# A comparison of some methods for solving sparse linear least squares problems

by

Alan George and Esmond Ng

Department of Computer Science
University of Waterloo

Research Report CS-80-42

September 1980

# A comparison of some methods for solving sparse linear least squares problems

Alan George[1] and Esmond Ng[2]

Department of Computer Science

University of Waterloo

## Abstract

The method of normal equations, the Peters-Wilkinson algorithm and an algorithm based on Givens rotations for solving large sparse linear least squares problems are discussed and compared. Numerical experiments show that the method of normal equations should be considered when the observation matrix is sparse and numerical stability is not important. However, if numerical stability is the major issue, the algorithm based on Givens rotations is then preferable.

## I. Introduction

Let $A$ be an $m$ by $n$ sparse matrix with $m \geq n$, and consider the system of linear equations

$$A x = b \quad , \tag{1.1}$$

where $b$ and $x$ are vectors of length $m$ and $n$ respectively. In general, there may not exist a solution $x$

---

(1) The work of this author was supported in part by the Canadian Natural Sciences and Engineering Research Council under grant A8111.

(2) The work of this author was supported in part by the Canadian Natural Sciences and Engineering Research Council through a postgraduate scholarship.

such that (1.1) is exactly satisfied. In those cases, (1.1) is usually solved in the "least squares" sense; that is, the solution $x$ is chosen to minimize the Euclidean norm of the residual vector

$$r = A x - b \quad . \tag{1.2}$$

Throughout this paper, we will assume that the columns of $A$ are linearly independent. Under this assumption, it is easy to show that the unique solution $x$ satisfies the symmetric positive definite $n$ by $n$ system of linear equations

$$A^T A x = A^T b \quad , \tag{1.3}$$

which is referred to as the normal equations.

However, it is well-known that computing $A^TA$ explicitly may not be desirable since the condition number of $A^TA$ is the square of that of $A$. Thus the matrix $A^TA$ may be quite ill-conditioned if $A$ is poorly conditioned, and the solution $x$ will be sensitive to perturbations in (1.3). Moreover, severe numerical cancellation and roundoff may result in computing $A^TA$ [6].

Several numerically stable algorithms have been proposed for solving (1.1) without computing $A^TA$ explicitly. In this paper, we will compare two such algorithms. In addition to being numerically stable, the two algorithms also attempt to exploit sparsity in $A$.

The first algorithm was originally proposed by Peters and Wilkinson for solving (1.1) without considering the sparsity of $A$ [8]. Recently, Björck and Duff have

advocated its use for sparse A [1]. The algorithm is based essentially on Gaussian elimination with complete pivoting. The second algorithm is due to George and Heath [4], and is based on the use of Givens rotations.

An outline of the remainder of this paper is as follows. In sections 2 and 3, we review briefly the two algorithms. Then some numerical experiments are provided in section 4 and some concluding remarks appear in section 5.

## II. The Peters-Wilkinson (P-W) Algorithm

The first step of this algorithm is the computation of an LU-decomposition of A using both row and column interchanges. Thus, we have

$$P A Q = L U \quad , \tag{2.1}$$

where P and Q are respectively m by m and n by n permutation matrices, L is an m by n unit lower trapezoidal matrix and U is an n by n upper triangular matrix. When A is sparse, the matrices P and Q are chosen to simultaneously maintain numerical stability and preserve sparsity. Then (1.1) can be written as

$$P A Q Q^T x = P b \quad , \text{ or}$$

$$L U Q^T x = P b \quad . \tag{2.2}$$

If $y = Q^T x$ and $d = Pb$, then (2.2) becomes

$$L U y = d \quad . \tag{2.3}$$

When $m = n$, the matrix $L$ is unit lower triangular, and the solution $x$ can be obtained by solving two triangular systems

$$L z = d ,$$

$$U y = z ,$$

and then computing

$$x = Q y .$$

However, if $m > n$, we let $z = Uy$ and (2.3) becomes

$$L z = d , \tag{2.4}$$

which is also a least squares problem. Apparently, nothing has been gained so far. However, experience has shown that if both row and column interchanges are used to limit the size of the off-diagonal elements in $L$, then $L$ is usually well-conditioned. Thus (2.4) can be solved via the normal equations; that is, we can solve

$$L^T L z = L^T d . \tag{2.5}$$

Then the solution $x$ can be obtained by solving

$$U y = z ,$$

and

$$x = Q y .$$

This algorithm is an attractive candidate for solving (1.1) when $A$ is sparse, because there already exist efficient sparsity-exploiting algorithms for computing the LU-decomposition of $A$ and for solving (2.5).

## III. An Algorithm Based on Givens Rotations [4]

The basic step in this algorithm is to determine an m by m orthogonal matrix Q which reduces A to an upper trapezoidal matrix

$$(3.1) \qquad Q\,A = \begin{bmatrix} R \\ O \end{bmatrix},$$

where R is an n by n upper triangular matrix. Then (1.1) can be written as

$$(3.2) \qquad Q\,A\,x = Q\,b\,.$$

Let

$$Q\,b = \begin{bmatrix} c \\ d \end{bmatrix},$$

where c and d are vectors of length n and (m-n) respectively. Then (3.2) is

$$\begin{bmatrix} R \\ O \end{bmatrix} x = \begin{bmatrix} c \\ d \end{bmatrix},$$

and the solution x can be obtained by solving a triangular system

$$(3.3) \qquad R\,x = c\,.$$

Note that if m = n, the vector d will be null.

Even though it is well-known that application of orthogonal transformations is numerically stable (except for some pathological situations), this approach is not as

popular as the previous one for solving (1.1) when A is sparse. Apparently it has been assumed that orthogonal transformations will cause unacceptable fill-in during the reduction of A.

Recently, George and Heath have proposed a new and efficient way to compute R [4], which we will refer to as the George-Heath (G-H) algorithm. They observe that the lower triangular matrix $R^T$ is mathematically equivalent to the Cholesky factor of the matrix $A^TA$. Furthermore any column permutation on the columns of A induces a symmetric permutation on the rows and columns of $A^TA$, or vice versa. Thus one can choose a column permutation P for A so that the Cholesky factor of $P^TA^TAP$ suffers low fill-in. Note that since no row or column interchanges are necessary during the Cholesky decomposition [9], the pivotal sequence is known once the permutation matrix P (or ordering) has been determined. Thus the positions of the non-zeros, and the storage requirement for the Cholesky factor (hence R) can be determined before the actual numerical decomposition begins. These two steps can be done without actually carrying out the transformation on A; only the non-zero structure of A is required. The readers are referred to [4] for more details.

After the data structure for R has been determined and set up, the rows of A can then be rotated one by one into R using Givens rotations. Thus, this algorithm has

the advantage that the storage requirement can be determined and fixed before any numerical computation is carried out. Moreover, if the orthogonal matrix Q is discarded, this algorithm requires no more storage than the normal equations.

## IV. Numerical Experiments

Two sparse matrix packages were used to implement the P-W and G-H algorithms. The first package was MA28 from Harwell [2], which was designed to solve general sparse systems of equations. It was used to compute the LU-decomposition of A in the P-W algorithm (equation (2.1)). The pivoting strategy, due to Markowitz [7], attempts to maintain numerical stability and preserve sparsity at the same time. This involves the use of a so-called "threshold pivoting" technique; during the decomposition, an element in the diagonal of the partially reduced matrix may be considered as a pivot if its magnitude is larger than the product of a user-specified threshold parameter and the absolute value of the element with the largest magnitude in that row and column. The threshold parameter we used in the experiments was 0.1.

The second package was SPARSPAK, developed at the University of Waterloo [3] and designed to solve sparse symmetric positive definite systems. It was used to

determine the non-zero structure of the Cholesky factor $R^T$ in the G-H algorithm and to solve the normal equations in (2.5) in the P-W algorithm. In both cases, the ordering (i.e. the column and row permutations) used was provided by the minimum degree algorithm [5] which is a symmetric version of the Markowitz's pivoting strategy.

The experiments were performed on an IBM 4341, and all times reported are in seconds. The storage requirements reported were provided either explicitly or implicitly through some variables appearing in the internal labelled common blocks used by the two packages. The programs were written in FORTRAN and compiled using the IBM Extended Optimizing FORTRAN Compiler. There are three sets of test problems which are typical least squares problems from finite element applications and surveying.

In the G-H method, only the ordering P of the columns of A is specified, since the non-zero structure of the Cholesky factor $R^T$ depends only on A and the column permutation. The row ordering does not have any effect on the sparsity of $R^T$. However, the cost of computing the Cholesky factor depends very much on the row ordering. Experience has shown that if the rows of AP are sorted in increasing order of last column subscripts, the cost of computing R can be reduced substantially. This heuristic strategy was used in the experiments.

As a comparision, we have also provided results from

solving directly the normal equations in (1.3) using SPARSPAK.

## Method of normal equations

There are four distinct phases in the method of normal equations:

(1) Ordering phase -- this determines a symmetric permutation $P$ for the matrix $A^TA$ via a minimum degree algorithm.

(2) Storage allocation phase -- after the permutation (or ordering) is determined, the data structure required to store the non-zeros is determined and set up.

(3) Numerically compute $A^TA$. It is assumed that the rows of $A$ reside on an external file. Thus the time reported for this phase includes some I/O overhead.

(4) Solution phase -- this factors the matrix $P^TA^TAP$ and solve for the solution $x$.

The storage and time reported in each phase in Tables 1(a), 1(b), 2, 3(a) and 3(b) are respectively the minimal storage and the processor time required to successfully execute that phase. Note that in phases 1 and 2, only the non-zero structure of $A$ is required. The numerical values of $A$ are used in the last two phases.

Table 1(a) Problem set 1

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 219 | 958 | 331 | 608 | 313 |
| No. of columns | 85 | 292 | 104 | 188 | 176 |
| No. of non-zeros | 438 | 1916 | 662 | 1216 | 1557 |
| Ordering phase | | | | | |
| storage | 1642 | 6461 | 2261 | 4125 | 6821 |
| time | 0.113 | 0.563 | 0.157 | 0.370 | 0.857 |
| Storage allocation phase | | | | | |
| storage | 1418 | 5560 | 1880 | 3558 | 4573 |
| time | 0.020 | 0.067 | 0.023 | 0.043 | 0.057 |
| Time to compute $A^TA$ | 0.513 | 2.303 | 0.877 | 1.460 | 1.500 |
| Solution phase | | | | | |
| storage | 1490 | 6264 | 1995 | 3969 | 3547 |
| fact. time | 0.037 | 0.260 | 0.067 | 0.153 | 0.140 |
| soln. time | 0.010 | 0.067 | 0.023 | 0.043 | 0.040 |
| Total time | 0.693 | 3.260 | 1.147 | 2.069 | 2.594 |
| Max. rel. error | 1.2E-5 | 1.4E-5 | 1.6E-5 | 2.2E-5 | 9.5E-4 |

Table 1(b) Problem set 1

| | | | | |
|---|---|---|---|---|
| No. of rows | 1033 | 1033 | 1850 | 1850 |
| No. of columns | 321 | 321 | 713 | 713 |
| No. of non-zeros | 5765 | 5765 | 10608 | 10608 |
| Ordering phase | | | | |
| storage | 11478 | 11478 | 26094 | 26094 |
| time | 19.233 | 18.710 | 41.186 | 40.790 |
| Storage allocation phase | | | | |
| storage | 7906 | 7906 | 19311 | 19311 |
| time | 0.107 | 0.107 | 0.277 | 0.260 |
| Time to compute $A^TA$ | 5.490 | 5.547 | 10.057 | 10.013 |
| Solution phase | | | | |
| storage | 6513 | 6513 | 17646 | 17646 |
| fact. time | 0.273 | 0.293 | 0.933 | 0.947 |
| soln. time | 0.073 | 0.077 | 0.197 | 0.207 |
| Total time | 25.176 | 24.734 | 52.650 | 52.217 |
| Max. rel. error | 1.4E-1 | 1.4E-1 | 8.1E-1 | 9.6E-1 |

Table 2 Problem set 2

| | | | | | | |
|---|---|---|---|---|---|---|
| No. of rows | 132 | 360 | 696 | 1140 | 1692 | 2352 |
| No. of columns | 72 | 186 | 352 | 570 | 840 | 1162 |
| No. of non-zeros | 624 | 1704 | 3296 | 5400 | 8016 | 11144 |
| Ordering phase | | | | | | |
| storage | 2137 | 5695 | 10913 | 17791 | 26329 | 36527 |
| time | 0.127 | 0.410 | 0.937 | 1.803 | 3.090 | 4.927 |
| Storage allocation phase | | | | | | |
| storage | 1524 | 4111 | 7979 | 13107 | 19514 | 27093 |
| time | 0.017 | 0.047 | 0.090 | 0.150 | 0.227 | 0.307 |
| Time to compute $A^TA$ | 0.573 | 1.570 | 3.073 | 4.957 | 7.373 | 10.227 |
| Solution phase | | | | | | |
| storage | 1415 | 4027 | 8138 | 13787 | 21201 | 30361 |
| fact. time | 0.057 | 0.183 | 0.433 | 0.780 | 1.357 | 2.243 |
| soln. time | 0.017 | 0.050 | 0.097 | 0.167 | 0.260 | 0.387 |
| Total time | 0.791 | 2.260 | 4.630 | 7.857 | 12.307 | 18.091 |
| Max. rel. error | 2.4E-4 | 6.9E-4 | 1.7E-3 | 9.5E-4 | 2.3E-3 | 6.2E-3 |

Table 3(a) Problem set 3

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 324 | 484 | 676 | 900 | 1156 |
| No. of columns | 100 | 144 | 196 | 256 | 324 |
| No. of non-zeros | 1296 | 1936 | 2704 | 3600 | 4624 |
| Ordering phase | | | | | |
| storage | 2269 | 3321 | 4573 | 6025 | 7677 |
| time | 0.167 | 0.257 | 0.403 | 0.550 | 0.723 |
| Storage allocation phase | | | | | |
| storage | 1931 | 2872 | 4000 | 5305 | 6765 |
| time | 0.023 | 0.033 | 0.050 | 0.070 | 0.087 |
| Time to compute $A^TA$ | 1.183 | 1.803 | 2.603 | 3.513 | 4.523 |
| Solution phase | | | | | |
| storage | 2176 | 3389 | 4915 | 6945 | 9043 |
| fact. time | 0.093 | 0.163 | 0.273 | 0.467 | 0.647 |
| soln. time | 0.023 | 0.040 | 0.060 | 0.083 | 0.110 |
| Total time | 1.489 | 2.296 | 3.389 | 4.683 | 6.086 |
| Max. rel. error | 3.6E-4 | 4.1E-4 | 9.1E-4 | 3.8E-4 | 8.8E-4 |

Table 3(b) Problem set 3

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 1444 | 1764 | 2116 | 2500 | 2916 |
| No. of columns | 400 | 484 | 576 | 676 | 784 |
| No. of non-zeros | 5776 | 7056 | 8464 | 10000 | 11664 |
| Ordering phase | | | | | |
| storage | 9529 | 11581 | 13833 | 16285 | 18937 |
| time | 0.933 | 1.240 | 1.387 | 1.647 | 1.963 |
| Storage allocation phase | | | | | |
| storage | 8442 | 10357 | 12384 | 14623 | 17061 |
| time | 0.103 | 0.137 | 0.150 | 0.180 | 0.217 |
| Time to compute $A^TA$ | 5.490 | 6.760 | 7.893 | 9.470 | 10.936 |
| Solution phase | | | | | |
| storage | 11706 | 14885 | 18254 | 21898 | 25962 |
| fact. time | 0.900 | 1.273 | 1.670 | 2.217 | 2.597 |
| soln. time | 0.147 | 0.190 | 0.233 | 0.287 | 0.343 |
| Total time | 7.573 | 9.600 | 11.333 | 13.711 | 16.056 |
| Max. rel. error | 1.9E-4 | 4.1E-4 | 1.5E-3 | 4.1E-4 | 2.0E-4 |

## The George-Heath algorithm

There are also four phases in the G-H algorithm, and they are similar to those in the method of normal equations:

(1) Ordering phase -- this determines a symmetric permutation P for the matrix $A^TA$ via a minimum degree algorithm.

(2) Storage allocation phase -- after the permutation P is determined, the data structure for the non-zeros is determined and set up.

(3) Sort the rows of AP in increasing order of last column subscripts. As in the method of normal equations, the rows of A are assumed to reside on an external file and must be read before sorting can be performed. Thus, the time reported also includes some I/O overhead.

(4) Solution phase -- the rows of AP are rotated into R using Givens rotations and the solution x is computed.

As in the previous case, the storage and time reported in each phase in Tables 4(a), 4(b), 5, 6(a) and 6(b) are respectively the minimal storage and the processor time required to successfully execute each phase. As in the method of normal equations, only the non-zero structure of A is required in phases 1 and 2.

Table 4(a) Problem set 1

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 219 | 958 | 331 | 608 | 313 |
| No. of columns | 85 | 292 | 104 | 188 | 176 |
| No. of non-zeros | 438 | 1916 | 662 | 1216 | 1557 |
| Ordering phase | | | | | |
| storage | 1642 | 6461 | 2261 | 4125 | 6821 |
| time | 0.120 | 0.587 | 0.160 | 0.380 | 0.887 |
| Storage allocation phase | | | | | |
| storage | 1418 | 5560 | 1880 | 3558 | 4573 |
| time | 0.020 | 0.067 | 0.020 | 0.040 | 0.057 |
| Time to sort rows of AP | 0.473 | 3.270 | 0.853 | 1.623 | 0.793 |
| Solution phase | | | | | |
| storage | 1490 | 6264 | 1995 | 3969 | 3547 |
| fact. time | 0.813 | 5.090 | 1.590 | 3.297 | 1.730 |
| soln. time | 0.010 | 0.037 | 0.013 | 0.023 | 0.027 |
| Total time | 1.436 | 9.051 | 2.636 | 5.363 | 3.494 |
| Max. rel. error | 1.9E-5 | 2.2E-5 | 1.7E-5 | 2.2E-5 | 4.1E-5 |

Table 4(b) Problem set 1

| | | | | |
|---|---|---|---|---|
| No. of rows | 1033 | 1033 | 1850 | 1850 |
| No. of columns | 321 | 321 | 713 | 713 |
| No. of non-zeros | 5765 | 5765 | 10608 | 10608 |
| Ordering phase | | | | |
| storage | 11478 | 11478 | 26094 | 26094 |
| time | 18.346 | 18.423 | 40.816 | 40.860 |
| Storage allocation phase | | | | |
| storage | 7906 | 7906 | 19311 | 19311 |
| time | 0.103 | 0.103 | 0.260 | 0.263 |
| Time to sort rows of AP | 2.470 | 2.453 | 7.413 | 7.470 |
| Solution phase | | | | |
| storage | 6513 | 6513 | 17646 | 17646 |
| fact. time | 8.003 | 7.987 | 27.673 | 27.916 |
| soln. time | 0.040 | 0.040 | 0.110 | 0.103 |
| Total time | 28.962 | 29.006 | 76.272 | 76.612 |
| Max. rel. error | 4.5E-2 | 3.3E-2 | 4.3E-2 | 4.7E-2 |

Table 5 Problem set 2

| | | | | | | |
|---|---|---|---|---|---|---|
| No. of rows | 132 | 360 | 696 | 1140 | 1692 | 2352 |
| No. of columns | 72 | 186 | 352 | 570 | 840 | 1162 |
| No. of non-zeros | 624 | 1704 | 3296 | 5400 | 8016 | 11144 |
| Ordering phase | | | | | | |
| storage | 2137 | 5695 | 10913 | 17791 | 26329 | 36527 |
| time | 0.130 | 0.447 | 0.963 | 1.873 | 3.073 | 5.113 |
| Storage allocation phase | | | | | | |
| storage | 1524 | 4111 | 7979 | 13107 | 19514 | 27093 |
| time | 0.020 | 0.047 | 0.093 | 0.157 | 0.223 | 0.313 |
| Time to sort rows of AP | 0.297 | 0.940 | 2.140 | 4.350 | 8.437 | 13.783 |
| Solution phase | | | | | | |
| storage | 1415 | 4027 | 8138 | 13787 | 21201 | 30361 |
| fact. time | 0.783 | 2.463 | 5.453 | 10.377 | 19.010 | 37.856 |
| soln. time | 0.010 | 0.027 | 0.057 | 0.090 | 0.133 | 0.207 |
| Total time | 1.240 | 3.924 | 8.706 | 16.847 | 30.876 | 57.272 |
| Max. rel. error | 1.2E-4 | 3.2E-4 | 6.4E-4 | 1.1E-3 | 1.5E-3 | 3.7E-3 |

Table 6(a) Problem set 3

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 324 | 484 | 676 | 900 | 1156 |
| No. of columns | 100 | 144 | 196 | 256 | 324 |
| No. of non-zeros | 1296 | 1936 | 2704 | 3600 | 4624 |
| Ordering phase | | | | | |
| storage | 2269 | 3321 | 4573 | 6025 | 7677 |
| time | 0.177 | 0.273 | 0.377 | 0.547 | 0.690 |
| Storage allocation phase | | | | | |
| storage | 1931 | 2872 | 4000 | 5305 | 6765 |
| time | 0.023 | 0.040 | 0.050 | 0.070 | 0.080 |
| Time to sort rows of AP | 0.837 | 1.333 | 1.980 | 3.017 | 4.287 |
| Solution phase | | | | | |
| storage | 2176 | 3389 | 4915 | 6945 | 9043 |
| fact. time | 2.293 | 3.810 | 6.087 | 10.683 | 14.680 |
| soln. time | 0.017 | 0.023 | 0.030 | 0.047 | 0.063 |
| Total time | 3.347 | 5.479 | 8.524 | 14.364 | 19.800 |
| Max. rel. error | 3.6E-4 | 6.5E-4 | 4.8E-4 | 4.0E-4 | 3.4E-4 |

Table 6(b) Problem set 3

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 1444 | 1764 | 2116 | 2500 | 2916 |
| No. of columns | 400 | 484 | 576 | 676 | 784 |
| No. of non-zeros | 5776 | 7056 | 8464 | 10000 | 11664 |
| Ordering phase | | | | | |
| storage | 9529 | 11581 | 13833 | 16285 | 18937 |
| time | 0.907 | 1.157 | 1.413 | 1.653 | 2.040 |
| Storage allocation phase | | | | | |
| storage | 8442 | 10357 | 12384 | 14623 | 17061 |
| time | 0.103 | 0.127 | 0.157 | 0.180 | 0.227 |
| Time to sort rows of AP | 6.420 | 8.497 | 10.900 | 15.063 | 19.510 |
| Solution phase | | | | | |
| storage | 11706 | 14885 | 18254 | 21898 | 25962 |
| fact. time | 20.213 | 29.173 | 39.853 | 47.199 | 56.769 |
| soln. time | 0.077 | 0.100 | 0.120 | 0.146 | 0.173 |
| Total time | 27.720 | 39.054 | 52.543 | 64.241 | 78.719 |
| Max. rel. error | 2.7E-4 | 1.2E-3 | 3.3E-4 | 1.4E-3 | 2.4E-4 |

## The Peters-Wilkinson algorithm

The P-W algorithm is more complicated than the method of normal equations and the G-H algorithm in terms of implementation. There are six phases:

(1) Initial LU decomposition -- this computes an LU-decomposition of A as in equation (2.1).

(2) Ordering phase -- this determines a symmetric permutation S for the matrix $L^TL$ using a minimum degree algorithm.

(3) Storage allocation phase -- after the symmetric permutation is determined, the data structure for storing the non-zeros in $S^TL^TLS$ is determined and set up.

(4) Numerically compute the matrix $S^TL^TLS$.

(5) Solution phase -- the matrix $S^TL^TLS$ is factored and the solution to the normal equations in (2.5) is computed.

(6) Back subsitution phase -- the solution to the orginal least squares problem is finally computed using the upper triangular matrix obtained in the first phase.

As in the previous cases, the storage and time reported in each phase in Tables 7(a), 7(b), 8, 9(a) and 9(b) reflect the minimal storage and processor time required to successfully execute that phase. Note that the

numerical values of A are required only in the first phase. In phases 2 and 3, only the non-zero structure on L is needed. Thus the storage required for the initial LU-decomposition can be released by writing the LU decomposition onto an external file and reading back when they are needed in phases 4 and 6. This means that the storage reported in the solution phase only includes the storage required for $L^TL$; it does not include any storage required for the upper triangular matrix U which is obtained in the initial LU-decomposition. The matrix U remains on external storage. It is read back in the back substitution phase. Thus, the times reported in phases 4 and 6 include some I/O overhead as well.

Table 7(a) Problem set 1

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 219 | 958 | 331 | 608 | 313 |
| No. of columns | 85 | 292 | 104 | 188 | 176 |
| No. of non-zeros | 438 | 1916 | 662 | 1216 | 1557 |
| Initial LU decomposition | | | | | |
| storage | 4209 | 18254 | 6318 | 11596 | 9252 |
| time | 0.280 | 1.217 | 0.303 | 0.767 | 0.847 |
| Ordering phase | | | | | |
| storage | 1670 | 6473 | 2261 | 4141 | 7145 |
| time | 0.130 | 0.593 | 0.197 | 0.403 | 0.897 |
| Storage allocation phase | | | | | |
| storage | 1423 | 5581 | 1894 | 3585 | 4829 |
| time | 0.017 | 0.070 | 0.027 | 0.043 | 0.063 |
| Time to compute $L^TL$ | 0.550 | 2.343 | 0.873 | 1.467 | 1.473 |
| Solution phase | | | | | |
| storage | 1483 | 6352 | 2015 | 4005 | 3762 |
| fact. time | 0.040 | 0.287 | 0.067 | 0.153 | 0.153 |
| soln. time | 0.013 | 0.067 | 0.020 | 0.037 | 0.043 |
| Back substitution phase | | | | | |
| storage | 801 | 3105 | 1087 | 1987 | 1861 |
| time | 0.003 | 0.007 | 0.003 | 0.007 | 0.010 |
| Total time | 1.033 | 4.584 | 1.490 | 2.877 | 3.486 |
| Max. rel. error | 1.0E-5 | 3.4E-5 | 1.2E-5 | 3.1E-5 | 1.7E-3 |

Table 7(b) Problem set 1

| | | | | |
|---|---|---|---|---|
| No. of rows | 1033 | 1033 | 1850 | 1850 |
| No. of columns | 321 | 321 | 713 | 713 |
| No. of non-zeros | 5765 | 5765 | 10608 | 10608 |
| **Initial LU decomposition** | | | | |
| storage | 32764 | 32759 | 59254 | 59127 |
| time | 11.070 | 11.050 | 34.710 | 34.433 |
| **Ordering phase** | | | | |
| storage | 12470 | 12346 | 29674 | 29694 |
| time | 17.900 | 18.390 | 40.076 | 39.289 |
| **Storage allocation phase** | | | | |
| storage | 8499 | 8413 | 21169 | 21123 |
| time | 0.110 | 0.110 | 0.293 | 0.290 |
| Time to compute $L^TL$ | 5.640 | 5.690 | 10.260 | 10.217 |
| **Solution phase** | | | | |
| storage | 6611 | 6567 | 17815 | 17743 |
| fact. time | 0.283 | 0.280 | 0.987 | 0.960 |
| soln. time | 0.073 | 0.070 | 0.203 | 0.197 |
| **Back substitution phase** | | | | |
| storage | 5699 | 5679 | 11075 | 11039 |
| time | 0.020 | 0.020 | 0.043 | 0.040 |
| Total time | 35.096 | 35.610 | 86.572 | 85.426 |
| Max. rel. error | 1.7E-1 | 3.8E-1 | 4.9E-1 | 8.8E-1 |

Table 8 Problem set 2

| | | | | | | |
|---|---|---|---|---|---|---|
| No. of rows | 132 | 360 | 696 | 1140 | 1692 | 2352 |
| No. of columns | 72 | 186 | 352 | 570 | 840 | 1162 |
| No. of non-zeros | 624 | 1704 | 3296 | 5400 | 8016 | 11144 |
| Initial LU decomposition | | | | | | |
| storage | 4253 | 10793 | 20751 | 33979 | 50223 | 69855 |
| time | 0.740 | 2.417 | 7.823 | 18.657 | 40.226 | 73.016 |
| Ordering phase | | | | | | |
| storage | 2809 | 6599 | 12633 | 20607 | 30209 | 41927 |
| time | 0.213 | 0.567 | 1.217 | 2.303 | 3.723 | 6.330 |
| Storage allocation phase | | | | | | |
| storage | 1836 | 4626 | 8900 | 14569 | 21525 | 29906 |
| time | 0.023 | 0.053 | 0.107 | 0.180 | 0.263 | 0.373 |
| Time to compute $L^TL$ | 0.830 | 1.900 | 3.630 | 6.177 | 9.167 | 12.746 |
| Solution phase | | | | | | |
| storage | 1437 | 4238 | 8277 | 14013 | 21456 | 30576 |
| fact. time | 0.067 | 0.207 | 0.430 | 0.817 | 1.490 | 2.223 |
| soln. time | 0.017 | 0.053 | 0.097 | 0.173 | 0.280 | 0.390 |
| Back subsitution phase | | | | | | |
| storage | 893 | 1985 | 3747 | 6065 | 8937 | 12289 |
| time | 0.007 | 0.007 | 0.013 | 0.023 | 0.033 | 0.043 |
| Total time | 1.897 | 5.204 | 13.317 | 28.330 | 55.185 | 95.121 |
| Max. rel. error | 1.7E-2 | 2.1E-3 | 8.7E-3 | 5.7E-3 | 2.5E-2 | 2.3E-2 |

Table 9(a) Problem set 3

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 324 | 484 | 676 | 900 | 1156 |
| No. of columns | 100 | 144 | 196 | 256 | 324 |
| No. of non-zeros | 1296 | 1936 | 2704 | 3600 | 4624 |
| Initial LU decomposition | | | | | |
| storage | 8121 | 12121 | 16921 | 22521 | 28921 |
| time | 0.447 | 0.660 | 0.923 | 1.273 | 1.587 |
| Ordering phase | | | | | |
| storage | 2277 | 3329 | 4581 | 6033 | 7685 |
| time | 0.163 | 0.260 | 0.373 | 0.517 | 0.687 |
| Storage allocation phase | | | | | |
| storage | 1948 | 2864 | 3976 | 5291 | 6764 |
| time | 0.023 | 0.033 | 0.047 | 0.063 | 0.083 |
| Time to compute $L^TL$ | 1.163 | 1.760 | 2.520 | 3.270 | 4.257 |
| Solution phase | | | | | |
| storage | 2171 | 3458 | 4886 | 6854 | 9210 |
| fact. time | 0.083 | 0.173 | 0.257 | 0.430 | 0.670 |
| soln. time | 0.023 | 0.040 | 0.053 | 0.083 | 0.117 |
| Back subsitution phase | | | | | |
| storage | 1107 | 1611 | 2211 | 2907 | 3699 |
| time | 0.003 | 0.007 | 0.007 | 0.007 | 0.010 |
| Total time | 1.905 | 2.933 | 4.180 | 5.643 | 7.411 |
| Max. rel. error | 5.0E-4 | 4.5E-4 | 2.6E-4 | 9.7E-4 | 9.8E-5 |

Table 9(b) Problem set 3

| | | | | | |
|---|---|---|---|---|---|
| No. of rows | 1444 | 1764 | 2116 | 2500 | 2916 |
| No. of columns | 400 | 484 | 576 | 676 | 784 |
| No. of non-zeros | 5776 | 7056 | 8464 | 10000 | 11664 |
| Initial LU decomposition | | | | | |
| storage | 36121 | 44121 | 52921 | 62521 | 72929 |
| time | 2.030 | 2.470 | 2.957 | 3.510 | 5.030 |
| Ordering phase | | | | | |
| storage | 9537 | 11589 | 13841 | 16293 | 18953 |
| time | 0.900 | 1.107 | 1.463 | 1.640 | 1.937 |
| Storage allocation phase | | | | | |
| storage | 8428 | 10248 | 12354 | 14592 | 17040 |
| time | 0.107 | 0.127 | 0.160 | 0.193 | 0.217 |
| Time to compute $L^TL$ | 5.320 | 6.627 | 8.263 | 9.523 | 11.140 |
| Solution phase | | | | | |
| storage | 11650 | 14428 | 18193 | 21899 | 26229 |
| fact. time | 0.883 | 1.160 | 1.833 | 2.130 | 2.917 |
| soln. time | 0.143 | 0.183 | 0.247 | 0.283 | 0.357 |
| Back subsitution phase | | | | | |
| storage | 4587 | 5571 | 6651 | 7827 | 9103 |
| time | 0.013 | 0.013 | 0.020 | 0.017 | 0.020 |
| Total time | 9.396 | 11.687 | 14.943 | 17.296 | 21.618 |
| Max. rel. error | 4.1E-4 | 5.1E-4 | 9.9E-4 | 3.4E-4 | 9.9E-4 |

The following table is a summary of the numerical experiments. The columns labelled "max. store" and "total time" in each algorithm represent the maximum storage and total time (in seconds) required to successfully execute the whole algorithm.

Table 10 Summary

| No. of rows | No. of cols | No. of non-zeros | normal equations | | G-H algorithm | | P-W algorithm | |
|---|---|---|---|---|---|---|---|---|
| | | | max. store | total time | max. store | total time | max. store | total time |
| 219 | 85 | 438 | 1642 | 0.693 | 1642 | 1.436 | 4209 | 1.033 |
| 958 | 292 | 1916 | 6461 | 3.260 | 6461 | 9.051 | 18254 | 4.584 |
| 331 | 104 | 662 | 2261 | 1.147 | 2261 | 2.636 | 6318 | 1.490 |
| 608 | 188 | 1216 | 4125 | 2.069 | 4125 | 5.363 | 11596 | 2.877 |
| 313 | 176 | 1557 | 6821 | 2.594 | 6821 | 3.494 | 9252 | 3.486 |
| 1033 | 321 | 5765 | 11478 | 25.176 | 11478 | 28.962 | 32764 | 35.096 |
| 1033 | 321 | 5765 | 11478 | 24.734 | 11478 | 29.006 | 32759 | 35.610 |
| 1850 | 713 | 10608 | 26094 | 52.650 | 26094 | 76.272 | 59254 | 86.572 |
| 1850 | 713 | 10608 | 26094 | 52.217 | 26094 | 76.612 | 59127 | 85.426 |
| 132 | 72 | 624 | 2137 | 0.791 | 2137 | 1.240 | 4253 | 1.897 |
| 360 | 186 | 1704 | 5695 | 2.260 | 5695 | 3.924 | 10793 | 5.204 |
| 696 | 352 | 3296 | 10913 | 4.630 | 10913 | 8.706 | 20751 | 13.317 |
| 1140 | 570 | 5400 | 17791 | 7.857 | 17791 | 16.847 | 33979 | 28.330 |
| 1692 | 840 | 8016 | 26329 | 12.307 | 26329 | 30.876 | 50223 | 55.185 |
| 2352 | 1162 | 11144 | 36527 | 18.091 | 36527 | 57.272 | 69855 | 95.121 |
| 324 | 100 | 1296 | 2269 | 1.489 | 2269 | 3.347 | 8121 | 1.905 |
| 484 | 144 | 1936 | 3389 | 2.296 | 3389 | 5.479 | 12121 | 2.933 |
| 676 | 196 | 2704 | 4915 | 3.389 | 4915 | 8.524 | 16921 | 4.180 |
| 900 | 256 | 3600 | 6945 | 4.683 | 6945 | 14.364 | 22521 | 5.643 |
| 1156 | 324 | 4624 | 9043 | 6.086 | 9043 | 19.800 | 28921 | 7.411 |
| 1444 | 400 | 5776 | 11706 | 7.573 | 11706 | 27.720 | 36121 | 9.396 |
| 1764 | 484 | 7056 | 14885 | 9.600 | 14885 | 39.054 | 44121 | 11.687 |
| 2116 | 576 | 8464 | 18254 | 11.333 | 18254 | 52.543 | 52921 | 14.943 |
| 2500 | 676 | 10000 | 21898 | 13.711 | 21898 | 64.241 | 62521 | 17.296 |
| 2916 | 784 | 11664 | 25962 | 16.056 | 25962 | 78.719 | 72929 | 21.618 |
| average ... | | | | | | | | |
| 1154 | 402 | 5034 | 12605 | 11.468 | 12605 | 26.620 | 31224 | 22.090 |

## V. Concluding remarks

Following are some observations about the experiments.

(1) The storage requirements for the normal equations and the G-H algorithm are the same. This is not surprising since the basic steps in these two algorithms are the same. Recall that the Givens rotations used in the G-H algorithm are not saved.

(2) The storage requirements for the method of normal equations and the G-H algorithm are better than the P-W scheme. The minimal storage required to successfully execute the P-W algorithm can be 2 to 3 times that required for the other two methods.

(3) The method of normal equations executes much faster than the other two methods.

(4) Whether the P-W algorithm or the G-H algorithm is faster seems to be problem-dependent.

(5) In the P-W algorithm, the most expensive phases are the computations of the LU-decomposition of A and normal equations. The storage requirement and the execution time are large in the initial LU-decomposition phase. On the other hand, the most expensive phases in the G-H algorithm are the sorting and reduction phases.

(6) Among the test problems we have tried, the method of normal equations is apparently the most accurate method, except for some test problems in the first set. (See Tables 1(b), 4(b) and 7(b)). However, if we

ignore the normal equations, then the G-H algorithm gives, in general, more accurate results than the P-W algorithm.

(7) It is interesting to note that the storage required to determine the non-zero structure and store the non-zeros of the Cholesky factors of the normal equations in (1.3) and (2.5) are very close. This suggests that $A^TA$ and $L^TL$ may have similar structure.

The above observations suggest that if numerical stability is not important, the method of normal equations should be seriously considered because of its small storage requirement and execution time. However, since there exist problems in which the normal equations will fail due to numerical difficulties (for example, consider the first set of test problems), the G-H algorithm should be used in those situations if the available storage is restricted and time is not a major issue.

Note that it is assumed in the method of normal equations and the G-H algorithm that if the matrix A is sparse, then the matrix product $A^TA$ is also sparse. If it is not the case, these two methods are likely to be inefficient. However in most of the latter cases, this phenomenon is due to the presence of a few relatively dense rows in A. George and Heath have proposed a solution in [4]. Those dense rows are temporarily discarded, yielding a problem in which $A^TA$ is sparse. After the solution x to

the modified least squares problem is determined, the solution to the original problem can be obtained by modifying x using those discarded rows (see section 5 of [4] for details).

## VI. References


[1] A. Björck and I.S. Duff, "A direct method for the solution of sparse linear least squares problems", to appear in Linear Algebra and Its Applications (1980).

[2] I.S. Duff, "MA28 - a set of FORTRAN subroutines for sparse unsymmetric linear equations", Report AERE R-8730, July 1977.

[3] A. George, J. Liu and E. Ng, "User's guide for SPARSPAK: Waterloo sparse linear equations package", Research Report CS-78-30 (revised, 1980), Dept. of Computer Science, University of Waterloo.

[4] A. George and M.T. Heath, "Solution of sparse linear least squares problems using Givens rotations", to appear in Linear Algebra and Its Applications (1980).

[5] A. George and J. Liu, Computer Solution of Large Sparse Positive Definite Systems, to be published by Prentice-Hall, Inc. (1980)

[6] C. Lawson and R. Hanson, Solving Least Squares Problems, Prentice-Hall, Inc. (1974).

[7] H.M. Markowitz, "The elimination form of the inverse and its application to linear programming", Management Sci. 3 (1957), pp. 255-269.

[8] G. Peters and J.H. Wilkinson, "The least squares problems and pseudo-inverses", Comput. J., 13 (1970), pp. 309-316.

[9] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, 1965.