Proceedings of the

Computer Science and Statistics:
12th Annual Symposium on the Interface

Edited by Jane F. Gentleman

May 10 & 11, 1979

University of Waterloo

Waterloo, Ontario, Canada

Design and Composition: Arnie Dyck and Richard Beach, Dept. of Computer Science
Design and Printing: Graphic Services, University of Waterloo

Availability of Proceedings

12th (1979)	Jane F. Gentleman Dept. of Statistics University of Waterloo Waterloo, Ontario Canada N2L 3G1
11th (1978)	Institute of Statistics North Carolina State Univ. P.O. Box 5457 Raleigh, North Carolina 27650
10th (1977)	David Hogben Statistical Engineering Laboratory Applied Mathematics Division National Bureau of Standards U.S. Dept. of Commerce Washington, D.C. 20234
9th (1976)	Prindle, Weber, and Schmidt, Inc. 20 Newbury St. Boston, Massachusetts 02116
8th (1975)	Health Sciences Computing Facility, AV-111 Center for Health Sciences Univ. of California Los Angeles, California 90024
7th (1974)	Statistical Numerical Analysis and Data Processing Section 117 Snedecor Hall Iowa State Univ. Ames, Iowa 50010
4,5,6th (1971, 1972, 1973)	Western Periodicals Company 13000 Raymer Street North Hollywood, California 91605

Future Interface Symposia

13th Interface Symposium (1980):
Univ. of California at Los Angeles
Chairman: Robert M. Elashoff

14th Interface Symposium (1981):
Carnegie-Mellon Univ., Pittsburgh
Co-Chairmen: William F. Eddy & Michael I. Shamos

SPARSE LEAST SQUARES BY CONJUGATE GRADIENTS: A COMPARISON OF PRECONDITIONING METHODS

Michael A. Saunders, DSIR, New Zealand and Stanford University, California

The conjugate-gradient method is ideal for sparse least squares because of its simplicity and low storage requirements, and because it can be relied upon to converge to a satisfactory solution. However, the number of iterations required can be unacceptably high if the design matrix X is ill-conditioned.

In this context, preconditioning means a change of variables aimed at accelerating convergence. For some nonsingular C, the usual linear model $y = X\beta + e$ is transformed into $y = \bar{X}\bar{\beta} + e$, where $\bar{X} = XC^{-1}$. (The product is not computed explicitly.) If $\bar{X}$ is better conditioned than X, the transformed problem should require fewer iterations.

Here we give some computational results for two particular choices of C. The SSOR approach of Björck and Elfving promises to give useful savings generally. When X is very ill-conditioned, it appears that an LU factorization of X (with $C = U$) can lead to even faster convergence.

1. INTRODUCTION

The problem discussed here is the usual linear model

$$y = X\beta + e \qquad (1)$$

in which X is large and sparse. If X is n by p, we have $n \geq p$ with both n and p large. The least-squares solution required satisfies the normal equations $X^TX\beta = X^Ty$, but we assume that both X^TX and $(X^TX)^{-1}$ are too expensive to compute.

For such problems, conjugate-gradient methods offer the advantages of simplicity and low storage requirements. Several conjugate-gradient (cg) algorithms are known (we refer to two of them later). In practice they must be regarded as iterative methods, and much of the work per iteration lies in computing two matrix-vector products of the form

$$Xp_i \quad \text{and} \quad X^Tr_i \qquad (2)$$

for certain vectors p_i and r_i. If X is well-conditioned, an accurate least-squares solution β can often be obtained in far less than p iterations.

Let the condition of X be defined as $\text{cond}(X) = \sigma_{max}/\sigma_{min}$ where $\sigma_i > 0$ are the singular values of X (ignoring any $\sigma_i = 0$). Also, let ε be the relative precision of the machine being used; for example, $\varepsilon \simeq 10^{-15}$ for double-precision arithmetic on an IBM 370. A rough guide to the number of iterations required is as follows:

cond(X)	cg iterations
10	$\frac{1}{10}p$
100	$\frac{1}{2}p$
$\varepsilon^{-1/4}$	$1 - 2p$
$\varepsilon^{-1/2}$	$5 - 15p$

The number of iterations also depends on the distribution of the singular values (in general if the σ_i occur in clusters there will be fewer iterations than otherwise) and to a lesser extent on the accuracy required for the computed β.

1.1. Preconditioning

The broad aim of preconditioning is to transform the original problem (1) into one of the form

$$y = \bar{X}\bar{\beta} + e$$

in which $\bar{X}$ is better conditioned than X. To this end, let C be a nonsingular matrix of order p and consider the problem

$$y = XC^{-1}\bar{\beta} + e \qquad (3)$$

in which $\bar{X} = XC^{-1}$ is not computed explicitly but is regarded as the product of two separate operators. When a cg algorithm is applied to (3) to compute $\bar{\beta}$, the matrix-vector products in (2) become

$$XC^{-1}p_i \qquad \text{and} \qquad C^{-T}X^T r_i \ . \qquad (4)$$

Hence the only additional work required per iteration lies in solving two linear equations of the form

$$Ct_i = p_i \qquad \text{and} \qquad C^T s_i = w_i \ . \qquad (5)$$

On termination the original parameters can be obtained from one final system $C\beta = \bar{\beta}$.

Clearly it will be advantageous to use this type of preconditioning if the extra work in (5) is offset by a sharp reduction in the number of cg iterations. Our aim here is to compare the effectiveness of two particular matrices C. In both cases C is upper triangular. The comparison is more computational than analytical. The results suggest that the best choice of C depends on the condition of X.

2. SSOR PRECONDITIONING

This is a new method due to Björck and Elfving (1978). (See also Björck (1979) in this proceedings.) It was derived as a means for accelerating the Symmetric SSOR method when the latter is applied to the normal equations. For simplicity we shall assume that columns of X have been scaled to unit length. Then $X^TX = L + I + L^T$ where L is strictly lower triangular, and the conditioning matrix suggested by SSOR proves to be

$$C = I + \omega L^T$$

for some relaxation parameter $\omega \geq 0$. Björck and Elfving show that the matrix-vector products in (4) can be computed efficiently without forming L. They give a Fortran implementation of the resulting cg algorithm, subroutine CGPCNE.

Algebraically, the matrix-vector products with $C = I + \omega L^T$ involve exactly twice as much arithmetic as when $C = I$. Thus in broad terms, SSOR preconditioning will give a useful saving in computation time if the total iteration count is reduced by a factor of 2 or more. The method's advantages are that it applies to arbitrary sparse matrices, it can be coded easily as demonstrated by subroutine CGPCNE, and in this form it requires no additional storage to implement the preconditioning.

2.1. The Choice of ω

For an arbitary statistical application the optimal relaxation parameter ω is not likely to be known in advance. Björck and Elfving recommend setting $\omega = 1$ when no other information is available. On the least-squares problems of Section 4 below, subroutine CGPCNE converged significantly more quickly with $\omega = 0.9$, 1 and 1.1 than it did with the more extreme values 0, 1.2, 1.5 and 1.9. Although this range of values is incomplete, it appeared that $\omega = 1$ was very close to optimal for these examples. We therefore use $\omega = 1$ in the comparisons below.

3. LU PRECONDITIONING

Gaussian elimination with row interchanges may be used to compute a stable, sparse factorization $X = LU$, where L is a product of permutations and elementary triangular matrices, and U is upper triangular. The row interchanges can be biased towards keeping L well-conditioned. Since $L = XU^{-1}$, the natural conditioning matrix is

$$C = U$$

and if cond(L) is significantly smaller than cond(X) a useful reduction in cg iterations should result. This general approach was first suggested by Björck (1976). Some preliminary computational results have been given by Paige and Saunders (1978).

An LU factorization may be computed in several ways, each giving a different L and U. In this context a product of elementary matrices will be used to reduce X to triangular form:

$$\ldots E_3E_2E_1X = \begin{bmatrix} U \\ 0 \end{bmatrix},$$

so that L would be the product $E_1^{-1}E_2^{-1}E_3^{-1}\ldots\begin{bmatrix} I \\ 0 \end{bmatrix}$. (This is never needed explicitly.) The condition of L is controlled by keeping the multipliers in each E_j less than $1/\tau$ for some pivot tolerance τ in the range $(0,1)$.

3.1. Row-wise Elimination

The simplest method is to eliminate the rows of X one by one in any convenient order. The matrices E_j do not need to be saved. The nonzeros in U can be stored compactly by rows in an ordered list, with pointers marking the start of each row. Whenever a row interchange is required (thus causing "fill-in") a modified row can be moved to the end of the list, leaving behind its previous nonzeros. Occasional compression of the list recovers disused storage.

The time to compute U in this manner is equivalent to about 5 or 10 cg iterations. This is not a serious overhead.

In implementing this row-wise elimination we included a preliminary pass through the rows of X to select a triangular subset with maximal diagonal elements. These rows can be placed directly into U. The remaining rows were included one at a time in their natural order. The pivot tolerance used was $\tau = 0.9$. (The resulting LU factors were slightly more sparse and well-conditioned than with $\tau = 0.99$.)

Subsequent use of the operator XU^{-1} involves back-substitution with U and multiplication with X. Similarly for the transposed operator. Hence the additional work per iteration depends solely on the density of U. If U contained as many nonzeros as X, the time per iteration would be virtually the same as for SSOR preconditioning. However there is often very little net fill-in, and so U is likely to contain only p/n times as many nonzeros as X. In statistical applications this factor is typically less than $1/2$.

For problems A and B below ($n = 1033$, $p = 320$), the ratio of nonzeros in U and X proved to be exactly $p/n = 0.31$. For two larger examples ($n = 1850$, $p = 712$) the ratio was 0.40 and 0.49, compared with $p/n = 0.39$. This confirms that the time per iteration with LU preconditioning will typically be midway between the times for cg with and without SSOR preconditioning.

3.2. Column-wise Elimination

If columns rather than rows are eliminated one at a time there is reason to believe (e.g., Wilkinson (1965) pp. 162-166) that the final L will be better conditioned. Hence the final U should be more effective for preconditioning.

In this case some additional storage is required for the matrices E_j during the factorization, but there are no fill-in difficulties with U, which is formed sequentially by columns. A rather complex strategy is needed to order the rows and columns of X. The Markowitz strategy used in many linear-programming codes can easily be generalized to handle a rectangular X and would probably be ideal. (Duff and Reid (1976) have used such a scheme in some experiments with non-iterative least-squares methods.)

This form of LU factorization has not been tested here, but we wish to note certain implications. First, suppose that the rows of X selected to be "pivot rows" form a submatrix X_1. Thus for some permutation P,

$$PX = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

where X_1 is square and nonsingular. The LU factorization of X is really just the LU factorization of PX without further row interchanges. Ignoring any column permutations, we have

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} U = LU .$$

We can now make the following deductions.

1. $X_1 = L_1U$ is the factorization that would be obtained if the same column-wise elimination were applied to X_1.

2. L is well-conditioned and $\|L\|$ is not large.
3. L_1 is well-conditioned and $\|L_2\|$ is not large.
4. $\|L_2 L_1^{-1}\|$ should not be large.
5. $\begin{bmatrix} I \\ X_2 X_1^{-1} \end{bmatrix} = \begin{bmatrix} I \\ L_2 L_1^{-1} \end{bmatrix}$ should be reasonably well-conditioned.

(By "not large" we mean that the quantity is not directly related to $\mathrm{cond}(X)$, although it may be a function of n and p.)

The conclusion is that $C = X_1$ should be useful as a conditioning matrix.

The least-squares problem to be solved now takes the form

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} I \\ X_2 X_1^{-1} \end{bmatrix} \bar{\beta} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} .$$

To improve efficiency a new factorization $X_1 = L_1 U$ should be computed with greater emphasis on the sparsity of the factors. (The condition of L_1 is no longer vital, so a smaller pivot tolerance τ should be used.) If L_1 and U are almost as sparse as X_1, the subsequent time per iteration should not be much greater than without preconditioning.

It remains unclear which of the options

$$C = I$$
$$C = I + \omega L^T$$
$$C = U$$
$$C = L_1 U = X_1$$

should be used in general, but the choice $C = X_1$ has given excellent results in one practical example quoted by Chen (1975). In this case X_1 was already triangular and was known in advance. For other cases, column-wise elimination may be the only practical method for finding a useful X_1.

4. COMPUTATIONAL RESULTS

Here we compare the effectiveness of the matrices

$$C = I$$
$$C = I + L^T$$
$$C = U$$

for solving the least-squares problem $y = \bar{X}\bar{\beta} + e$ where $\bar{X} = XC^{-1}$. The run-times quoted below are seconds of processor time on a Burroughs B6700, using Fortran routines and single-precision arithmetic ($\varepsilon \approx 0.7 \times 10^{-11}$).

Subroutine LSQR of Paige and Saunders (1978) was used as the basic cg algorithm for all three operators $\bar{X}$. The SSOR operator was implemented as in Björck and Elfving's subroutine CGPCNE, with very slight changes to take advantage of the fact that the columns of X were already normalized.

LSQR provides estimates of $\|\bar{X}\|$, $\mathrm{cond}(\bar{X})$ and standard errors for $\bar{\beta}$. It terminates when

$$\frac{\|\bar{X}^T r\|}{\|\bar{X}\| \ \|r\|} \leq \mathrm{ATOL}$$

where $r = y - \bar{X}\bar{\beta}$ and ATOL is set by the user. There are theoretical reasons for choosing ATOL to be the relative error in the data $\bar{X}$. In this case the relevant value was $\mathrm{ATOL} = 10^{-8}$.

4.1. Test Problems

Two representative least-squares problems were obtained from an application in geophysics involving the analysis of gravity-meter observations (see Paige and Saunders (1978)). In both cases X is 1033 by 320, with 3 to 5 nonzero elements per row. The cases are distinguished by the condition of X:

Problem A	$\mathrm{cond}(X_A) \approx 3100$,
Problem B	$\mathrm{cond}(X_B) \approx 10^6$.

Although Problem B is extremely ill-conditioned, only some of the parameters are poorly determined. When it became clear that convergence was intolerably slow (without preconditioning), the model was reformulated to obtain Problem A. Algebraically, $X_A = X_B T$ for some triangular T which shifted the origin for some of the parameters without altering the sparsity of X_B. Even with direct methods, unnecessary ill-conditioning of this kind should be avoided, but it is all the more important when cg methods are in use.

4.2. Comparison

Some statistics for the three conjugate-gradient methods are summarized in the tables and figures below. The main results apparent are the following:

1. For both Problems A and B, SSOR preconditioning led to a substantial reduction in iterations and a significant saving in total computation time.
2. LU preconditioning had a negative effect on Problem A, but performed excellently on Problem B. In particular, the rate of convergence was almost identical for the two examples in spite of problem B's poor condition.

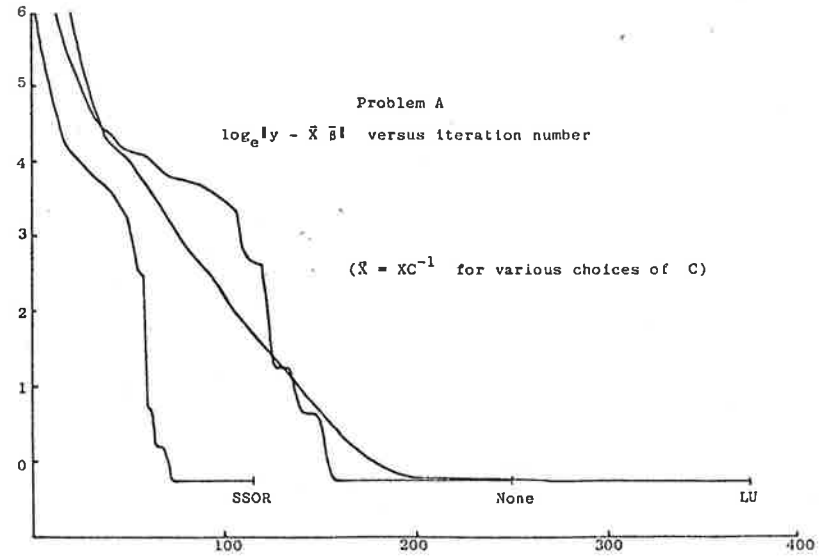
Preconditioning method		cond(XC^{-1})	Time per iteration	Total iterations	Total time
None	C=I	3100	1.4	250	340
SSOR	$C=I+L^T$	900	2.4	114	260
LU	C=U	2600	1.6	375	600

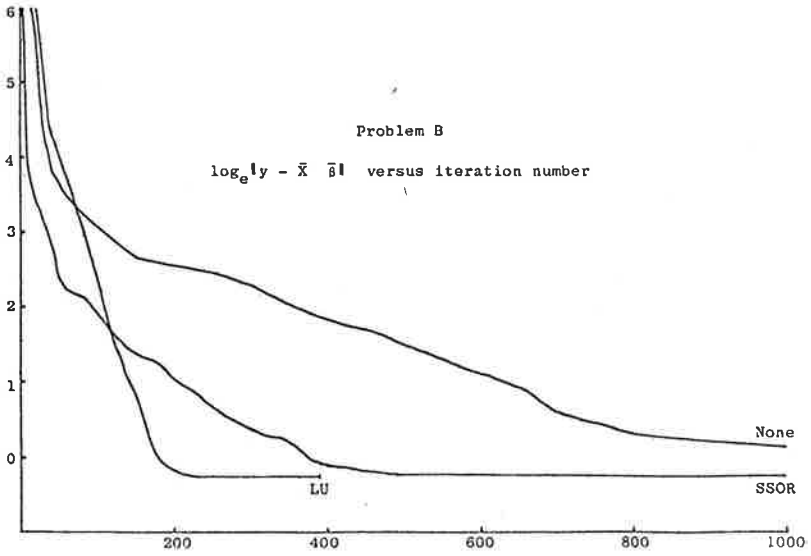
Statistics for Problem A

Preconditioning method		cond(XC^{-1})	Time per iteration	Total iterations	Total time
None	C=I	10^{6*}	1.4	4000*	5500*
SSOR	$C=I+L^T$	10^5	2.4	1600	3700
LU	C=U	2900	1.6	390	620

*Estimates

Statistics for Problem B





5. DISCUSSION

Until the various preconditioning methods are compared on a wider range of examples it would be unsafe to draw too many conclusions. At this stage it does appear that Björck and Elfving's SSOR approach has many advantages for sparse least-squares problems in general, and yet computation time is likely to remain unacceptably high when the design matrix X is very ill-conditioned.

For LU preconditioning the limitation appears to be at the other end of the spectrum. If X is already well-conditioned, the transformed operator $XU^{-1} = L$ need not be any better. Even if it is, the distribution of its singular values may be less favorable and lead to a greater number of cg iterations (cf. Problem A). Only when X is extremely ill-conditioned is there room for a substantial reduction in condition number (cf. Problem B).

One might argue that well-formulated models are typically very well-conditioned and will converge rapidly without transformation, whereas badly-conditioned problems are very sensitive to data changes and should not be solved at all.

However, we believe that many real-life applications lie somewhere between the two extremes. The conjugate-gradient approach has so many intrinsic advantages that any effort to accelerate its convergence is very easily justified. This paper has given results for two practical methods and suggested a third. It is hoped that the methods will find widespread use amongst practitioners. The computer codes used here are written in ANSI Standard Fortran and are available from the author.

6. REFERENCES

[1] Å. BJÖRCK (1976). Methods for sparse linear least squares problems. In J.R. Bunch and D.J. Rose (eds.), Sparse Matrix Computations, Academic Press, New York, 177-199.

[2] Å. BJÖRCK (1979). SSOR-preconditioning methods for sparse least squares problems. In this proceedings.

[3] Å. BJÖRCK and T. ELFVING (1978). Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. Research Report LiTH-MAT-R-1978-5, Department of Mathematics, Linköping University, Sweden. To be published in BIT, 19 (1979).

[4] Y.T. CHEN (1975). Iterative methods for linear least-squares problems. Research Report CS-75-04, Department of Computer Science, University of Waterloo, Canada.

[5] I.S. DUFF and J.K. REID (1976). A Comparison of some methods for the solution of sparse overdetermined systems of linear equations. J. Inst. Math. and its Applics., 14, 267-280.

[6] C.C. PAIGE and M.A. SAUNDERS (1978). A bidiagonalization algorithm for sparse linear equations and least-squares problems. Technical Report SOL 78-19, Department of Operations Research, Stanford University, USA.

[7] J.H. WILKINSON (1965). The Algebraic Eigenvalue Problem. Oxford University Press (Clarendon), London and New York.