflops	nflops	time
amoco1	3.823	0.132	1.145	8.797	0.207	3.355	3.250	0.122	1.459
belladit	0.162	0.012	0.078	0.202	0.017	0.132	0.149	0.012	0.07
bellmedt	361.597	2.117	80.514	451.962	3.673	115.541	343.701	2.063	162.357
illc1850	103.631	0.936	22.432	12.506	0.646	3.342	0.122	0.031	0.117
well1850	103.631	0.936	22.294	12.498	0.646	3.409	0.122	0.031	0.117
well1033	6.805	0.163	1.684	1.585	0.141	0.416	0.032	0.010	0.036
dunes8	1.744	0.131	0.783	2.245	0.132	1.174	0.670	0.080	0.391
*conev8									
strat8	4.605	0.392	2.274	5.232	0.415	3.432	4.516	0.351	2.37
bnl1	32.903	0.508	7.808	33.416	0.592	9.306	11.219	0.238	5.958
ship12s	9.600	0.332	2.295	0.143	0.047	0.229	0.100	0.027	0.256
gen	17.176	0.425	4.533	16.379	0.702	4.181	0.241	0.039	0.173
fs_760_1	2.545	0.171	1.058	4.984	0.206	2.247	2.113	0.144	1.087
*fs_760_3	2.545	0.171	1.057						
gre_115	0.102	0.012	0.062	0.118	0.012	0.091	0.042	0.007	0.028
hwatt_2				51.227	0.990	17.234	36.402	0.891	16.088
mc_fe	11.244	0.331	3.163	30.462	0.586	10.562	10.278	0.313	4.782
orsreg_1	48.922	1.235	13.106	52.834	1.275	18.506	62.724	1.227	27.731
pde_9511	1.998	0.164	0.95	2.112	0.174	1.401	1.519	0.127	0.787
*pores_2	13.686	0.472	3.834	20.744	0.504	6.502	3.531	0.188	1.750
*saylr4	120.897	2.518	31.177	131.668	2.568	43.052	106.067	2.062	46.956
*steam2	6.351	0.249	1.850	6.572	0.279	3.008	7.159	0.241	3.173
*fffff800									
*nzfri				15.132	0.705	4.084	0.849	0.075	0.505
pilot4	3.444	0.139	1.036	5.117	0.151	1.823	0.955	0.058	0.476
scsd6	.279	0.022	0.131	0.063	0.017	0.068	0.055	0.010	0.036
seba	19.935	0.342	4.873	12.531	0.502	3.222			
*shell									
*standata									
woodw	48.425	0.867	11.721	58.638	0.719	18.109	3.274	0.192	1.658

Table A.44 Performance information of normal equations: SPARSPAK-A

Table A.45 shows the performance of SPARSPAK-B with and without row orderings.

	compute residule		without row ordering		with row ordering	
name	nflops	time	nflops	time	nflops	time
amoco1	38082	1.723	103.238	33.216	70.957	23.259
belladit	2091	0.297	3.126	1.197	2.759	1.060
bellmedt	63674	5.236	$\geq 10^4$	2890.406	$\geq 10^4$	2572.652
illc1850	14308	1.544	4.950	2.802	4.956	2.804
well1850	14308	1.550	4.981	2.811	4.983	2.810
well1033	7831	0.865	0.931	0.896	0.876	0.873
dunes8	36258	5.205	144.291	52.356	27.340	11.542
conev8	22576	3.214	68.616	24.880	25.238	9.895
strat8	111578	14.026	$\geq 10^4$	339.616	180.826	62.732
bnl1	12208	1.510	103.219	35.393	187.977	61.968
ship12s	14022	2.646	5.416	3.260	2.540	2.100
gen	6272	1.359	10.313	5.636	2.105	1.350
fs_760_1	7496	0.662	6.491	3.030	6.409	2.997
fs_760_3	7496	0.674	69.625	20.196	69.536	20.311
gre_115	651	0.094	0.140	0.103	0.149	0.109
hwatt_2	15262	1.571	111.236	34.504	123.568	37.950
mc_fe	25912	1.042	30.481	15.381	28.228	14.758
orserg_1	18543	1.892	214.884	65.463	219.834	67.222
pde_9511	6603	0.785	5.039	2.040	5.338	2.143
pores_2	12061	1.075	11.559	4.491	12.020	4.614
saylr4	29444	3.029	342.830	104.965	360.508	109.700
steam2	14960	0.710	20.630	7.035	22.105	7.469
fffff800	7943	0.738	7.609	3.703	6.680	3.325
nzfri	22945	2.870	22.195	9.488	41.933	15.767
pilot4	7145	0.826	18.425	6.573	13.350	5.018
scsd6	8366	1.084	2.685	1.446	2.700	1.430
seba	6930	0.842	65.156	22.150	41.781	14.942
shell	8450	1.385	4.443	2.272	5.419	2.599
standata	5188	2.453	3.457	1.619	2.958	1.427
woodw	54288	6.852	291.954	94.458	356.444	112.765

Table A.45 Performance information of QR method: SPARSPAK-B

REFERENCES

[1] M. Arioli, I. Duff, and P. P. M. De Rijk. On the augmented system approach to sparse least-squares problems. *Numerische Mathematik*, 55:667–684, 1989.

[2] O. Axelsson, editor. *Proc. of Conf. on Preconditioned Conjugate Gradient Methods*, University of Nijmegen, The Netherlands, 1989.

[3] O. Axelsson and P. S. Vassilevski. A survey of multilevel preconditioned iterative methods. *BIT*, 29:769–793, 1989.

[4] R. Beauwens. Approximate factorizations with S/P consistently ordered M-factors. *BIT*, 29:658–681, 1989.

[5] M. W. Berry. *Multiprocessor sparse SVD algorithms and Applications*. PhD thesis, University of Illinois Urbana-Champaign, 1990. Also available as Technical Report # 1049, Center for Supercomputing Research and Development, University of Illinois at Urbana – Champaign.

[6] M. W. Berry and R. J. Plemmons. Algorithms and experiments for structural mechanics on high-performance architectrues. *Computer Methods in Applied Mechanics and Engineering*, 64:487–507, 1987.

[7] A. Björck. Iterative refinement of linear least squares solutions. *BIT*, 7:257–278, 1967.

[8] A. Björck. Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT*, 7:1–21, 1967.

[9] A. Björck. SSOR preconditioning methods for sparse least squares problems. In J. F. Gentleman, editor, *Proceedings of the Computer Science and Statistics: 12-th Annual Symposium on the Interface*, pages 21–25, University of Waterloo, Waterloo, Ontario, Canada, May 1979.

[10] A. Björck. Stability analysis of the method of seminormal equations for linear least squares problems. *Linear Algebra and its Applications*, 88/89:31–48, 1987.

[11] A. Björck. A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations. *BIT*, 28:659–670, 1988.

[12] A. Björck. Least squares methods. Technical Report LiTH-MAT-R-1991, Department of Mathematics, Linköping University, 1991.

[13] A. Björck. Pivoting and stability in the augmented system method. Technical Report LiTH-MAT-R-1991-30, Department of Mathematics, Linköping University, 1991.

[14] A. Björck and I. S. Duff. A direct method for the solution of sparse linear least squares problems. *Linear Algebra and its Applications*, 34:43–67, 1980.

[15] J. R. Bunch and L. Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation*, 31:162–179, 1977.

[16] T. F. Chan. Fourier analysis of relaxed incomplete factorization preconditioners. *SIAM Journal of Scientific and Statistical Computing*, 12(3):668–680, May 1991.

[17] I.C. Chio, C. L. Monma, and D.F. Shanno. Further development of a primal-dual interior point method. *ORSA Journal on Computing*, 2(4):304–311, 1990.

[18] T. F. Coleman, A. Edenbrandt, and J. R. Gilbert. Predicting fill for sparse orthogonal factorization. *Journal of the ACM*, 33(3):517–532, 1986.

[19] R. J. Van der Bei. Splitting dense columns in sparse linear systems. *Linear Algebra and its Applications*, pages 107–117, 1991.

[20] I. S. Duff. The solution of large-scale least squares problems on supercomputers. *Annals of Operations Research*, 22:241–252, 1990.

[21] I. S. Duff and G. A. Meurand. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29:635–657, 1989.

[22] I. S. Duff and J. K. Reid. A comparison of some methods for the solution of sparse overdetermined systems of linear equations. *Journal of the Institute of Mathematics and Its Applications*, 17(3):267–280, 1976.

[23] V. Eijkhout. Beware of modified incomplete point factorizations. Technical Report CSRD Report No. 1048, CSRD, University of Illinois at Urbana-Champaign, 1990.

[24] V. Eijkhout. Beware of unperturbed modified incomplete factorizations. Technical Report CSRD Report No. 1109, CSRD, University of Illinois at Urbana-Champaign, 1991.

[25] D. H. Evans, editor. *Preconditioning Methods: Analysis and Applications*. Gordon and Breach Science Publishers, 1983.

[26] T. Fiala. On the monotonicity of incomplete factorizations. *Numerische Mathematik*, 57:473–479, 1990.

[27] K. Gallivan, A. Sameh, and Z. Zlatev. A parallel hybrid sparse linear system solver. *Computing Systems in Engineering*, 1(2-4):183–195, 1990.

[28] W. M. Gentleman. Least squares computations by Givens transformations without square roots. *Journal of the Institute of Mathematics and Its Applications*, pages 329–336, 1973.

[29] A. George and M. T. Heath. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra and its Applications*, 34:69–83, 1980.

[30] A. George and J. Liu. Householder reflections versus Givens rotations in sparse orthogonal decomposition. *Linear Algebra and its Applications*, 88/89:223–238, 1987.

[31] A. George, J. Liu, and E. Ng. Row-ordering schemes for sparse Givens transformations. I. bipartite graph model. *Linear Algebra and its Applications*, 61:55–81, 1984.

[32] A. George, J. Liu, and E. Ng. A data structure for sparse QR and LU factorizations. *SIAM Journal of Scientific and Statistical Computing*, 9(1):100–121, 1988.

[33] A. George and J. W.-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.

[34] A. George and E. Ng. Orthogonal reduction of sparse matrices to upper triangular form using Householder transformations. *SIAM Journal of Scientific and Statistical Computing*, 7(2):460–472, 1986.

[35] P.E. Gill, G. H. Golub, and M.A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28:505–35, 1974.

[36] G. H. Golub and R. J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra and its Applications*, 34:3–27, 1980.

[37] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins, 2nd edition, 1989.

[38] J. F. Grcar. Matrix streching for linear equations. Technical Report SAND90-8723, Sandia National Laboratory, 1990.

[39] A. Greenbaum and G. H. Rodrigue. Optimal preconditioners of a given sparsity pattern. *BIT*, 29:610–634, 1989.

[40] M. T. Heath. Numerical methods for large sparse linear least squares problems. *SIAM Journal of Scientific and Statistical Computing*, 5(3):497–513, 1984.

[41] M. T. Heath, R. J. Plemmons, and R. C. Ward. Sparse orthogonal schemes for structural optimization using the force method. *SIAM Journal of Scientific and Statistical Computing*, 5(3):514–532, 1984.

[42] D. James. *Conjugate Gradient Methods for Constrained Least Squares Problems*. PhD thesis, North Carolina State University, 1990.

[43] A. Jennings and M. A. Ajiz. Incomplete methods for solving $A^T Ax = b$. *SIAM Journal of Scientific and Statistical Computing*, 5(4):978–987, 1984.

[44] A. Jennings and G. M. Malik. Partial elimination. *Journal of the Institute of Mathematics and Its Applications*, 20:307–316, 1977.

[45] C.L. Lawson and R.J. Hanson. *Solving Least Square Problems*. Prentice-Hall, 1974.

[46] J. Liu. On general row merging schemes for sparse Givens transformations. *SIAM Journal of Scientific and Statistical Computing*, 7(4):1190–1211, 1986.

[47] T. A. Manteuffel. Shifted incomplete Choleski factorization. In I. S. Duff and G.W. Stewart, editors, *Sparse Matrix Proceedings 1978*, Philadelphia,PA, 1979. SIAM Publications.

[48] T.L. Markham, M. Neumann, and R.J. Plemmons. Convergence of a direct-iterative method for large-scale least square problems. *Linear Algebra and its Applications*, 69:155–167, 1985.

[49] P. Matstoms. The multifrontal solution of sparse linear least squares problems. Technical Report LIU-TEK-LIC-1991:33, Department of Mathematics, Linköping University, 1991.

[50] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, 31(137):148–162, 1977.

[51] M.Fortin and R. Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. North-Holland, 1983.

[52] R. Morandi and F. Sgallari. Parallel algorithms for the iterative solution of sparse least squares problems. *Parallel Computing*, 13:271–280, 1990.

[53] G. Ostrouchov. Symbolic Givens reduction and row-ordering in large sparse least squares problems. *SIAM Journal of Scientific and Statistical Computing*, 8(3):248–264, 1987.

[54] C. Paige and M. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, 1982.

[55] C. C. Paige. Bidiagonalization of matrices and the solution of linear equations. *SIAM Journal of Numerical Analysis*, 11:197–209, 1974.

[56] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of equations and least squares problems. Technical Report STAN-CS-73-399, Stanford University, 1973.

[57] G. Peters and J. H. Wilkinson. The least squares problem and pseudo-inverses. *The Computer Journal*, 13:309–316, 1970.

[58] Youcef Saad. Preconditioning techniques for nonsymmetric and indefinite linear systems. *Journal of Computational and Applied Mathematics*, 24:89–105, 1988.

[59] M. A. Saunders. Sparse least squares by conjugate gradients: A comparison of preconditioning methods. In J. F. Gentleman, editor, *Proceedings of the Computer Science and Statistics: 12-th Annual Symposium on the Interface*, pages 15–20, University of Waterloo, Waterloo, Ontario, Canada, May 1979.

[60] J. H. Wilkinson. A priori error analysis of algebraic processes. In *Proc. International Congress of Mathematicians*, pages 119–129, 1968.

[61] G. Wittum and F. Liebau. On truncated incomplete decompositions. *BIT*, 29:719–740, 1989.

[62] Ulrike Meier Yang. Preconditioned conjugate gradient-like methods for nonsymmetric linear systems. Technical Report 1210, CSRD, University of Illinois at Urbana-Champaign, 1992.

[63] D. M. Young. *Iterative soultion of large linear systems.* Academic Press, 1971.

[64] Z. Zlatev. Comparison of two pivotal strategies in sparse plane rotations. *Computers and Mathematics with Applications*, 8(2):119–135, 1982.

[65] Z. Zlatev. *Computational Methods for General Sparse Matrices.* Kluwer Academic Publishers, 1991.

[66] Z. Zlatev and H.B.Nielson. Solving large and sparse linear least squares problems by conjugate gradient algorithm. *Computers and Mathematics with Applications*, 15:185–202, 1988.

VITA

Xiaoge Wang was born on the 12th of August, 1957 in Fujian, China. She finished her high school education in 1974. She spent three and a half years in the countryside where she learned the real meaning of life before she was admitted into Tsinghua University in 1978. She studied in Tsinghua for her B.S. and M.S. degrees in Computer Science and later joined the faculty of the University. Xiaoge Wang came to the United States of America as a visiting scholar and later decided to pursue Ph.D degree. After obtaining her Ph.D in 1993, Xiaoge Wang continued her study in scientific computing as a postdoctoral research associate at Indiana University at Bloomington.