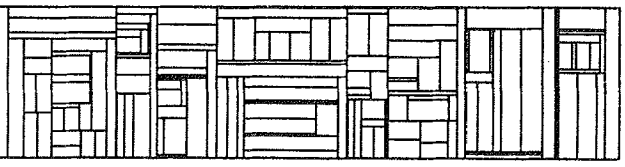
ISSN 0105-8517

Comparison of two pivotal strategies in sparse plane rotations

by
Zahari Zlatev

DAIMI PB-122
July 1980

Computer Science Department
AARHUS UNIVERSITY
Ny Munkegade – DK 8000 Aarhus C – DENMARK
Telephone: 06 – 12 83 55



COMPARISON OF TWO PIVOTAL STRATEGIES IN SPARSE PLANE ROTATIONS

by

Zahari Zlatev

Computer Science Department

Aarhus University

Aarhus, Denmark

ABSTRACT Let the rectangular matrix A be large and sparse. Assume that plane rotations are used to decompose A into RDS where $R^TR=I$, D is diagonal and S is upper triangular. Both column and row interchanges have to be used in order to preserve the sparsity of matrix A during the decomposition. It is proved that if the column interchanges are fixed, then the number of non-zero elements in S does not depend on the row interchanges used. However,this does not mean that the computational work is also independent of the row interchanges. Two pivotal strategies, where the same rule is used in the choice of pivotal columns, are described and compared. It is verified (by many numerical examples) that if matrix A is not very sparse, then one of these strategies will often perform better than the other both with regard to the storage and the computing time. The accuracy and the robustness of the computations are also discussed.

1.Statement of the problem

Let $m \in \mathbb{N}$, $n \in \mathbb{N}$, $b \in \mathbb{R}^{mx1}$ and $A \in \mathbb{R}^{mxn}$ be given. Asume that: (i) $m \geq n$, (ii) $rank(A)=n$, (iii) n is large (say, $n \geq 100$) and (iv) A is sparse (i.e. many elements of A are equal to zero). Consider the decomposition

$$(1.1) \qquad RDS = PAQ + E$$

where $P \in \mathbb{R}^{mxm}$ and $Q \in \mathbb{R}^{nxn}$ are permutation matrices, $E \in \mathbb{R}^{mxn}$ is a perturbation matrix, $R \in \mathbb{R}^{mxn}$ is such that $R^T R = I \in \mathbb{R}^{nxn}$, $D \in \mathbb{R}^{nxn}$ is a diagonal matrix and $S \in \mathbb{R}^{nxn}$ is an upper triangular matrix.

Denote $A_1 = A$ and assume for the moment that the diagonal matrix $D_1 \in \mathbb{R}^{mxm}$ is given. The decomposition (1.1) is often carried out by the method of plane rotations in n major steps. During the k'th major step $(k=1(1)n)$ the product $D_k A_k$ (where $D_k \in \mathbb{R}^{mxm}$ is diagonal and $A_k \in \mathbb{R}^{mxn}$ contains only zero elements under the diagonal in its first $k-1$ columns for $k > 1$) is transformed into $D_{k+1} A_{k+1}$ (where $D_{k+1} \in \mathbb{R}^{mxm}$ is diagonal and $A_{k+1} \in \mathbb{R}^{mxn}$ contains only zero elements under the diagonal in its first k columns) by the following algorithm:

(i) Choose the pivotal column j $(k \leq j \leq n)$ and interchange column j and column k if $j \neq k$. Denote the matrix so found by $\bar{A}_k$.

(ii) Assume that the k'th column in $\bar{A}_k$ has s_k+1 non-zero elements on and/or under the diagonal. Then s_k multiplications with elementary orthogonal matrices are performed when $s_k > 0$ so that after each multiplication one zero element in the pivotal column is produced (on or under the diagonal).

(iii) Let the i'th row ($k \leq i \leq m$) be the only row which contains a non-zero element in the pivotal column after the last multiplication with an elementary orthogonal matrix during the major step k. Interchange row i and row k if $i \neq k$.

When the computations in the n'th major step are completed, the product $D_{n+1}A_{n+1}$ will be available. It is clear that the first n rows of A_{n+1} form matrix S and matrix D from (1.1) is the upper left (nxn) - submatrix of D_{n+1}.

Multiplication by an elementary orthogonal matrix is often called a minor step and is carried out as follows. Consider the k'th major step again. Assume that (i) $\mu \geq k$, $\nu \geq k$, $\mu \neq \nu$ and (ii) $a_{\mu k} \neq 0$, $a_{\nu k} \neq 0$. Then the elementary orthogonal matrix $O_{\mu\nu}$ (which differs from the identity matrix $I \in \mathbb{R}^{mxm}$ only by $o_{\mu\mu} = o_{\nu\nu} = \gamma$ and $o_{\mu\nu} = -o_{\nu\mu} = \sigma$ with $\gamma^2 + \sigma^2 = 1$) is used to transform into zero one of $a_{\mu k}$ and $a_{\nu k}$. It is readily seen that in fact the multiplication

$$(1.2) \qquad \begin{bmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{bmatrix} \begin{bmatrix} d_\mu & 0 \\ 0 & d_\nu \end{bmatrix} \begin{bmatrix} a_{\mu k} & \cdots & a_{\mu n} \\ a_{\nu k} & \cdots & a_{\nu n} \end{bmatrix}$$

has to be performed (when n is large and A is sparse many of the elements of the last matrix in (1.2) are normally zero elements; therefore some sparse matrix technique, where only the non-zero elements are stored and used in the arithmetic operations, has to be implemented). In the classical Givens method [12,13] $D_k = I \in \mathbb{R}^{mxm}$ (k=1(1)n) and the multiplication (1.2) is carried out directly, i.e. the new elements are computed by

$$(1.3) \qquad \bar{a}_{\mu j} = \gamma a_{\mu j} + \sigma a_{\nu j}, \quad \bar{a}_{\nu j} = -\sigma a_{\mu j} + \gamma a_{\nu j}, \quad j = k(1)n.$$

This means that 2 multiplications are needed in the com-

putation of each new element. If $D_k=I$ $(k=1(1)n)$ is not required, then the first two matrices in (1.2) can be refactorized, e.g. in the following way:

$$(1.4)\qquad \begin{bmatrix}\gamma & \sigma\\ \sigma & \gamma\end{bmatrix}\begin{bmatrix}d_\mu & 0\\ 0 & d_\nu\end{bmatrix} = \begin{bmatrix}d_\mu\gamma & 0\\ 0 & d_\nu\gamma\end{bmatrix}\begin{bmatrix}1 & \alpha\\ \beta & 1\end{bmatrix}.$$

Then the multiplication

$$(1.5)\qquad \begin{bmatrix}1 & \alpha\\ \beta & 1\end{bmatrix}\begin{bmatrix}a_{\mu k} & \cdots & a_{\mu n}\\ a_{\nu k} & \cdots & a_{\nu n}\end{bmatrix}$$

is carried out and the new elements are computed by the formulae:

$$(1.6)\qquad \bar{a}_{\mu j}=a_{\mu j}+\alpha a_{\nu j},\qquad \bar{a}_{\nu j}=\beta a_{\mu j}+a_{\nu j},\qquad j=k(1)n.$$

It is clear that only one multiplication per new element is needed when (1.6) are used instead of (1.3). The refactorization (1.4) is proposed by Gentleman [9] (see also [16,11]). This is not the only possible refactorization of the left-hand side of (1.4). Some other refactorizations are given by Wilkinson [25], pp.12-13. In this way Wilkinson has shown that both the classical Givens method and the Gentleman version can be found as special cases in the generalized plane rotations with a suitable choice of the matrices D_k $(k=1(1)n)$. The refactorization (1.4) will be used in this paper.

The following algorithm can be used in the computation of α, β and γ (see also Lawson et al. [17], Björck [5] and Zlatev and Nielsen [30]).

If

$$(1.7) \qquad d_\mu^2 a_{\mu k}^2 \geq d_\nu^2 a_{\nu k}^2$$

then produce a zero element in row ν by

$$(1.8) \qquad \beta = -a_{\nu k}/a_{\mu k}, \qquad \alpha = -\beta d_\nu^2/d_\mu^2 .$$

If

$$(1.9) \qquad d_\mu^2 a_{\mu k}^2 < d_\nu^2 a_{\nu k}^2$$

then produce a zero element in row μ by

$$(1.10) \qquad \alpha = -a_{\mu k}/a_{\nu k}, \qquad \beta = -\alpha d_\mu^2/d_\nu^2 .$$

In both cases

$$(1.11) \qquad \gamma^2 = 1/(1-\alpha\beta)$$

and the new elements of the diagonal matrix D_k involved in this minor step are computed by

$$(1.12) \qquad \bar{d}_\mu^2 = \gamma^2 d_\mu^2, \qquad \bar{d}_\nu^2 = \gamma^2 d_\nu^2 .$$

Remark 1.1 It is easily seen that the above algorithm ensures that the squares of the diagonal elements of D_k $(k=1(1)n)$ are decreased by a factor $\delta_{ki} \in [\frac{1}{2},1)$ after the i'th minor step $(i=1(1)s_k)$ of major step k .

Remark 1.2 If D_k^2 $(k=1(1)n+1)$ are stored instead of D_k,

then no square root is needed.

Remark 1.3 If the problem under consideration is not weighted, then $D_1=I$ can be chosen. If the problem is weighted, then D_1^2 must have diagonal elements equal to the squares of the weights.

If matrix A is sparse, then a good choice of P and Q (in other words, a good pivotal strategy) will normally lead to a considerable reduction in the computing time and the storage needed. Two pivotal strategies will be considered in this paper. These pivotal strategies can be defined as follows.

Definition 1.1 The set

$$\bar{C}_{kj} = \{a_{ij} \;/\; i=k(1)m\} \tag{1.13}$$

is called the active part of column j $(j=k(1)n)$ at major step k $(k(1(1)n)$.

Rule 1 At any major step k $(k=1(1)n)$ the column which contains a minimal number of non-zero elements in its active part (or one of these columns if there are several) is chosen as pivotal.

Definition 1.2 The set

$$\bar{R}_{ki} = \{a_{ij} \;/\; j=k(1)n\} \tag{1.14}$$

is called the active part of row i $(i=k(1)m)$ at major step k $(k=1(1)n)$.

Definition 1.3 Consider major step k $(k=1(1)n)$. The number of non-zero elements in row i $(i=k(1)m)$ before minor step s $(s=1(1)s_k)$ will be denoted by $r(k,s,i)$.

Rule 2 Consider major step k $(k=1(1)n)$. Choose a row i for which (1) $k \leq i \leq m$, (2) $a_{ik} \neq 0$ and (3) if μ is any row for which $k \leq \mu \leq m$ and $a_{\mu k} \neq 0$, then $r(k,1,i) \leq r(k,1,\mu)$. Consider any two rows ν and τ $(k \leq \nu, \tau \leq m, \nu \neq i, \tau \neq i)$ with $a_{\nu k} \neq 0$ and $a_{\tau k} \neq 0$. The elements $a_{\nu k}$ and $a_{\tau k}$ are transformed into zero by multiplication with the elementary orthogonal matrices $O_{i\nu}$ and $O_{i\tau}$ and, moreover, if $r(k,1,\nu) < r(k,1,\tau)$ then $a_{\nu k}$ will be transformed into zero before $a_{\tau k}$ (the multiplication with $O_{i\nu}$ will be performed before the multiplication with $O_{i\tau}$).

Rule 3 Consider major step k $(k=1(1)n)$. Before any minor step s $(s=1(1)s_k)$ choose two rows μ and ν for which (1) $k \leq \mu, \nu \leq m$, (2) $a_{\mu k} \neq 0$, $a_{\nu k} \neq 0$, (3) if τ is any row with $k \leq \tau \leq m$ and $a_{\tau k} \neq 0$, then $max(r(k,s,\mu), r(k,s,\nu)) \leq r(k,s,\tau)$. Produce a zero element in one of these two rows (multiplying by $O_{\mu\nu}$) according to the algorithm described by (1.7) - (1.11).

Definition 1.4 The pivotal strategy based on Rule 1 and Rule 3 will be called Strategy 1 or a variable pivotal row strategy.

Definition 1.5 The pivotal strategy based on Rule 1 and Rule 2 will be called Strategy 2 or a fixed pivotal row strategy.

Strategy 2 has been used by Duff [7], see also [8,21]. Strategy 1 is based on ideas suggested by Gentleman [10] and has been implemented in the sparse code described in [29,30]. A comparison of these two strategies will be carried out in this paper. The main results obtained in this comparison are:

A. If Strategy 1 and Strategy 2 are used with the same matrix Q (i.e. the column interchanges are the same), then the two strategies produce the same number of non-zero elements in matrix S. However, this does not mean that the computational work for obtaining the decomposition (1.1) will be the same; see Corollary 2.1 and Example 2.1 in Section 2.

B. If matrix A is very sparse and the sparsity is well preserved during the computations, then the two strategies will give similar results (especially when the number of non-zero elements in the active part of the pivotal column is less than 4 for each k (k=1(1)n); see Theorem 2.2).

C. If $n \geq 100$ and if the matrix is not very sparse, then Strategy 1 will often give better results than Strategy 2 with regard both to the storage used and the computing time needed; moreover, Strategy 1 ensures more robust computations; see the numerical results in Section 3.

2. Computations with a fixed matrix Q

The following definitions and theorems are needed before the discussion of the question: what should be expected when the strategies defined in Section 1 are used with the same column interchanges (the same matrix Q)?

Definition 2.1 The set

$$(2.1) \qquad B_k = \{a_{ij} \in \bar{A}_k \ / \ k \leq i \leq m \ \wedge \ k \leq j \leq n\}$$

will be called the active part of $\bar{A}_k$ (k=1(1)n).

Definition 2.2 The set

$$(2.2) \qquad L_{ki} = \{j \ / \ a_{ij} \neq 0 \ \wedge \ a_{ij} \in B_k\}$$

will be called the sparsity pattern of the active part of row i (i=k(1)m) at major step k (k=1(1)n).

Definition 2.3 The set

$$(2.3) \qquad M_{kj} = \{i \ / \ a_{ij} \neq 0 \ \wedge \ a_{ij} \in B_k\}$$

will be called the sparsity pattern of the active part of column j (j=k(1)n) at major step k (k=1(1)n).

Definition 2.4 The set

$$(2.4) \qquad N_k = \{j \ / \ a_{ij} \neq 0 \ \wedge \ a_{ij} \in B_k \ \wedge \ i \in M_{kk}\}$$

will be called the union sparsity pattern of the active parts of the rows involved in the computations at major step k $(k=1(1)n)$.

Remark 2.1 It is clear that

$$N_k = \bigcup_{i \in M_{kk}} L_{ki} \quad . \tag{2.5}$$

Definition 2.5 The set

$$N^*_{kj} = \{i \;/\; a_{i\mu} \neq 0 \;\wedge\; a_{i\mu} \in C_{kj} \;\wedge\; i \in M_{kj} \;\wedge\; k < n\} \tag{2.6}$$

where the submatrix C_{kj} is defined by

$$C_{kj} = \{a_{i\mu} \in B_k \;/\; j \leq \mu \leq n\} \tag{2.7}$$

will be called the union sparsity pattern of the rows in matrix C_{kj} which have non-zero elements in the active part of column j $(j=k+1(1)n)$.

Theorem 2.1 The number of non-zero elements in matrix S found by plane rotations does not depend on the permutation matrix P when the permutation matrix Q is kept fixed.

Proof Assume that the multiplication with the fixed matrix Q has been performed before the beginning of the computations, i.e. $A_1=AQ$ and $A_k=\bar{A}_k$ $(k=1(1)n)$. It is clear that this assumption is not a restriction.

Consider the k'th major step $(k=1(1)n)$. After the multiplication with any elementary orthogonal matrix $O_{\mu\nu}$ $(\mu \in M_{kk} \wedge \nu \in M_{kk})$ the sparsity patterns of the active parts of the rows in-

volved in the computations are given by

$$(2.8) \qquad L_{k\mu} \cup L_{k\nu} \qquad \text{and} \qquad (L_{k\mu} \cup L_{k\nu}) \setminus \{k\},$$

where the second sparsity pattern is for the row in which a zero element in the pivotal column is produced. The expressions (2.8) show that after the last minor step in the k'th major step the sparsity patterns of the active parts of the rows involved are given by

$$(2.9) \qquad \bigcup_{i \in M_{kk}} L_{ki} = N_k \qquad \text{and} \qquad \Big(\bigcup_{i \in M_{kk}} L_{ki}\Big) \setminus \{k\} = N_k \setminus \{k\} .$$

It is clear that the expressions in (2.9) do not depend on the order in which the zero elements are produced.

Consider now the sets N^*_{kj} $(j=k+1(1)n \wedge k<n)$. These sets are invariant during the computations at major step k $(k=1(1)n-1)$. Indeed, new non-zero elements may be produced in some rows i $(i \in M_{kk})$ but this happens only in the positions where there are non-zero elements in one or several of the other rows belonging to M_{kk}. Therefore $N^*_{k,k+1}=N_{k+1}$ and $N^*_{kj}=N^*_{k+1,j}$ $(j=k+2(1)n \wedge k+1<n)$.

Thus the theorem is proved because: (i) an arbitrary major step k has been considered, (ii) the number, N_k, of the non-zero elements in the k'th row of S does not depend on the order in which the zero elements in the pivotal column k are produced and (iii) the union sparsity pattern of the active parts of the rows which will be involved in the computations during the next major step, k+1'st, does not depend on the computations during the k'th major step.

□

The following two results follow immediately from Theorem 2.1.

Corollary 2.1 Assume that the same column interchanges have been used both with Strategy 1 and Strategy 2. Then the number of non-zero elements in matrix S will be the same for these two strategies.

□

Theorem 2.2 Assume that: (i) if at any major step there are several columns which have a minimal number of non-zero elements in their active parts, then the same column will be chosen as pivotal with both Strategy 1 and Strategy 2, (ii) after the column interchanges the number of elements in M_{kk} is smaller than 4 for any k (k=1(1)n-1). Then the numbers of non-zero elements in B_k found by Strategy 1 and Strategy 2 are the same for all k.

□

Corollary 2.1 states that if the same matrix Q is used, then Strategy 1 and Strategy 2 will produce matrices S with the same number of non-zero elements. This does not mean, however, that the computational work needed to obtain (1.1) is also the same. The next example illustrates this.

Example 2.1 Consider the (14x8) matrix whose sparsity pattern is given in Fig. 2.1. The decomposition (1.1) has been performed with Q=I and both Strategy 1 and Strategy 2 have been used to determine P (it is readily seen that the use of Q=I satisfies Rule 1 at each major step). The elementary orthogonal matrices which can be used at each major step are listed in Table 2.1 (the second index always shows the row where a zero element is pro-

$$\begin{bmatrix} x & x & 0 & 0 & x & 0 & 0 & 0 \\ 0 & x & 0 & x & x & 0 & 0 & 0 \\ 0 & 0 & x & 0 & 0 & 0 & x & 0 \\ 0 & 0 & 0 & x & 0 & x & 0 & 0 \\ 0 & 0 & 0 & 0 & x & 0 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & x & x & x \\ 0 & 0 & x & 0 & 0 & 0 & 0 & x \\ x & x & 0 & x & 0 & 0 & 0 & x \\ 0 & x & 0 & 0 & 0 & x & 0 & x \\ 0 & 0 & x & 0 & x & 0 & 0 & x \\ 0 & 0 & x & x & 0 & x & 0 & x \\ x & 0 & 0 & 0 & x & 0 & x & x \\ x & 0 & 0 & 0 & x & 0 & x & x \\ 0 & 0 & x & 0 & 0 & x & 0 & 0 \end{bmatrix}$$

Fig 2.1

The sparsity pattern of a (14x8) matrix (the non-zero elements are denoted by x in this figure)

duced). The numbers of minor steps and of fill-ins produced at each major step are given in Table 2.2. The total results show that even for this very small (but not very sparse) matrix the number of minor steps and fill-ins is reduced by about 7% when Strategy 1 is used and when Q is the same for both strategies. Of course, the numbers of non-zero elements in S are the same, 33, for the two strategies.

□

Corolary 2.1 and Example 2.1 indicate that even when the same matrix Q is used Strategy 1 may be more profitable. One can expect that for large matrices ($n \geq 100$) the reduction in minor steps and fill-ins will be greater than 7% . However, if the matrix is extremely sparse, then Theorem 2.2 shows that both the storage and the computing time will be the same for the two strategies when Q is the same and the sparsity is preserved well. Some ex-

	Strategy 2							Strategy 1						
k \ i	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1	$O_{1,8}$	$O_{1,12}$	$O_{1,13}$					$O_{1,8}$	$O_{12,13}$	$O_{1,12}$				
2	$O_{2,9}$	$O_{2,8}$	$O_{2,12}$	$O_{2,13}$				$O_{2,9}$	$O_{2,8}$	$O_{2,12}$				
3	$O_{3,7}$	$O_{3,14}$	$O_{3,10}$	$O_{3,11}$				$O_{3,7}$	$O_{3,14}$	$O_{3,10}$	$O_{3,11}$			
4	$O_{4,8}$	$O_{4,9}$	$O_{4,11}$	$O_{4,12}$	$O_{4,13}$			$O_{4,8}$	$O_{4,9}$	$O_{4,11}$	$O_{4,12}$			
5	$O_{5,8}$	$O_{5,9}$	$O_{5,10}$	$O_{5,11}$	$O_{5,12}$	$O_{5,13}$		$O_{5,13}$	$O_{8,9}$	$O_{5,8}$	$O_{5,10}$	$O_{5,11}$	$O_{5,12}$	
6	$O_{6,8}$	$O_{6,9}$	$O_{6,10}$	$O_{6,11}$	$O_{6,12}$	$O_{6,13}$	$O_{6,14}$	$O_{6,9}$	$O_{6,8}$	$O_{6,10}$	$O_{6,11}$	$O_{6,12}$	$O_{6,13}$	
7	$O_{7,8}$	$O_{7,9}$	$O_{7,10}$	$O_{7,11}$	$O_{7,12}$	$O_{7,13}$	$O_{7,14}$	$O_{7,8}$	$O_{7,9}$	$O_{7,10}$	$O_{7,11}$	$O_{7,12}$	$O_{7,13}$	$O_{7,14}$
8	$O_{8,9}$	$O_{8,10}$	$O_{8,11}$	$O_{8,12}$	$O_{8,13}$	$O_{8,14}$		$O_{8,9}$	$O_{8,10}$	$O_{8,11}$	$O_{8,12}$	$O_{8,13}$	$O_{8,14}$	

Table 2.1

The elementary orthogonal matrices used in the decomposition of the matrix from Fig. 2.1 ($k=1(1)8$, $i=1(1)s_k$).

Major step k	Strategy 2		Strategy 1	
	Number of minor steps	Number of fill-ins	Number of minor steps	Number of fill-ins
1	3	8	3	6
2	4	8	3	7
3	4	11	4	11
4	5	3	4	3
5	6	4	6	3
6	7	0	6	1
7	7	0	7	0
8	6	0	6	0
Total	42	34	39	31

Table 2.2

The number of minor steps and fill-ins found in the decomposition of the matrix from Fig.2.1

perimental results (Duff [7]) indicate that for very sparse matrices this is also true even when the requirement for the same Q is removed.

In the real computations the matrices will not be extremely sparse (note that in [7] there are examples where the number of non-zero elements in the matrix is smaller than the number of rows) and Q will not be the same. In the next section some numerical experiments are carried out in order to compare the two strategies in such situations for matrices with large n.

3. Numerical results

In this section a comparison of the performance of the two strategies will be carried out for some matrices with large n. Some information about the codes, the sparse technique and the test-matrices used is needed before the discusion of the numerical results which are given in Table 3.1 - Table 3.9.

3.1. The codes used The decomposition (1.1) is used for solving linear least-squares problems

$$x = A^{\dagger}b \tag{3.1}$$

where $A^{\dagger}$ is the pseudo-inverse ([18,19]) of matrix A. Assume that parallel computations with vector b are carried out during the decomposition (1.1) so that

$$b^* = R^T P b \tag{3.2}$$

is available after the n'th major step. Then an approximation x_1 to x can be found by

$$x_1 = QS^{-1}D^{-1}b^*. \tag{3.3}$$

Sometimes many problems (3.1) with the same matrix A have to be solved. In this case it is necessary to keep R, D and S. When A is dense S is stored on and over the diagonal, while information about R can be stored in an economical and stable way under the diagonal [23]. When matrix A is sparse this is not efficient because matrix R contains normally more non-

zero elements than matrix A. Therefore it is more profitable to store and keep a copy of the non-zero elements of A [3,5]. If matrices S, D and A are available, then an approximation x_1 (in general different from that computed by (3.3)) can be found from

$$x_1 = HA^T b \tag{3.4}$$

where

$$H = QS^{-1}D^{-2}(S^T)^{-1}Q^T. \tag{3.5}$$

A special parameter T (drop-tolerance, see [6,20,24,26]) can be used during (1.1) so that if an element computed by (1.3) or (1.6) is smaller (in absolute value) than T, then this element is considered as a zero element in the further computations. Both storage and computing time may be saved when positive values of T are used ([27,28,22,31]). However, the use of $T>0$ may cause large errors in the approximation x_1 also. Therefore the following iterative process $(i=1(1)p-1)$

$$r_i = b-Ax, \qquad r_i^* = A^T r_i, \tag{3.6}$$

$$d_i^* = Hr_i^* \tag{3.7}$$

$$x_{i+1} = x_i + d_i^* \tag{3.8}$$

has to be used when T is large.

The combination, *sparse matrix technique + large drop- tolerance + iterative refinement*, is used in the code LLSS01 (see

[29,30]). The vectors r_i , r_i^* , d_i^* and x_i are stored in double precision. All inner products (3.4)-(3.7) are accumulated in double precision. The non-zero elements of all matrices are stored in single precision. In this way the accepted approximation, x_p , as a rule has accuracy of order $O(\varepsilon^2)$ where ε is the machine accuracy in single precision [1,2,4]. Strategy 1 is implemented in LLSS01. Another version, LLSS02, where Strategy 2 is implemented, has been developed. Since the only difference between LLSS01 and LLSS02 is the pivotal strategy, any difference in the results is caused by the pivotal strategy only.

3.2. Storage scheme. The storage scheme is the same for both codes and is based on ideas proposed by Gustavson [14,15]. Let NZ be the number of non-zero elements in A . Before the beginning of the computations the non-zero elements of matrix A are stored in the first NZ locations of a real array A . The order of the non-zero elements can be arbitrary but if $A(L)=a_{ij}$ then $SNR(L)=j$ and $RNR(L)=i$ must be assigned (L=1(1)NZ, SNR and RNR are integer arrays of length NN). The codes order the non-zero elements by rows and store them in the first NZ locations of array A again, so that $A(L)=a_{ij} \wedge A(M)=a_{st} \wedge i<s \Rightarrow L<M$ and $A(L)=a_{ij} \Rightarrow SNR(L)=j$. Some additional information about the row starts and row ends is also stored by the codes. A copy of array A and array SNR is made in the real array A1 and integer array SN (both of length NZ). The information about the row starts and row ends is also copied. This information is used in the iterative process (see (3.6)). In the pivotal search it is necessary to scan the columns. Therefore the structure is ordered by columns, but only the row numbers of the non-zero elements so ordered are stored before the beginning of the decomposition in the first NZ loca-

m	Strategy 1				Strategy 2			
	COUNT	Time	Iterations	Accuracy	COUNT	Time	Iterations	Accuracy
100	210	0.58	5	4.16E-26	210	0.60	5	1.91E-26
110	220	0.73	7	1.06E-27	227	0.82	9	2.88E-26
120	230	0.62	5	3.77E-27	230	0.65	5	3.51E-27
130	240	0.64	5	3.74E-27	240	0.64	5	8.48E-27
140	250	0.68	6	2.88E-27	250	0.74	7	2.55E-27
150	260	0.65	5	3.74E-27	260	0.71	6	6.21E-27
160	270	0.77	7	1.54E-27	270	0.67	5	5.96E-27
170	280	0.71	6	2.88E-27	280	0.84	8	1.82E-27
180	290	0.72	6	3.10E-27	290	0.67	5	5.35E-27
190	300	0.85	7	1.06E-27	300	0.96	9	2.88E-27
200	310	0.83	6	3.69E-27	310	0.97	8	1.74E-27

Table 3.1

Problems with matrices A=F2(m,100,11,1,10) are solved. NZ=m+110, NN=4800, $T=10^{-25}$. The experiment has been run on a CDC Cyber 173.

tions of array RNR so that if RNR(L)=i and RNR(M)=s (where i and s are the row numbers of a_{ij} and a_{st}) and if $j < t$, then $L < M$. The row numbers of the non-zero elements in column k are not needed after the major step k and are therefore removed from array RNR by the codes. The non-zero elements a_{ik} $(i > k)$ are transformed into zero in major step k, therefore these elements and their column numbers are removed from arrays A and SNR. Some elements (with their row and column numbers) may be removed when positive values of the drop-tolerance T are used. Unfortunately, some new non-zero elements (fill-ins) are also created (when one of the elements involved in (1.6) is zero, while the other is non-zero). Such new elements and their column numbers (row numbers) are stored at the end of the row (column) if there are free lo-

m	Strategy 1				Strategy 2			
	COUNT	Time	Iterations	Accuracy	COUNT	Time	Iterations	Accuracy
100	872	1.42	5	4.53E-26	878	1.60	7	3.04E-26
110	559	1.23	7	9.59E-26	568	1.45	10	4.97E-27
120	522	1.18	7	2.90E-27	526	1.27	8	2.20E-27
130	536	1.20	7	1.89E-27	560	1.48	10	6.94E-27
140	611	1.15	5	1.96E-26	617	1.33	7	8.78E-27
150	677	1.54	8	9.59E-28	653	1.50	8	6.79E-27
160	638	1.42	7	1.62E-26	687	1.58	8	6.21E-27
170	717	1.75	9	2.32E-27	711	1.56	7	6.41E-27
180	776	1.73	7	9.59E-28	783	1.80	8	1.01E-27
190	811	1.84	7	1.32E-26	812	2.00	9	1.56E-26
200	817	1.73	5	3.02E-26	858	2.10	9	9.69E-26

Table 3.2

Problems with matrices A=F2(m,100,11,2,10) are solved. NZ=2m+110, NN=4800, $T=10^{-25}$. The experiment has been run on a CDC Cyber 173.

cations. Otherwise a copy of the row (column) is made at the end of arrays A and SNR (RNR). It is clear that there is a limit to the number of new copies that can be made without exceeding the capacity of the arrays. Therefore occasional "garbage" collections are necessary. The "garbage" collections increase the computing time and should be avoided (using a large length for the arrays A, SNR and RNR) when this is possible. See more details about the storage scheme in [30].

3.3. Test-matrices. The results (especially the storage and the computing time) depend on the preservation of sparsity during (1.1). The preservation of sparsity depends on the dimensions of matrix A, on the distribution of the non-zero elements within A,

m	Strategy 1				Strategy 2			
	COUNT	Time	Iterations	Accuracy	COUNT	Time	Iterations	Accuracy
100	1997	5.26	15	2.73E-27	2076	4.70	10	1.57E-27
110	2217	4.64	7	3.89E-27	2439	5.76	7	2.68E-27
120	2331	4.89	5	1.34E-26	2095	5.09	8	1.97E-27
130	2336	5.96	9	7.57E-26	2378	6.50	9	2.02E-27
140	2287	5.94	7	5.58E-27	2476	6.71	8	7.57E-28
150	2896	8.35	7	2.12E-27	2504	6.66	5	1.27E-26
160	2644	8.82	9	2.27E-27	3052	11.48	10	8.58E-28
170	2676	8.73	7	4.04E-28	3198	13.21	11	1.36E-27
180	3361	14.14	8	1.92E-27	3595	16.93	5	8.96E-27
190	3334	14.46	7	3.08E-27	3435	16.33	5	1.26E-26
200	3690	16.83	7	3.34E-27	4143	22.64	8	2.63E-27

Table 3.3

Problems with matrices A=F2(m,100,11,3,10) are solved. NZ=3m+110, NN=4800, $T=10^{-25}$. The experiment has been run on a CDC Cyber 173.

on the magnitude of the non-zero elements and on the number of the non-zero elements before the beginning of the decomposition. Therefore, it is desirable to develop a generator for test-matrices where one or several of the above characteristics can be specified and changed automatically. Such a generator (subroutine MATRF2) has been developed (a full documentation is in preparation). This generator produces matrices of class F2(m,n,c,r,α) where the parameters can be varied and have the following significance: (i) the number of rows in the desired matrix can be specified by m, (ii) the number of columns in the desired matrix can be specified by n, (iii) the positions of certain non-zero elements within the desired matrix can be specified by c, (iv) the number NZ of non-zero elements in the desired matrix can be specified by r so that NZ=rm+110,

T	Strategy 1			Strategy 2		
	COUNT	Time	Iterations	COUNT	Time	Iterations
10^{-25}	2706(0.85)	8.91(0.85)	8.0(1.03)	2854	10.54	7.8
10^{-2}	1169(0.71)	3.72(0.75)	16.8(1.10)	1646	4.98	15.3
10^{-1}	816(0.65)	3.32(0.72)	22.2(1.02)	1259	4.58	21.7

Table 3.4

Problems with matrices A=F2(m,100,11,3,10), m=100(10)200, are solved. NZ=3m+110, NN=4800. The average results (for the 11 problems solved with each value of the drop-tolerance T) are given in this table. The ratios of the characteristics obtained by Strategy 1 and the corresponding characteristics obtained by Strategy 2 are given in brackets. The experiment has been run on a CDC Cyber 173.

(v) the magnitude of the non-zero elements in the desired matrix can be specified by α so that $max(|a_{ij}|)/min(|a_{ij}|) = 10\alpha^2$, $a_{ij}\neq 0$.

The matrices of class F2(m,n,c,r,α) were used in all experiments. All parameters were varied in quite large intervals; some of the results are given in Table 3.1 - Table 3.9.

3.4. Comparison of the storage required by the two strategies.

It is not so easy to compare the storage required by the two strategies because the optimal value of NN (the length of the three large arrays A, SNR, RNR) is not known in advance. The number of non-zero elements in S should not be used in the comparison (if the column interchanges happen to be the same, then the same number of non-zero elements in matrix S will be produced by the two strategies,

m	Strategy 1		Strategy 2	
	Growth factor	Smallest element	Growth factor	Smallest element
100	6.50	6.21E-2	2.88E+15	2.98E-32
110	4.69	8.85E-3	1.05E+11	4.07E-27
120	2.36	1.69E-2	2.34E+17	7.12E-26
130	1.46	1.31E-1	5.94E+ 8	1.66E-23
140	2.07	3.27E-2	2.38E+12	2.32E-27
150	1.12	1.97E-2	1.54E+14	5.13E-30
160	2.41	3.88E-2	1.02E+ 9	2.22E-20
170	4.42	6.98E-3	1.75E+11	6.31E-27
180	4.50	1.59E-3	1.63E+ 8	1.55E-19
190	6.52	5.45E-4	4.05E+12	1.73E-27
200	4.14	2.65E-3	1.13E+ 8	1.04E-19

Table 3.5

Problems with matrices A=F2(m,100,11,3,10) are solved. NZ=3m+110, NN=4800, $T=10^{-2}$. "Growth factor" is the ratio of the largest (in absolute value)element found in array A during any step of the decomposition and the largest (in absolute value) element in array A before the beginning of the decomposition. "Smallest element" is the smallest element in matrix D_{n+1}. The experiment has been run on a CDC Cyber 173.

but this does not mean that the computational work and the number of fill-ins are the same; see Corollary 2.1 and Example 2.1).

Denote by $COUNT_1$ and $COUNT_2$ the maximal numbers of non-zero elements kept in array A when Strategy 1 and Strategy 2 are used. $COUNT_1$ and $COUNT_2$ can be used in the comparison (note that COUNT without any index will be used when there is no danger of misunderstanding). If e.g. $COUNT_1 < COUNT_2$ on a given class of problems, then Strategy 1 performs better than Strategy 2 for

n	Strategy 1				Strategy 2			
	COUNT	Time	Iterations	Accuracy	COUNT	Time	Iterations	Accuracy
100	4323	2.95	9	1.35E-14	4685	3.82	9	1.67E-14
110	4686	3.78	10	1.80E-14	4607	3.72	11	2.22E-14
120	3212	1.15	9	2.33E-14	4220	2.48	10	1.35E-14
130	3160	1.22	11	1.60E-14	4144	2.25	9	1.87E-14
140	2946	1.23	9	2.98E-14	4145	1.92	7	1.82E-14
150	3683	1.43	8	2.38E-14	4202	2.14	8	2.35E-14

Table 3.6

Problems with matrices A=F2(200,n,11,6,10) are solved. NZ=1110, NN=10000, $T=10^{-2}$. The experiment has been run on an IBM 3033.

this class because either smaller values of NN could be specified with Strategy 1 (reducing the storage) or the number of "garbage" collections with Strategy 1 will often be smaller when the same values of NN are used with both strategies.

Denote $E_s = COUNT_1/COUNT_2$. E_s will be called the efficiency factor with regard to the storage requirements. By the use of E_s the following conclusions can be drawn for the two strategies from our experiments with matrices of class F2(m,n,c,r,α) (which were run with different values of the parameters).

(i) If the matrix is very sparse, then $E_s \approx 1$ is normally observed (see Table 3.1, Table 3.2 and the first line of Table 3.8). This result should be expected (because in this situation it is very likely that the conditions of Theorem 2.2 are satisfied) and has already been reported in [7].

(ii) If the matrix is not very sparse, then often $E_s < 1$ (i.e. Strategy 1 performs better than Strategy 2), see Table 3.3, Table 3.6, Table 3.7 and Table 3.8 .

	Strategy 1				Strategy 2			
c	COUNT	Time	Iterations	Accuracy	COUNT	Time	Iterations	Accuracy
20	4118	2.40	9	1.38E-14	4088	2.30	7	1.07E-14
25	3975	2.50	8	3.94E-15	4101	2.50	10	1.51E-14
30	3683	2.24	11	2.80E-14	4088	2.33	10	2.42E-14
35	4033	2.57	8	2.93E-14	4525	2.87	8	3.33E-14
40	3441	2.04	9	3.31E-14	4560	3.13	8	3.20E-14
45	2612	1.26	9	3.09E-14	3407	1.70	8	1.95E-14
50	3025	1.57	10	2.93E-14	3840	1.94	10	2.64E-14
55	3478	1.92	10	2.64E-14	4288	2.66	10	2.78E-14

Table 3.7

Problems with matrices A=F2(150,100,c,6,10) are solved. NZ=1010, NN=10000, $T=10^{-2}$. The experiment has been run on an IBM 3033 .

(iii) If the drop-tolerance T is increased and if the matrix is not very sparse, then E_s becomes smaller (see Table 3.4). An explanation of this phenomenon can be given as follows. The algorithm described by (1.7)-(1.11) cannot be applied with Strategy 2. This leads to very small elements in D_{n+1} and very large elements in A_k (k=1(1)n) when Strategy 2 is used in the decomposition (1.1). Let $a = \max_{i,j}(|a_{ij}|)$ where a_{ij} (i=1(1)m, j=1(1)n) are the elements of matrix A before the beginning of the decomposition. Let $\bar{a}$ be the absolute value of the largest in absolute value element kept in array A during any step of the decomposition (1.1). The ratio $\bar{a}/a$ is called the growth factor. The results given in Table 3.5 show that the growth factors are very large when Strategy 2 is used (note that the growth factors become larger for $T=10^{-25}$; this shows that the growth factors tend to increase when the drop-tolerance T is decreased). There-

r	Strategy 1			Strategy 2		
	COUNT	Time	Iterations	COUNT	Time	Iterations
2	427(0.96)	1.62(1.02)	16.7(1.06)	447	1.59	15.7
3	816(0.65)	3.32(0.72)	22.7(1.05)	1259	4.58	21.7
4	1346(0.68)	4.76(0.66)	20.1(1.00)	1989	7.16	20.2
5	1962(0.72)	7.96(0.71)	21.6(1.03)	2718	11.28	21.0
6	2144(0.70)	8.94(0.66)	21.3(1.22)	3044	13.47	17.4

Table 3.8

Problems with matrices A=F2(m,100,11,r,10), m=100(10)200, are solved. NZ=rm+110, NN=4800, $T=10^{-1}$. The average results (for the 11 problems solved with each value of parameter r) are given in this table. The ratios of the characteristics obtained by Strategy 1 and the corresponding characteristics obtained by Strategy 2 are given in brackets. The experiment has been run on a CDC Cyber 173.

fore the number of non-zero elements which are removed by the drop-tolarance is smaller for Strategy 2.

It is necessary to emphasize here that the use of iterative refinement with a large value of the drop-tolarance gives a great reduction in storage for these problems (the use of two extra arrays A1 and SN in the iterative process is fully compensated because the length of the other three large arrays can be chosen much smaller when T is large). This is normally so for problems which produce many fill-ins. If the problem does not produce many fill-ins (if e.g. r=1 or r=2), then the use of iterative refinement will require some extra storage.

	Strategy 1			Strategy 2		
α	m=100	m=150	m=200	m=100	m=150	m=200
10	4.16E-26	2.88E-27	1.06E-27	1.96E-26	6.21E-27	1.74E-27
10^2	4.38E-25	3.33E-26	2.29E-26	9.08E-26	3.07E-26	3.50E-26
10^3	2.20E-24	2.24E-25	2.01E-24	2.04E-24	5.46E-25	2.36E-25
10^4	1.17E-23	1.53E-24	4.55E-24	1.66E-23	1.15E-24	6.97E-25
10^5	4.93E-23	2.32E-23	8.45E-24	3.10E-23	3.83E-24	2.12E-23
10^6	1.06E-21	2.00E-22	1.05E-22	3.78E-22	2.64E-22	3.70E-22
10^7	3.60E-21	1.24E-21	1.31E-21	2.63E-21	1.50E-21	2.18E-21
10^8	1.60E-19	3.38E-20	1.11E-19	5.77E-20	1.22E-19	4.55E-19
10^9	3.81E-17	4.48E-17	1.46E-17	8.44E-17	4.79E-17	1.79E- 3
10^{10}	2.39E-13	7.21E-16	2.68E-16	3.66E-15	1.11E-13	2.90E 0

Table 3.9

Problems with matrices $A=F2(m,100,11,2,\alpha)$ are solved. NZ=2m+110, NN=4800, $T=10^{-25}$. The accuracy of the approximations calculated by the two codes is given in this table. The experiment has been run on a CDC Cyber 173.

(iv) If $r < 3$, then $E_s \approx 1$. If $r \geq 3$, then $E_s \approx 0.7$; see Table 3.8. No clear relation between E_s and any of the parameters m, n, c and α has been observed; see some results in Table 3.3, Table 3.6 and Table 3.7. However, note that if the matrix is not very sparse, then Strategy 1 performs better than Strategy 2 for any choice of the parameters m, n, c and α.

3.5. Comparison of the computing time required by the two strategies. Denote by t_1 the computing time used by LLSS01 and by t_2 that used by LLSS02. The number $E_t = t_1/t_2$ will be called the efficiency factor with regard to the computing time required. Conclusions (i) - (iv) from Section 3.4 hold also when

E_s is replaced by E_t. This means that if the matrix is not very sparse, then the computing time will very often be reduced when Strategy 1 is used instead of Strategy 2. In some examples the computing time obtained by Strategy 1 is halved in comparison with that obtained by Strategy 2. See e.g. Table 3.6.

Note again the great efficiency of the use of large values of the drop-tolerance T (see Table 3.4). However, it should be noted that if the number of fill-ins is small (which is typical for very sparse matrices), then the use of a large drop-tolerance will cause more iterations and therefore extra computing time. Of course, the use of large values for the drop-tolerance leads to extra iterations also when the matrix is not very sparse and many fill-ins are produced. However, the reduction in the computing time for the decomposition is so great in this case (because many fill-ins are removed by the drop-tolerance) that the total computing time is also reduced considerably.

3.6. Comparison of the accuracy achieved by the two strategies.

If the matrix is well conditioned (if $\alpha < 100$ in our experiments), then the accuracy achieved by the two subroutines is approximately the same (of order of magnitude $O(\varepsilon^2)$ where ε is the machine accuracy; $\varepsilon=O(10^{-14})$ for CDC Cyber 173 and $\varepsilon=O(10^{-7})$ for IBM 3033). The fact that the accuracy achieved has order of magnitude $O(\varepsilon^2)$ is in full agreement with the results obtained by Björck [1,2,4]. Note that the number of iterations needed to obtain such accuracy is approximately the same for both subroutines when the drop-tolerance is very small, $T=10^{-25}$ in our experiments (in Table 3.1 and Table 3.2 the numbers of iterations for Strategy 1 are slightly smaller than those for Strategy 2). There is a tendency that the numbers of iterations for Strategy 2 are smaller than those for Strategy 1 when the drop-tolerance is

large (see Table 3.4 and Table 3.8). This is probably caused by the fact that more elements are removed when Strategy 1 is used with a large drop-tolerance (see the values of COUNT in the tables and Section 3.4) and thus the decomposition computed by Strategy 1 is not very accurate. Since the computational cost per iteration is much smaller than that for the decomposition (1.1), the fact that Strategy 1 uses more iterations has no visible influence on the efficiency factor with regard to the computing time required (see Section 3.5).

An experiment with different values of α has also been carried out. The matrices $A=F2(m,100,11,2,\alpha)$ with $m=100(10)200$ have been used in this experiment with $T=10^{-25}$ and $\alpha=10(10)10^{10}$. The results for $\alpha < 10^9$ are comparable. For $\alpha=10^9$ and $\alpha=10^{10}$ Strategy 1 gives much more accurate results for some problems. The results for $m=100$, $m=150$ and $m=200$ are given in Table 3.9.

The experiments indicate that the fact,that Strategy 2 produces very small elements in D_{n+1} and very large (in absolute value) elements in A_k $(k=1(1)n+1)$, has not a great influence on the accuracy of the approximations computed using this pivotal strategy.

3.7. Robustness of the computations. The fact, that the elements of D_{n+1} become very small and the growth factors (see Section 3.4) become very large when Strategy 2 is used (see Table 3.5), can lead to underflows and overflows. Note that,when the problem with matrix $A=F2(110,100,11,2,10)$ was solved with Strategy 2, the growth factor was of magnitude $O(10^{140})$. On CDC Cyber 173 this did not cause trouble, but on many other computers such a large growth factor will lead to overflow. Of course, there is no guarantee against overflows when Strategy 1 is used. But the results given in Table 3.5

indicate that the computations with Strategy 1 are much more robust than those with Strategy 2.

3.8. General conclusion. The numerical experiments indicate that Strategy 1 should be preferred in the choice between the two strategies discussed in this paper when problem (3.1) is solved and when the plane rotations have to be used in the decomposition of matrix A .

R E F E R E N C E S

1. Å. Björck, *Iterative Refinement of Linear Least Squares Problems I* . BIT 7, 257-278 (1967).

2. Å. Björck, *Iterative Refinement of Linear Least Squares Problems II* . BIT 8, 1-30 (1968).

3. Å. Björck, *Methods for Sparse Linear Least Squares Problems*. In *Sparse Matrix Computations* (Edited by J. Bunch and D. Rose), pp. 179-199. Academic Press, New York (1976).

4. Å. Björck, *Comment on the Iterative Refinement of Least Squares Problems*. J. Amer. Stat. Assoc. 73, 161-166 (1978).

5. Å. Björck, *Numerical Algorithms for Linear Least Squares Problems*. Report No. 2/78. Department of Mathematics, University of Trondheim, Trondheim, Norway (1978).

6. R. J. Clasen, *Techniques for Automatic Tolerance in Linear Programming*. Comm. ACM 9, 802-803 (1966).

7. I. S. Duff, *Pivot Selection and Row Ordering in Givens Reduction of Sparse Matrices*. Computing 13, 239-248 (1974).

8. I. S. Duff and J. K. Reid, *A Comparison of Some Methods for the Solution of Sparse Overdetermined Sistems of Linear Equations*. J. Inst. Math. Appl. 17, 267-280 (1976).

9. W. M. Gentleman, *Least Squares Computations by Givens' Transformations Without Square Roots.* J. Inst. Math. Appl. 12, 329-336 (1973).

10. W. M. Gentleman, *Row Elimination for Solving Sparse Linear Systems and Least Squares Problems.* In *Numerical Analysis Dundee 1975* (Edited by G. A. Watson), pp. 122-133. Lecture Notes in Mathematics No 506, Springer, Berlin (1975).

11. W. M. Gentleman, *Error Analysis of QR Decompositions by Givens' Transformations.* Lin. Alg. Appl. 10, 189-197 (1975).

12. J. W. Givens, *Numerical Computation of the Characteristic Values of a Real Symmetric Matrix.* Report No. ORNL-1574. Oak Ridge National Laboratory (1954).

13. J. W. Givens, *Computations of Plane Unitary Rotations Transforming a General Matrix to Triangular Form.* J. Soc. Ind. Appl. Math. 6, 26-50 (1958).

14. F. G. Gustavson, *Some Basic Techniques for Solving Sparse Systems of Linear Equations.* In *Sparse Matrices and Their Applications* (Edited by D. J. Rose and R. A. Willoughby), pp. 41-52. Plenum Press, New York (1972).

15. F. G. Gustavson, *Two Fast Algorithms for Sparse Matrices: Multiplication and Permuted Transposition.* ACM Trans. Math. Software 4, 250-269 (1978).

16. S. Hammarling, *A Note on Multiplications to Givens Plane Rotations*. J. Inst. Math. Appl. 13, 215-218 (1974).

17. C. L. Lawson, R. J. Hanson, D. R. Kinsaid and F. T. Krogh, *Basic Linear Algebra Subprograms for Fortran Usage*. ACM Trans. Math. Software 5, 308-323 (1979).

18. E. H. Moore, *On the Reciprocal of the General Algebraic Matrix*. Bull. Amer. Math. Soc. 26, 394-395 (1919-1920).

19. R. Penrose, *A Generalized Inverse for Matrices*. Proc. Cambridge Phil. Soc. 51, 506-513 (1955).

20. J. K. Reid, *Fortran Subroutines for Handling Sparse Linear Programming Bases*. Report No. R8269, A.E.R.E., Harwell, England (1976).

21. J. K. Reid, *Sparse Matrices*. In *The State of the Art in Numerical Analysis* (Edited by D. A. H. Jacobs), pp. 85-146. Academic Press, London (1977).

22. K. Schaumburg, J. Wasniewski and Z. Zlatev, *The Use of Sparse Matrix Technique in the Numerical Integration of Stiff Systems of Linear Ordinary Differential Equations*. Computers and Chemistry 4, to appear.

23. G. W. Stewart, *The Economical Storage of Plane Rotations*. Numer. Math. 25, 137-139 (1976).

24. R. P. Tewarson, *Sparse Matrices*. Academic Press, New York (1973)

25. J. H. Wilkinson, *Some Recent Advances in Numerical Linear Algebra*. In *The State of the Art in Numerical Analysis* (Edited by D.A.H.Jacobs), pp. 3-53. Academic Press, New York (1977).

26. P. Wolfe, *Error in the Solution of Linear Programming Problems*. In *Error in Digital Computations* (Edited by L.B.Rall), Vol.2, pp.271-284. Wiley, New York (1965).

27. Z. Zlatev, *Use of Iterative Refinement in the Solution of Sparse Linear Systems*. Report 1/79. Institute of Mathematics and Statistics, The Royal Veterinary and Agricultural University, Copenhagen, Denmark (1979).

28. Z. Zlatev, *On Solving Some Large Linear Problems by Direct Methods*. Report 111. Computer Science Department, Aarhus University, Aarhus, Denmark (1980).

29. Z. Zlatev and H. B. Nielsen, *LLSS01 - a Fortran Subroutine for Solving Lest Squares Problems (USER'S GUIDE)*. Report No. 79-07. Institute for Numerical Analysis, Technical University of Denmark, Lyngby, Denmark (1979).

30. Z. Zlatev and H. B. Nielsen, *Least Squares Solution of Large Linear Problems*. In *Symposium i Anvendt Statistik 1980* (Edited by A.Höskuldsson,K.Conradsen,B.Sloth Jensen and K.Esbensen), pp.17-52. NEUCC, Lyngby, Denmark (1980).

31. Z. Zlatev, J. Wasniewski and K. Schaumburg, *Comparison of Two Algorithms for Solving Large Linear Systems*. Report 80/9. Regional Computing Centre at the University of Copenhagen,Copenhagen, Denmark (1980).