ISSN 0105-8517

ON SOLVING SOME LARGE LINEAR PROBLEMS BY DIRECT METHODS

by

Zahari Zlatev

DAIMI PB-111
February 1980

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

ON SOLVING SOME LARGE LINEAR PROBLEMS BY DIRECT METHODS

by

Zahari Zlatev

ABSTRACT

Let $n \in \mathbb{N}$, $m \in \mathbb{N}$, $b \in \mathbb{R}^{m \times 1}$ and $A \in \mathbb{R}^{m \times n}$ be given. Assume that $m \leq n$ and $\operatorname{rank}(A)=n$. Consider the problem: find vector $x \in \mathbb{R}^{n \times 1}$ from $x=A^{\dagger}b$ where $A^{\dagger}$ is the pseudo-inverse of A. A general scheme for obtaining approximations $\bar{x}$ to x is proposed. It is shown that if the computations are performed without rounding errors then $\bar{x}=x$. It is demonstrated how some well-known and commonly used special methods can be found from the general scheme. In the case where A is large and sparse some general rules (use some sparse matrix technique, choose carefully the pivotal strategy, select a large drop-tolerance, perform iterative refinement of the solution found by the general scheme), which often lead to a considerable reduction of the computing time and/or the storage needed and to an improved accuracy, are given. Assume that: (A) a part of the computations can be performed without errors. Then from several theorems it follows that: (i) there exists a relationship between the drop-tolerance and the condition number of the coefficient matrix, (ii) the iterative process can sometimes be accelerated. Many experiments have been carried out in order to show that there is a tendency that (i) and (ii) hold even if the assumption (A) is removed. The efficiency of the suggested general rules is illustrated by numerical results.

1. The general k-stage direct method: definition and examples

Let $n \in \mathbb{N}$, $m \in \mathbb{N}$, $b \in \mathbb{R}^{m\times 1}$ and $A \in \mathbb{R}^{m\times n}$ be given. Assume that $m \geq n$ and $\mathrm{rank}(A)=n$. Denote by $A^{\dagger}$ the pseudo-inverse (or the Moore-Penrose generalized inverse, see [30,32]) of matrix A. It is clear that the assumption $\mathrm{rank}(A)=n$ implies $A^{\dagger}=(A^TA)^{-1}A^T$. Consider the following problem: find an approximation $\bar{x} \in \mathbb{R}^{n\times 1}$ to the vector $x \in \mathbb{R}^{n\times 1}$ given by

$$(1.1) \qquad x = A^{\dagger}b .$$

Note that x can equivalently be considered: (i) as a least-squares solution (i.e. as the vector which minimizes $\|r\|_2$ where $r=b-Ax$) or (ii) as the solution of the system $Ax=b-r$ where $A^Tr=0$ is satisfied.

In this paper it will be shown that: (i) most of the direct methods used in the approximate solution of (1.1) can be found as special cases in a quite general scheme, (ii) some general rules will normally lead to a considerable improvement in the performance of many special methods in the general scheme when problem (1.1) is large.

Assume that it is possible to replace problem (1.1) by the following problem

$$(1.2) \qquad y = B_1^{\dagger}c , \quad B_1 \in \mathbb{R}^{\bar{p}\times\bar{q}}, \quad \bar{p} \in \mathbb{N} , \quad \bar{q} \in \mathbb{N} , \quad \bar{p} \geq \bar{q} ,$$

$$\mathrm{rank}(B_1) = \bar{q} ,$$

where (i) B_1 and c can be computed from A and

b , (ii) there exists a simple relationship between x and y .

Use the following two steps to obtain an approximation y_1 to the solution vector y of (1.2) .

Step 1 - Generalized decomposition. Compute

$$(1.3) \quad \bar{B}_i = P_i B_i Q_i + E_i , \quad i=1(1)k , \quad k \in \mathbb{N} ,$$

where P_i and Q_i are permutation matrices (if B_i is symmetric for some i and if the symmetry is exploited in (1.3) , then $Q_i = P_i^T$), E_i are perturbation matrices, $\bar{B}_k$ is such that $\bar{B}_k^{\dagger} z$ (where z is an arbitrary vector of appropriate order) can easily be computed; if $k > 1$ then

$$(1.4) \quad \bar{B}_i = C_i \bar{C}_i D_i , \quad B_{i+1} = C_i^T C_i \bar{C}_i , \quad i=1(1)k-1$$

(D_i are such that $D_i^{\dagger} z_i$, z_i are arbitrary vectors of appropriate order, can easily be computed).

Step 2 - Generalized back substitution. Compute vector y_1 by the use of

$$(1.5) \quad y_1 = \Big(\prod_{i=1}^{k-1} Q_i D_i^{\dagger}\Big) Q_k \bar{B}_k^{\dagger} P_k \Big(\prod_{i=1}^{k-1} P_i^T C_i\Big)^T c ,$$

where it is assumed that the terms in brackets are equal to the identity matrix I (of appropriate order) when $k=1$.

It is clear that if y_1 is an approximation to y then the relationship between x and y can be used to obtain an approximation $\bar{x}$ to the solution of (1.1) from y_1. Therefore it is necessary to prove that y_1 will be a good approximation to y when the perturbation matrices E_i $(i=1(1)k)$ are sufficiently small. Denote

$$(1.6) \qquad H = \Big(\prod_{i=1}^{k-1} Q_i D_i^{\dagger}\Big) Q_k \bar{B}_k^{\dagger} P_k \Big(\prod_{i=1}^{k-1} P_i^T C_i\Big)^T , \qquad H \in \mathbb{R}^{\bar{q}\times\bar{p}}$$

and

$$(1.7) \qquad F = I - HB_1 , \qquad F \in \mathbb{R}^{\bar{q}\times\bar{q}}, \qquad I \in \mathbb{R}^{\bar{q}\times\bar{q}} .$$

Then the following theorem holds.

<u>Theorem 1.1</u> <u>Assume that $\bar{B}_k$ and D_i $(i=1(1)k-1)$ have a full column rank. Then</u>

$$(1.8) \qquad F = \sum_{j=1}^{k} H_j$$

<u>with</u>

$$(1.9) \qquad H_j = \Big(\prod_{i=1}^{k-1} Q_i D_i^{\dagger}\Big) Q_k \bar{B}_k^{\dagger} P_k \Big(\prod_{i=j}^{k-1} P_i^T C_i\Big)^T P_j^T E_j Q_j^T \Big(\prod_{i=1}^{j-1} Q_i D_i^T\Big)^T ,$$

<u>where $j=1(1)k$ and all products are assumed to be equal to the identity matrix of an appropriate order when the upper index is smaller than the lower one.</u>

<u>Proof</u> By the use of (1.7) , (1.6) , (1.3) and (1.4) the following equality can be obtained

$$(1.10)\qquad F = I - \left(\prod_{i=1}^{k-1} Q_i D_i^{\dagger}\right) Q_k \bar{B}_k^{\dagger} P_k \left(\prod_{i=2}^{k-1} P_i^T C_i\right)^T B_2 \left(\prod_{i=1}^{1} Q_i D_i^T\right)^T + H_1 .$$

By successive use (k-2 times) of (1.10) , (1.3) and (1.4) the equality (1.10) will be transformed into

$$(1.11)\qquad F = I - \left(\prod_{i=1}^{k-1} Q_i D_i^{\dagger}\right) Q_k \bar{B}_k^{\dagger} P_k B_k \left(\prod_{i=1}^{k-1} Q_i D_i^T\right)^T + \sum_{j=1}^{k-1} H_j .$$

By the use of (1.3) the last equality can be rewritten as follows

$$(1.12)\qquad F = I - \left(\prod_{i=1}^{k-1} Q_i D_i^{\dagger}\right) Q_k \bar{B}_k^{\dagger} \bar{B}_k Q_k^T \left(\prod_{i=1}^{k-1} Q_i D_i^T\right)^T + \sum_{j=1}^{k} H_j .$$

The second term in the right-hand side of (1.12) is equal to the identity matrix (the assumption that the matrices involved in this term have full column rank should be used in the proof of this statement) and, thus, the theorem is proved.

□

Corollary 1.1 *If* $E_i=0$ *(i=1(1)k) then* $H=B_1^{\dagger}$. *If, moreover, the computations by (1.5) can be performed without rounding errors, then* $y_1=y$.

□

Of course, the real computations are always connected with some perturbations (i.e. in the real computations $E_i \neq 0$, $i=1(1)k$). Nevertheless, Theorem 1.1 and Corollary 1.1 indicate that y_1 found by (1.5) can be considered as an aproximation to y and this (in connection with the relation between x and y) justifies the following definition (but does not remove the necessity to check if the approximation obtained satisfies the accuracy requirements; this will be discussed in Section 2).

Definition 1.1 The computational scheme given by Step 1 and Step 2 is called a general k-stage direct method for solving (1.1) .

Six examples are given below in order to illustrate how this general treatment of the direct methods can be applied to some particular algorithms which are well-known and commonly used in practice.

Example 1.1 Let $m=n$. Then the classical Gaussian elimination can be found from the k-stage scheme with $k=1$ and

$$(1.13) \quad B_1=A, \quad c=b, \quad y=x, \quad \bar{B}_1=L_g U_g, \quad \bar{B}_1^{\dagger}=U_g^{-1}L_g^{-1}, \quad \bar{x}=y_1 ,$$

where $L_g \in \mathbb{R}^{nxn}$ is a unit lower triangular matrix and $U_g \in \mathbb{R}^{nxn}$ is an upper triangular matrix.

This example is very simple and will be used as an illustrative example in the following sections. Therefore it is useful to give the main formula applied in the Gaussian transformations:

$$(1.14) \quad a_{ij}^{(s+1)}=a_{ij}^{(s)}-a_{is}^{(s)}a_{sj}^{(s)}/a_{ss}^{(s)} , \quad a_{ss}^{(s)}\neq 0 , \quad a_{ij}^{(1)}=a_{ij} \in A ,$$

where $s=1(1)n-1$, $i=s(1)n$, $j=s(1)n$.

Example 1.2 Assume that the normal equations are solved by some symmetric version of the Gaussian elimination. This special method can be found from the k-stage scheme with <u>k=1</u> and

$$(1.15)\qquad B_1=A^TA,\quad c=A^Tb,\quad y=x,\quad \bar{B}_1=L_cD_cL_c^T,\quad \bar{B}_1^\dagger=(L_c^T)^{-1}D_c^{-1}L_c^{-1},\quad \bar{x}=y_1,$$

where $L_c \in \mathbb{R}^{nxn}$ is a unit lower triangular matrix and $D_c \in \mathbb{R}^{nxn}$ is a diagonal matrix.

<u>Example 1.3</u> Consider the method where augmented matrices and Gaussian elimination are used. This method can be found from the general k-stage scheme with <u>k=1</u> and

$$(1.16)\qquad B_1=\begin{bmatrix}\alpha I & A\\ A^T & 0\end{bmatrix},\quad c=\begin{bmatrix}b\\ 0\end{bmatrix},\quad y\begin{bmatrix}\alpha^{-1}r\\ x\end{bmatrix},\quad \bar{B}_1=L_aU_a,\quad \bar{B}_1^\dagger=U_a^{-1}L_a^{-1},$$

where if $q=m+n$ then $L_a \in \mathbb{R}^{qxq}$ is a unit lower triangular matrix and $U_a \in \mathbb{R}^{qxq}$ is an upper triangular matrix. The aproximation $\bar{x}$ is the vector formed by the last n coordinates of the approximation y_1 found by (1.5).

Denote by

$$(1.17)\qquad \sigma_1 \geq \sigma_2 \geq \cdot\cdot\cdot \geq \sigma_n > 0$$

the singular values of matrix A. Then $\alpha \approx \sigma_n/\sqrt{2}$ in (1.6) ensures that the spectral condition number of matrix B_1 satisfies $\kappa(B_1) \approx \sqrt{2}\kappa(A)$, see e.g. Björck [5]. Note that $\kappa(B_1) = \kappa^2(A)$ for matrix B_1 in Example 1.2. Duff and Reid [16] report good results with the choice $\alpha=1$. However, if e.g. $\sigma_1 \approx \sqrt{2}$ then the choice $\alpha = 1$ will give $\kappa(B_1) \approx \kappa^2(A)$ (i.e. the same result as the normal equations); see more details about the choice of α in Björck [5] .

Example 1.4 Consider the case where the Peters-Wilkinson method is used (see [33]). This method can be found from the general k-stage scheme with $\underline{k=2}$ and

(1.18) $B_1=A$, $c=b$, $y=x$, $\bar{B}_1=L_pU_p$, $C_1=L_p$, $\bar{C}_1=I$, $D_1=U_p$;

(1.19) $B_2=L_p^TL_p$, $\bar{B}_2=\bar{L}_p\bar{D}_p\bar{L}_p^T$, $\bar{B}_2^{\dagger}=(\bar{L}_p^T)^{-1}\bar{D}_p^{-1}\bar{L}_p^{-1}$, $\bar{x}=y_1$;

where $L_p \in \mathbb{R}^{mxn}$ is a unit lower trapezoidal matrix, $U_p \in \mathbb{R}^{nxn}$ is an upper triangular matrix, $\bar{L}_p \in \mathbb{R}^{nxn}$ is a unit lower triangular matrix and $\bar{D}_p \in \mathbb{R}^{nxn}$ is a diagonal matrix.

Note that one can expect that the computations in the second stage will be performed with the same degree of accuracy as those in the first stage if $\kappa(L_p) \approx \sqrt{\kappa(A)}$. No theoretical proof that the Gaussian elimination applied during the first stage (1.18) produces matrices L_p whose spectral condition numbers satisfy the above relation is known. However, some heuristic considerations indicate that the ill-conditioning of A reflects normally in U and the lower matrix L is often well conditioned ([33] , see also Cline et al. [10]). Therefore it is not a surprise that the numerical evidence shows that the method is often numerically stable (see e.g. [16]).

Example 1.5 Consider the case where some orthogonal decomposition of matrix A is used. This method can be found from the general k-stage scheme with $\underline{k=1}$ and

(1.20) $B_1=A, \quad c=b, \quad y=x, \quad \bar{B}_1=RDS, \quad \bar{B}_1^{\dagger}=S^{-1}D^{-1}R^T, \quad \bar{x}=y_1$,

where $R \in \mathbb{R}^{m\times n}$ is such that $R^TR = I \in \mathbb{R}^{n\times n}$, $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix and $S \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. If the Householder method or the Givens method is used then $D=I$; if the Gentleman-Givens method is used then $D\neq I$.

Example 1.6 Another version of the method described in Example 1.5 can be derived as follows. Let k=2 . Assume that $P_i=Q_i=I$, $i=1,2$. Denote by P, Q, E, R, D and S the matrices used in (1.3) and (1.20) for Example 1.5 (i.e. ignore the index 1 in the matrices used in Example 1.5). Then (1.3) gives $PAQ+E=RDS$. Now a two-stage method can be defined by

(1.21) $B_1=A^TA, \quad c=A^Tb, \quad y=x, \quad \bar{B}_1=A^TP^TRDSQ^T, \quad E_1=A^TP^TEQ^T, \quad C_1=I,$

$\bar{C}_1=A^TP^TR, \quad D_1=DSQ^T;$

(1.22) $B_2=A^TP^TR, \quad \bar{B}_2=QS^TD, \quad \bar{B}_2^{\dagger}=D^{-1}(S^T)^{-1}Q^T, \quad E_2=QE^TR, \quad \bar{x}=y_1$.

Note that if the method from Example 1.5 is modified as above then the orthogonal matrix R does not participate in the computation of y_1 and, therefore, it is not necessary to store this matrix when the above method is applied.

The general k-stage scheme will be used in the next section in order to investigate the problem of how to find some general rules which applied to any particular method will normally improve its efficiency when the problem (1.1) is large.

Note that the permutation matrices used in (1.3) and (1.5) arise from the pivotal interchanges performed during the generalized decomposition. These interchanges are very important in the attempt to improve the efficiency of the method when large problems are solved. By including the permutation matrices in the description of the general k-stage direct method we want to emphasize that one should be very careful in the choice of these matrices (or, in other words, in the choice of the pivotal strategy; see also Section 6.1) and that the method selected will be improved considerably when the right choice of these matrices is made (when the problem (1.1) is large).

2. Practical aspects in the implementation of the general k-stage direct scheme in the solution of large problems

Consider the pseudo-inverse solution (1.1). Assume that: (i) n is large (say, $n \geq 100$) and (ii) A is sparse (i.e. many of its elements are equal to zero). Suppose that the problem is to be solved by some direct method selected from the general k-stage scheme described in Section 1 . In the practical implementation of the method chosen it is important to satisfy the following requirements:

Requirement 1 *The storage and the computing time needed to obtain an approximation* $\bar{x}$ *to vector* x *from (1.1) must be minimized.*

Requirement 2 *Let* ε_s *be an error tolerance (prescribed in advance). Then an attempt to check whether* $\| x-\bar{x} \| \leq \varepsilon_s$ *or not has to be carried out.*

Note that: (i) normally ε_s is not very stringent, but nevertheless the *a priori* bounds known are very pessimistic and can not be used in the check from Requirement 2 and (ii) in general, it will be impossible to satisfy simultaneously both Requirement 1 and Requirement 2, therefore some compromise has to be found.

Let us begin with Requirement 1 . It is well known that some sparse technique has to be used in order to satisfy this requirement (see e.g. Björck [3,5], Duff and Reid [17], Gustavson [26,27], Tewarson [41]). Roughly speaking, this means that: (i) only the non-zero elements of the matrices B_i and $\bar{B}_i$

$(i=1(1)k)$ are stored, (ii) an attempt to perform only the arithmetic operations which lead to alternations is carried out (e.g. the computations by formula (1.14) are not performed when $a_{is}^{(s)}=0$. and/or $a_{sj}^{(s)}=0$), and (iii) the pivotal interchanges are used not only as an attempt to preserve the numerical stability of the computations but rather to preserve the sparsity of the matrices B_i $(i=1(1)k)$. The last statement, (iii), is very important and it is necessary to explain it further. Look at formula (1.14) again. It is clear that if $a_{ij}^{(s)}=0$ but neither $a_{is}^{(s)}=0$ nor $a_{sj}^{(s)}=0$, then $a_{ij}^{(s+1)}\neq 0$, i.e. a new non-zero element (fill-in) is created in position (i,j). Such new non-zero elements are normally created by any direct method. Their number may sometimes be very large (see e.g. Brayton et al. [8]). There exist examples where matrix $B_1 \in \mathbb{R}^{nxn}$ has $3n-2$ non-zero elements, while the sum of the non-zero elements of L_g and U_g (found by the Gaussian elimination) is n^2 when the pivotal strategy is not carefully chosen (see e.g. Reid [36], pp. 108-109). To preserve the sparsity of matrices B_i means to reduce the number of fill-ins generated in Step 1 of the general k-stage direct method. It is worthwhile underlining here that the preservation of the sparsity by choosing appropriate permutation matrices will normally imply a loss of accuracy (this will be illustrated in Section 6.1).

Another means of preserving the sparsity, which is sometimes extremely efficient, is the use of a special parameter T ("drop-tolerance", see Tewarson [41], Clasen [9] and Wolfe [47]) so that if an element found during the computations in (1.3) is smaller (in absolute value) than T then it is replaced by zero (thus the location occupied by this element becomes in fact free and can be used to store another element). The use of large values of T may give much better preservation of the sparsity than the use of $T=0$ (see [48,56,37,53]). Note that the use of positive values of T is also recommended by Reid in [35]. However, this will normally lead to a loss of accuracy also.

The above considerations show that the preservation of sparsity (in other words, the efforts to satisfy Requirement 1) must as a rule be connected with an attempt to regain the accuracy lost. This can usually be done by adding iterative refinement to the general k-stage direct scheme as follows.

Step 3 - Generalized iterative refinement. Continue the computations after Step 2 in the following way

(2.1) $r_i = c - B_1 y_i$, $i=1(1)p-1$;

(2.2) $d_i = Hr_i$, $i=1(1)p-1$;

(2.3) $y_{i+1} = y_i + d_i$, $i=1(1)p-1$;

where different stop-criteria (see Wilkinson [43,44], Stewart [40] and Björck [4]) are used to terminate the iterative process after some $i=p-1$. This means that y_p is usually accepted as an approximation to y . The approximation $\bar{x}$ has to be found from y_r by the use of the relationship between x and y .

Fortunately, the iterative process gives also an estimation of $\|x-\bar{x}\|$, i.e. satisfies automatically Requirement 2.

It is clear now that a reasonable compromise in the efforts to satisfy Requirement 1 and Requirement 2 when n is large and A is sparse can be found by using an algorithm based on the following rules:

(i) Select some sparse matrix technique.

(ii) Choose carefully the pivotal strategy (so that it preserves the sparsity better).

(iii) Specify a large value for the drop-tolerance T .

(iv) Attempt to regain the accuracy lost during the generalized decomposition by iterative refinement.

In the following sections it is often assumed that the particular method selected from the general k-stage scheme is implemented so that the above rules are satisfied. The results given in Section 3 indicate that there exists a relationship between the drop-tolerance T and the condition number of matrix B_1 . A practical device which sometimes accelerates the speed of the convergence of the iterative process (2.1)-(2.3) is proposed in Section 4. Some numerical examples which illustrate how efficiently the rules described above can be used in the solution of large problems (with different special methods) are discussed in Section 5. Some concluding remarks are given in Section 6 .

3. Convergence of the iterative process (2.1)-(2.3)

In this section it is not assumed that matrix A is sparse, i.e. the results are also valid if this matrix is dense.

Consider the iterative process defined in Step 3 (see Section 2). Assume that (1.5) and (2.1)-(2.3) can be performed without rounding errors. Rewrite (1.2) in the following form

$$(3.1) \qquad s = c - B_1 y , \qquad s \in \mathbb{R}^{\bar{p}\times 1} , \qquad B_1^T s = 0 .$$

Then the following three theorems are needed before the discussion of the convergence of the iterative process (2.1)-(2.3).

<u>Theorem 3.1</u> <u>Let $\{y_i\}$ be the sequence of vectors calculated by (1.5) and (2.3) . Then</u>

$$(3.2) \qquad y_i = y + F^{i-j}(y_j - y) + \Big(\sum_{\nu=0}^{i-j-1} F^\nu\Big) Hs$$

<u>where $j < i$ is a fixed positive integer and it is assumed that $F^0 = I$.</u>

<u>Proof</u> It is readily seen that

$$(3.3) \qquad y_i - y = y_{i-1} + d_{i-1} - y = (y_{i-1} - y) + Hr_{i-1}$$

$$= (y_{i-1} - y) + H[B_1(y - y_{i-1}) + s] = (I - HB_1)(y_{i-1} - y) + Hs$$

$$= F(y_{i-1} - y) + Hs .$$

The assertion of Theorem 3.1 follows by the use of (3.3) for $y_{i-1}-y$, $y_{i-2}-y$, . . . , $y_{j+1}-y$.

□

<u>Theorem 3.2</u> <u>Let</u> $\{d_i\}$ <u>be the sequence of vectors found by (2.2) . Then</u>

(3.4) $d_i = F^{i-j}d_j$

<u>where</u> $j \leq i$ <u>is a fixed positive integer and it is assumed that</u> $F^0 = I$.

<u>Proof</u> Consider

(3.5) $d_i = d_{i-1}+(d_i-d_{i-1}) = d_{i-1}-HB_1(y_i-y_{i-1}) = (I-HB_1)d_{i-1}$

$= Fd_{i-1}$.

Use (3.5) for d_{i-1}, d_{i-2}, . . . , d_{j+1} . It is clear that the result will be (3.4) .

□

<u>Theorem 3.3</u> <u>Let</u> λ_i , <u>$i=1(1)\bar{q}$, be the eigenvalues of matrix F . Assume that</u>

(3.6) $|\lambda_1| \geq |\lambda_j|$, $j=2(1)\bar{q}$,

<u>and</u>

(3.7) $|\lambda_1| < 1$.

Then

(3.8) $$y = y_\nu + \sum_{i=\nu}^{\infty} d_i - (HB_1)^{-1}Hs$$

where ν is a fixed positive integer.

Proof It is clear that $\lim_{i\to\infty}(y_i)$ exists when (3.7) is satisfied (see e.g. Fadeev and Fadeeva [19], pp.111-114). From (3.2) the following relation can be obtained

(3.9) $$\lim_{i\to\infty}(y_i) = y + (HB_1)^{-1}Hs \; .$$

Use (2.3) and (3.4) . Then

(3.10) $$\lim_{i\to\infty}(y_i) = y_\nu + \sum_{i=\nu}^{\infty} d_i$$

and (3.8) can easily be found from (3.9) and (3.10) .

□

Corollary 3.1 Let (3.6) and (3.7) hold. Then the iterative process (2.1)-(2.3) is convergent to the true solution y of (1.2) only if one of the following three conditions is satisfied

(3.11) $s = 0$,

(3.12) $H = B_1^{\dagger}$,

(3.13) H is of the form $\bar{H}B_1^T$.

$\square$

Remark 3.1 It is obvious that the iterative process (2.1)-(2.3) will be convergent if (3.7) is replaced by

(3.14) $\|F\| < 1$

where $\|\bullet\|$ is any matrix norm induced by the vector norm used (see again [19], p.112).

Remark 3.2 Assume that B_μ $(\mu=1(1)k)$ is well scaled and that the drop-tolerance T_μ (used at stage μ of the general k-stage scheme) is chosen so that

(3.15) $T_\mu = w_\mu b_\mu$, $0 \le w_\mu < 1$, $\mu=1(1)k$,

where b_μ is the magnitude of the non-zero elements in B_μ . Assume that at stage μ $(\mu=1(1)k)$ of the general k-stage scheme the factorization (1.4) is obtained either by some version of the Gaussian elimination or by some orthogonal decomposition. Then

(3.16) $\|E_\mu\| \le f_\mu(m,n)\bar{\varepsilon}_\mu g_\mu(A)$, $\bar{\varepsilon}_\mu = \mathit{max}(\varepsilon, T_\mu/b_\mu)$,

where $f_\mu(m,n)$ is some function of m and n which depends on the method used at stage μ, ε is the machine accuracy and $g_\mu(A)$ is some function of the norm of matrix A corresponding to the norm used in the left-hand side of (3.16). For $T_\mu = 0$

explicit expressions for f_μ and g_μ for some special methods can be found e.g. in Wilkinson [43,44,45], Stewart [40], Björck [4] and Voevodin [42]. Denote

(3.17) $$f(m,n) = \max_{1\le\mu\le k} \{f_\mu(m,n)\},$$

(3.18) $$g(A) = \max_{1\le\mu\le k} \{g_\mu(A)\},$$

(3.19) $$\bar{g}(A) = \max_{1\le\mu\le k} \{\| (\prod_{i=1}^{k-1} Q_i D_i^\dagger) Q_k \bar{B}_k^\dagger P_k (\prod_{i=\mu}^{k-1} P_i^T C_i)^T P_\mu^T \| . \| Q_\mu^T (\prod_{i=1}^{\mu-1} Q_i D_i^T)^T \|\} .$$

Then

(3.20) $$\| F \| \le k f(m,n) \bar{\varepsilon} g(A) \bar{g}(A) , \qquad \bar{\varepsilon} = \max_{1\le\mu\le k} \{\bar{\varepsilon}_\mu\} .$$

In the case where $k=1$ and $T=0$ for many special methods $g(A)\bar{g}(A)$ is expressed by the condition number (corresponding to the matrix norm used) of matrix A (see again [43,44,45,40, 4,42]). Moreover, in the case where the spectral condition number $\kappa(A) = \| A \|_2 \| A^{-1} \|_2$ of matrix A is used for the methods from Example 2 and Example 6 Björck ([4], p. 163) has shown that $\kappa(A)$ can be replaced by

(3.21) $$\kappa'(A) = \inf_{D>0} \{\kappa(AD)\}$$

where D is a diagonal matrix and $D>0$ means that all diagonal elements of D are positive. Björck used the term "relevant" condition number for $\kappa'(A)$ in [4] .

Of course, (3.20) can not be used for a direct estimation of the norm of matrix F. Note that in the above references it is underlined that the values of $f(m,n)$ theoretically found (when $k=1$) are very crude and give a very great overestimate of the norm of F (this is especially true when m and/or n are large). However, at least in the cases where $g(A)\bar{g}(A)$ can be expressed as a function of the condition number (or the relevant condition number (3.21)) of matrix A, the bound (3.20) indicates an important relationship between the condition number (the relevant condition number) of matrix A and the drop-tolerance T. From (3.14), (3.16) and (3.20) it follows that: *the drop-tolerance can be chosen larger when the condition number (the relevant condition number) of matrix A is smaller.* This relationship will be verified (by numerical examples) in Section 5. It is necessary to emphasize here that the above rule restricts very strongly the use of large values for the drop-tolerance T when the normal equations (Example 1.2) are formed.

Remark 3.3 All results in this section are proved under the assumption that the computations with (1.5) and (2.1)-(2.3) can be performed without rounding errors. The experimental evidence shows that this is a realistic assumption because the computations connected with the back substitution are usually much more accurate than those with the decomposition (for several special methods this fact is emphasized e.g. by Wilkinson [44], Stewart [40] and Voevodin [42]). Note that this will be true to a greater degree when $T_\mu > \varepsilon b_\mu$ $(\mu=1(1)k)$ are used during (1.3). However, if matrix A is extremely ill-conditioned, then it may happen that the iterative process converges to a vector different from that in the right-hand side of (3.9), see [44,40] .

4. An experimental device for acceleration of the speed of convergence of the iterative process (2.1) - (2.3)

Let a sequence of problems (1.1) with the same matrix A and different vectors b be solved by some direct method followed by iterative refinement. Then Step 1 has to be performed only once (when the first problem of the sequence is solved), while the computations by (1.5) and (2.1)-(2.3) have to be carried out with each problem. Therefore it is very important to reduce the number, p-1 , of iterations in the process (2.1)-(2.3) because the computing time needed in the solution of each problem after the first one is, roughly speaking, proportional to p .

Assume again, as in Section 3, that the computations in (1.5) and (2.1)-(2.3) can be carried out without rounding errors. An experimentally found device, which sometimes accelerates the speed of convergence of the iterative process (2.1)-(2.3) and which has successfully been used in connection with the Gaussian elimination in Zlatev [48], will be generalized for the k-stage direct method in this section.

Theorem 4.1 Consider (1.2). Suppose that (3.7) is satisfied and let v_j (with $\|v_j\|=1$) be an eigenvector corresponding to λ_j $(j=1(1)\bar{q})$ of matrix F . Assume that v_j $(j=1(1)\bar{q})$ are linearly independent and that

(4.1) $|\lambda_1| > |\lambda_j|$, $j=2(1)\bar{q}$;

(4.2) $||c-s|| > 0;$

(4.3) $d_1 = \sum_{j=1}^{\bar{q}} a_j v_j$, a_j are constants, $a_1 \neq 0;$

(4.4) $\varepsilon^* > 0;$

(4.5) $d_i^* = \lambda_1^{i-1} a_1 v_1;$

(4.6) $\bar{c}_i = \sum_{j=2}^{\bar{q}} (\lambda_j/\lambda_1)^{i-1} a_j v_j$.

Then there exists an integer μ such that

(4.7) $||\bar{c}_i|| \leq \varepsilon^* |a_1|,$ for $i > \mu;$

(4.8) $(1-|\lambda_1|)^{-1} || d_\mu^* || \leq || y ||$;

(4.9) $|| y-y^* || \leq \varepsilon^* || y || + || (HB_1)^{-1} Hs ||;$

where

(4.10) $y^* = y_\mu + (1-\lambda_1)^{-1} d_\mu^*$.

Proof For $(HB_1)^{-1}Hs=0$ the theorem is proved in [48] (see Theorem 2.1 in [48]) . The proof for $(HB_1)^{-1}Hs \neq 0$ can be carried out in a similar way (using Theorem 3.1 - Theorem 3.3).

□

Theorem 4.1 states that if μ, d_μ^* and λ_1 are known and if $\| (HB_1)^{-1}Hs \| << \varepsilon^* \| y \|$, then y^* can be computed by (4.10) (instead of $y_{\mu+1}$ from (2.3)) so that a relative accuracy of ε^* will be achieved. However, the theorem is not constructive (since μ, d_μ^* and λ_1 are not known). Therefore the following result is more useful.

<u>Theorem 4.2</u> <u>Let all assumptions made in Theorem 4.1 be satisfied. Assume that</u>

(4.11) $|\lambda_1 - \tilde{\lambda}_1| \le \delta |\lambda_1|$, $0 \le \delta < 1$, δ <u>being given</u>;

(4.12) $\| F \| \le \tilde{q} < 1$;

(4.13) $|\tilde{\lambda}_1| < 1$.

<u>Then there exists an integer μ such that</u>

(4.14) $\| \bar{c}_i \| \le \delta \| d_1 \|$, <u>for</u> $i \ge \mu$;

(4.15) $\| y - \tilde{y}^* \| \le h\delta \tilde{q}^{\mu+1} \| y \| + \| (HB_1)^{-1}Hs \| + h\tilde{q}^{\mu}\delta \| FHs \|$

<u>where</u>

(4.16) $\tilde{y}^* = y_\mu + (1-\tilde{\lambda}_1)^{-1} d_\mu$;

(4.17) $h = [2 + (1-|\tilde{\lambda}_1|)^{-1}](1+\tilde{q})(1-|\lambda_1|)^{-1}$.

<u>Proof</u> For the case where any of the conditions (3.11)-(3.13) is satisfied the proof will be the same as the proof of Theorem 2.2 in [48]. A similar proof can be performed

when none of these conditions is satisfied. In the latter case the following equality

$$(4.18) \quad d_1 = F(I-F)y + FHs$$

should be used.

□

Note that in Theorem 4.2 d_μ (which is known) is used instead of d_μ^* and $\tilde{\lambda}_1$ (which is an approximation of λ_1 and can easily be found, e.g. by the power method) is used instead of λ_1 . The result is obviously weaker than that in Theorem 4.1 , however, it can be expected that if δ (which is a free parameter) is chosen sufficiently small, if λ_1 is not too close to one and if the last two terms in (4.15) are sufficiently small, then $\tilde{y}^*$ found by (4.16) will be a better approximation than $y_{\mu+1}$ found by (2.3). Unfortunately, this theorem is not constructive either (μ is not known). Nevertheless, a practical device can be based on this theorem in the following way.

Extrapolation device *Assume that* $\lambda_1^{(i)}$, *$i=2,3,\dots$, are found by the power method. Let the following relation hold*

$$(4.19) \quad |\lambda_1^{(\mu)}-\lambda_1^{(\mu+1)}| \le \tilde{\delta}|\lambda_1^{(\mu)}| \, .$$

Then the speed of convergence of the iterative process (2.1)-(2.3) will often be accelerated when the extrapolation

$$(4.20) \quad y_{\mu+1} = y_\mu + (1-\lambda_1^{(\mu)})^{-1} d_\mu \ .$$

is applied in the μ'*th iteration instead of* (2.3) .

This device is based on the expectation that if $\tilde{\delta}$ is chosen sufficiently small then the integer μ (for which (4.19) is satisfied) is larger than or equal to the corresponding integer μ in Theorem 4.2; thus the conditions of Theorem 4.2 hold and (4.20) will (hopefully) be better than the corresponding approximation obtained by (2.3). The experiments show that the device can efficiently be used in order to reduce the number of iterations, i.e. the expectation that the conditions of Theorem 4.2 hold is often satisfied. The only remaining problem in the implementation of the device is connected with the continuation of the computations after the application of (4.20). The following theorem is needed before the discussion of this problem.

<u>Theorem 4.3</u> <u>Let the conditions of Theorem 4.2 be satisfied. Assume that the computations must be continued after the use of (4.16). Denote</u> $y_{\mu+1}=\tilde{y}^*$ <u>and let</u> y_i <u>for</u> <u>$i>\mu+1$ be computed by the use of (2.3) Then</u>

$$(4.21) \quad d_i = (I-F)(y-y_{i-1}) + Hs$$

<u>and (for $i>\mu+1$)</u>

$$(4.22) \quad y_i - y = F^{i-\mu-1}(\tilde{y}^*-y) + (\sum_{j=0}^{i-\mu-2} F^j)Hs \ .$$

<u>Proof</u> For the case where one of the relations (3.11)-(3.1

is satisfied the proof is the same as the proof of Theorem 2.3 in [48]. If none of the relations (3.11)-(3.13) is satisfied then Theorem 4.3 can be proved in a similar way.

□

It is not clear what will happen if (4.20) is used successively several times. Theorem 4.3 indicates that y_i $(i>\mu+1)$ is a better approximation than $\tilde{y}^*$ if y_i is found by (2.3). This means that after the application of the extrapolation device it is necessary to ensure that (2.3) will be used in the next iterations (hoping that in this way the conditions of Theorem 4.3 will be satisfied). This can be done by reducing the control parameter $\tilde{\delta}$ (from (4.19)), say, by a factor 0.1.

An algorithm based on the above extrapolation device has been developed and tested by Zlatev [48] in the case where $m=n$ and the classical Gaussian elimination is used in the decomposition of large and sparse matrices. Similar rules have been used by Nielsen [31] in the case where $m=n$, the Gaussian elimination is used and matrix A is dense.

Note that the results in this section (and in Section 3) show that the solution found with iterative refinement may differ from the true solution when the norm of the residual vector s is large even if the iterative process is convergent. This is not a surprise (see e.g. the experimental results given in Björck [4]). This fact is explained in many papers by the appearance of the square of the condition number of matrix B_1 in the error estimation (see e.g. Golub and Wilkinson [25]). Of course, our results can also be explained by means of the condition number of matrix B_1. If B_1 is well conditioned then H is a good approximation of $B_1^\dagger$ and the terms containing s will be small. If B_1 is ill-conditioned then H is not a good approximation of $B_1^\dagger$ and the computed solution will differ considerably from the true solution.

5. Some numerical illustrations

Numerical experiments with the special methods discussed in Section 1 except the Peters-Wilkinson method have been carried out. There arise two difficulties when the Peters-Wilkinson method is used with some sparse technique. Assume that Gustavson's scheme (see [26,27]) is implemented (as is done, for example, in the code MA28, see [15,17]). Then after the first stage, see (1.18), the non-zero elements of $\bar{B}_1$ are stored in a one-dimensional array so that they are ordered by rows and in the rows first the non-zero elements of L_p and then (if the row number is smaller than n) the non-zero elements of U_p are stored. The first problem is: how to separate the elements of L_p and U_p and how to form $B_2=L_p^TL_p$ efficiently and without any use of extra storage? It is clear that if $m>>n$ and/or if L_p contains more than $\sqrt{n}$ non-zero elements per row in average (unfortunately the latter case occurs in practice; moreover, it is impossible to predict this in advance, i.e. before the beginning of Step 1) then it is better to store B_2 in a two-dimensional array (in the latter case this is so because a probabalistic analysis given in Björck [5] and Björck and Elfving [6] shows that B_2 will often be dense). The second problem is: how to decide when B_2 should be stored in a one-dimensional array and when in a two-dimensional array? The reason that the Peters-Wilkinson method has not been tested is that we have not succeeded in finding satisfactory answers to the above questions (which, of course, does not mean that these answers do not exist). Note that the second problem arises in the solution of (1.1) by forming the normal equations also. Therefore this method has been tested only with problems with relatively small n ($n \leq 75$) and the coefficient matrix of the normal system has always been stored in a two dimensional array.

It is necessary to emphasize that the comparison of the different methods is beyond the scope of this paper. In connection with the numerical results the following topics will be dicussed: (i) the effect of the use of a large drop-tolerance (in Step 1 of the general k-stage scheme) and iterative refinement (the additional step, Step 3, of the general k-stage scheme), (ii) the choice of the drop-tolerance, (iii) the dependence of the drop-tolerance on the condition number of matrix B_1 , (iv) the effect of the implementation of the extrapolation device.

5.1. The effect of the use of a large drop-tolerance and iterative refinement.

In the case where $m=n$ and the Gaussian elimination is used (Example 1.1) 192 linear systems in the range $250 \leq n \leq 1000$ have been tested in [48]. In these experiments the use of a large drop-tolerance T and iterative refinement is compared with the simple use of Step 1 and Step 2 (this mode will be called a direct solution, DS). The code Y12M (see[56]) have been used in this comparison. Some NAG(*) subroutines (which perform the DS mode only), namely, F04AXE and F01BRE, (based on ideas described in [15]), have

(*) NAG: Numerical Algorithms Group, Banbury Road 7,Oxford also been used in some examples. A great reduction in the computing time and, sometimes, in the storage used has been obtained by the use of a large drop-tolerance and iterative refinement. The reduction in the computing time is as much as 10 times for some examples. The greatest reductions in the storage have been observed with problems which produced many fill-ins, i.e. just when such reductions are most needed). Similar results have been obtained when the augmented method (Example 1.3), see [48], and some orthogonalization methods (Example 1.5 and Example 1.6 , some numerical results are given in [53]), have been tested as above.

5.2. The choice of the drop-tolerance. The efficiency of the use of iterative refinement depends essentially on the choice of the drop-tolerance T . Two practical rules which can successfully be used in the choice of T are described in [48]. These rules can be used with any method in the general k-stage scheme. Therefore they are briefly sketched below.

Rule 1 Assume that: (i) matrix B_1 is not too ill-conditioned and (ii) all numbers $b_i = \max_{1 \le j \le q}(|b_{ij}|)$, $i=1(1)\bar{p}$, $b_{ij} \in B_1$, have the same magnitude. Denote $a = \min_{1 \le i \le p}(a_i)$. Then $T \in [10^{-5}a, 10^{-3}a]$ will often be a good choice for the drop-tolerance. The assumption (ii) is not very stringent when row scaling is allowed (but note that if row scaling is performed, then our experiments show that it is better to restrict the interval for T to $[10^{-5}a, 10^{-4}a]$). If $k>1$ then the use of different values of T at the different stages may be more appropriate (however, the same T can be used in the method described in Example 1.6 because in the second stage of this method the matrices which are already computed during the first stage are used). In package Y12M there is an option where a is automatically computed and the drop-tolerance is set equal to ta (t can be chosen by the user; the recommended values are in the interval $[10^{-5},10^{-3}]$).

<u>Rule 2</u> Consider a set of linear problems (1.1) whose coefficient matrices are

$$(5.1) \qquad \underbrace{A_1,A_1,\ldots,A_1}_{j_1 \text{ times}}, \quad \underbrace{A_2,A_2,\ldots,A_2}_{j_2 \text{ times}}, \quad \ldots , \quad \underbrace{A_i,A_i,\ldots,A_i}_{j_i \text{ times}}$$

Assume that all matrices have the same sparsity pattern (A and B have the same sparsity pattern if ($a_{ij} \in A$, $b_{ij} \in B$) $a_{ij} \neq 0 \Rightarrow b_{ij} \neq 0$ and $b_{ij} \neq 0 \Rightarrow a_{ij} \neq 0$, $i=1(1)m$, $j=1(1)n$). Let the corresponding non-zero elements of A_μ and $A_{\mu+1}$ $(\mu=1(1)i-1)$ not differ too much. Set a large initial value of T (e.g. $T_{\text{initial}} = 10^{-1}a$, where a is as in Rule 1) before the beginning of the computations. If the error requirement (prescribed in advance, see Requirement 2 in Section 2) is not satisfied after the solution of the first problem with matrix A_μ $(\mu=1(1)i)$, then decrease the drop-tolerance (by some factor $C_T<1$) and solve the first problem with matrix A_μ again. After $J-1$ (where J is a fixed positive integer) multiplications of T by C_T perform a last attempt to satisfy the error requirement by $T=0$. In this rule some extra work at the beginning is accepted, hoping to obtain very good results in the remaining part of the computations and, thus, to reduce the total work. This rule has been used in [37] where some chemical problems which lead to large linear systems of ordinary differential equations are discussed. The discretization of these systems by two-stage diagonally implicit

Runge-Kutta methods leads to large sequences (5.1) (with $j_\mu = 2$, $\mu=1(1)i$) of linear systems of algebraic equations. A reduction in the computing time by a factor of 1/4 and a considerable reduction in storage have been observed in an example with $m=n=255$; see more details in [37,48].

<u>5.3. The dependence of the choice of a drop-tolerance on the condition number of matrix B_1.</u> The results found in Section 3 indicate that the choice of the drop-tolerance T depends on the condition number of matrix B_1. Many experiments have been carried out in order to confirm this conclusion. Some results are given in Table 1. The estimations of the condition numbers, COND's, have been found by a subroutine proposed by Forsythe, Malcolm and Moler [20]. The matrices have been generated by a special subroutine, MATRF2, developed at the Institute for Numerical Analysis, Technical University of Denmark [50]. Matrices which depend on five parameters can be generated by this subroutine as follows: (i) by the parameter M the number of rows in the desired matrix is specified, (ii) by the parameter N the number of columns in the desired matrix is specified, (iii) by the parameter C the positions of certain non-zero elements are determined, (iv) by the parameter INDEX the average number of non-zero elements per row is specified, (v) by the parameter ALPHA the ratio $max(|a_{\mu\nu}|)/min(|a_{\mu\nu}|)$ (where $a_{\mu\nu} \in A$, $a_{\mu\nu} \neq 0$) can be varied. In this experiment the first four parameters have been fixed ($M = N = 22$, $C = 11$, $INDEX = 2$), while the condition number of the matrix has been changed by ALPHA. Two codes, SIRSM (based on Gaussian elimination, see [51,52]) and LLSS01 (based on orthogonal transformations, see [53]), have

Tolerance	Gaussian elimination			Orthogonal transformations		
T	ALPHA	COND	$\lVert x-\bar{x} \rVert_\infty$	ALPHA	COND	$\lVert x-\bar{x} \rVert_\infty$
0.0	2^{24}	4.43E+14	8.46E-14	2^{15}	1.69E+12	7.09E- 7
10^{-4}	2^{16}	6.75E+12	2.79E-11	2^{9}	4.02E+ 8	7.54E-10
10^{-3}	2^{13}	1.01E+11	6.51E-12	2^{7}	2.35E+ 7	4.66E-10
10^{-2}	2^{10}	1.63E+ 9	3.38E-12	2^{4}	2.43E+ 5	5.17E-12

Table 1

The dependence of the drop-tolerance T on the condition number of matrix B_1 (the test-matrices have been generated by subroutine MATRF2, COND is the evaluated condition number by the subroutine proposed by Forsythe et al. [20]).

been tested. At the beginning ALPHA=1 has been used with all drop-tolerances. If the code succeeded in the solution then ALPHA was multiplied by 2 (until the code fails). This means that under "ALPHA" the last value with which the code has successfully solved the problem is given in Table 1 (the estimation of the condition number of this last matrix and the largest error in the last solution are also given in Table 1). It is seen that the behaviour of the codes is similar: with large values of ALPHA (i.e. when the condition number of the matrix is large) small values of the drop-tolerance should be chosen. Note that the iterative process converges even if the estimation of the condition number is much larger than $1/\varepsilon$ (where ε is the machine accuracy), moreover, this is also true when $T>>\varepsilon$.

Note too that the tendencies described above have been observed in many other experiments. Finally, it should be mentioned that all inner products have been accumulated in double precision by the codes used in this experiment (see more details in Section 6.3).

5.4. The use of the extrapolation device. The efficiency of the device described in Section 4 has been verified by many experiments. In [48] 96 systems (solved by the Gaussian elimination) have been tested with four different drop-tolerances. The version with the extrapolation device gave better results than the version without the extrapolation device (the number of iterations has been reduced by 2-3 , in average, when the extrapolation device has been used; note that this means that very often the number of iterations has been reduced by a factor 1/2). If only one problem (1.1) is to be solved then this reduction of the number of iterations does not lead to a great reduction in the computing time (a reduction of 2% - 6% has been found in [48]). However, if a sequence of problems (whose matrices are as in (5.1)) is to be solved by iterative refinement and if i is small but the sum $j_1+j_2+\ldots+j_i$ is large, then the extrapolation device will also give a great reduction in the computing time. A practical problem (arising from modelling of heat accumilation in underground tanks) where this situation takes place ($i=12$, $j_1+j_2+\ldots+j_{12}=1010$, $m=n=470$) is described in Zlatev and Thomsen [55], p. 1058, Example 3.

6. Some remarks

6.1. On the choice of a pivotal strategy. We are not able to propose some general rules which can be used in the choice of a pivotal strategy with all special methods in the k-stage scheme. The pivotal strategy may improve the method considerably. Both the storage requirements, the computing time and the accuracy are dependent on the pivotal strategy when some sparse matrix technique is applied. Sometimes even the details can be very important. An example, which shows that a small change in the pivotal strategy may lead to a great improvement of the accuracy achieved, is briefly discussed below (more details can be found in [49,48]).

Let $m=n$ and let the Gaussian elimination be used. Look at (1.14). Before the beginning of the computations with any s $(s=1(1)n-1)$ the pivotal element, $a_{ss}^{(s)}$, should be determined. Consider

$$(6.1)\qquad A_s = \{a_{ij}^{(s)} \;/\; s\le i \;,\; s\le j \;\} ,$$

$$(6.2)\qquad I_s = \{i_m \;/\; m=1(1)p(s),\;\; 1\le p(s)\le n-s+1,\;\; s\le i_m\le n \},$$

$$(6.3)\qquad (i_k \in I_s) \wedge (i_q \in I_s) \wedge (k<q) \;\Rightarrow\; r(i_k,s) \le r(i_q,s) ,$$

$$(6.4)\qquad (i \notin I_s) \wedge (s\le i\le n) \;\Rightarrow\; r(i_{p(s)},s) \le r(i,s) ,$$

$$(6.5)\qquad B_s = \{a_{ij}^{(s)} \in A_s \;/\; |a_{ij}^{(s)}|u > \max_{s\le k\le n} |a_{ik}^{(s)}|,\; i \in I_s,\; u>1\},$$

$$(6.6)\qquad M_{ijs} = [r(i,s)-1][c(j,s)-1] ,$$

$$(6.7) \qquad M_s = \min_{\substack{s\le i,j\le n\\ a_{ij}^{(s)}\in B_s}} M_{ijs} ,$$

$$(6.8) \qquad C_s = \{a_{ij}^{(s)} \in B_s \; / \; M_{ijs} = M_s\} ,$$

where $r(i,s)$ and $c(j,s)$ are the numbers of these non-zero-elements in row i and column j which belong to A_s. Then any element of C_s can be chosen as pivotal. This pivotal strategy is called a GMS (generalized Markowitz strategy) in [49]. Note that the original Markowitz strategy can be found from (6.1)-(6.8) for $u = \infty$ and $p(s)=n-s+1$ (see [29]). The pivotal strategies in the codes described in [11,15,35] can be found from (6.1)-(6.8) with $p(s)=n-s+1$ and $u \in [4,10]$. Zlatev [49] proposed choosing the largest in absolute value element in C_s as pivotal. This strategy is called an IGMS (improved generalized Markowitz strategy) in [49] and has been implemented in [52,54,56] with $u \in [4,16]$ and $p(s) \le 3$. Since the elements of C_s are also elements of the stability set B_s the change made to obtain an IGMS seems to be not very important for the accuracy of the results. However, in [49] it is proved that there exist classes of matrices for which <u>any</u> IGMS will guarantee stable results while the GMS's may cause numerical instability. Moreover, in [48,49] there are given some examples where the codes based on GMS's produce much poorer results than the codes based on IGMS's. The same situation is illustrated in Table 2. The code Y12M (based on an IGMS, see [56]) and the NAG subroutines F01BRE, F04AXE (based on a GMS, see [15]) are used in this experiment. More details about the matrices used can be found in

n	F01BRE + F04AXE	Y12M-without IR	Y12M - with IR
650	1.42E-2	1.06E-5	2.98E-8
700	2.87E-2	1.24E-5	1.49E-8
750	1.03E-2	1.38E-4	2.98E-8
800	4.13E-1	1.60E-5	5.96E-8
850	4.74E-3	1.79E-5	2.24E-7
900	7.34E-1	2.50E-5	9.24E-7
950	2.61E-2	2.65E-5	1.83E-6
1000	2.02E-1	2.98E-5	8.24E-7

Table 2

Comparison of the accuracy of the computed solution by Gaussian elimination ($m=n$) with the two different pivotal strategies described in Section 6.1.

[48 49]. It is seen (from Table 2) that small changes in the pivotal strategy may cause great differences in the performance of the method. The accuracy found by the use of iterative refinement and $T = 10^{-2}$ is also given in Table 2.

Discussion of different pivotal strategies can be found in Duff [14] for the Gaussian elimination and in Duff [13] for the Givens orthogonalization.A very interesting suggestion concerning the pivotal strategies with sparse implementation of the Peters-Wilkinson method is given by Björck in [5].

Finally, note that if iterative refinement (combined with a large drop-tolerance) is used, then the particular method seems to be not very sensitive to the pivotal strategy chosen (in [37] good results are reported even in the case where the pivotal interchanges are suppressed).

6.2. On the scaling of the problem. It is well known that if the coefficient matrix is not well scaled (i.e. if the elements differ significantly in magnitude) then scaling may give good results. Unfortunately, no general rule, which is efficient for any matrix, is known (see e.g. Forsythe and Moler [21], Reid [34], Stewart [40]).

In our experiments with matrices generated by subroutine MATRF2 ([50], see also Section 5.3) with large values of ALPHA the simple equilibration (scale first the rows then the columns) gave no improvement. The more sophisticated algorithm based on ideas proposed by Hamming [28] and described in details in Curtis and Reid [12] (note that similar ideas have also been exploited by Fulkerson and Wolfe [22]) performed slightly better for the same matrices.

Duff et al. [18] proposed accepting rounding errors during the scaling process This means that the scaling factors: $r_i=\bar{r}_i\beta^{\rho_i}$ and $c_j=\bar{c}_j\beta^{\sigma_j}$ ($i=1(1)m$, $j=1(1)n$, β is the base of the floating point representation in the machine, $1/\beta \le \bar{r}_i,\bar{c}_j \le 1$) are stored and used in the computations instead of the integer exponents ρ_i and σ_j . Two versions of SIRSM (a code based on the use of Gaussian elimination and iterative refinement, see [52 53]) have been tested. In the first, SIRSMA , the exponents of the scaling factors have been stored and used. In the second, SIRSMB , the scaling factors themselves have been stored and used. Matrices generated by MATRF2 and by large values of ALPHA have been used in this experiment. The accuracy of the results have been approximately the same in all examples (except one, where SIRSMB failed because the iterative process was not convergent). However, the total number of iterations used by SIRSMB was larger. Therefore the use of the exponents of the scaling factors seems to be more efficient in connection with iterative refinement. It should be mentioned that in our experiment (performed on an IBM 360/165 computer) the simple equilibration has been used, while the object of the discussion in [18] is the algorithm proposed in [12] .

Our experiments show that the iterative refinement without scaling gives good results in spite of the fact that the coefficient matrix is badly scaled (see also [37]). This confirms some results given in Skeel [38,39] .

<u>6.3. On the accumulation of the inner products in double precision.</u> Assume that: (i) the vectors r_i, d_i and y_i (see (1.5) and (2.1)-(2.3)) are stored in double precision and all inner products in (1.5) and (2.1)-(2.3) are accumulated in double precision (where the length of the real numbers is n_2 binary digits); (ii) the non-zero elements of all matrices used in the k-stage scheme are stored in single precision and all computations in Step 1 are carried out in single precision (where the length of the real numbers is n_1 binary digits); (iii) $n_2 \geq 2n_1$ and (iv) the iterative process (2.1)-(2.3) is convergent. Then about n_1 binary digits will normally be gained in the iterative process (i.e. the same accuracy as if $2n_1$ digits had been used throughout the computations is normally achieved). This result has been shown by Björck [1,2,4] for an algorithm developed by Björck and Golub [7]. Our experiments indicate that this is true for all special methods discussed in Section 1 (except the Peters-

Tolerance	0.0	10^{-4}	10^{-3}	10^{-2}
ALPHA	2^{16}	2^{16}	2^{13}	2^{9}

Table 3

The same experiment as in Section 5.3. The residual vectors have been accumulated in double precision and then rounded to single precision. The results should be compared with those in Table 1 found by the Gaussian elimination.

Wilkinson method which has not been tested). Note that the accumulation of the inner products from (1.5) and (2.1)-(2.3) in double precision causes a considerable increase of the number of iterations (3-4 iterations more per problem in average). Nevertheless, in our opinion, this option is better than the classical mode where the residual vectors, r_i, are accumulated in double precision and then rounded to single precision . Assume that the coefficient matrix is badly scaled and/or ill-conditioned. In our experiments with such matrices the option advocated by Björck gave much more reliable error estimations (see the estimations of the accuracy given in [53]). Moreover, the experiments show a tendency that the subroutines based on this option are more robust (compare the results in Table 3 found by a subroutine based on the classical option and Gaussian elimination with the corresponding results in Table 1).

6.4. On the condition rank(A)=n. When the problem is large it is difficult to establish the validity of this condition. Unfortunately, this is not the whole story. Let $r < n$,

$r \in \mathbb{N}$. Then the method under consideration will perform badly not only if rank(A)=r but also if matrix A is close to a matrix $\bar{A}$ with rank($\bar{A}$)=r (see Wilkinson [45]). Different tests are used in the code Y12M ([56]) in order to detect the above situation (call it "rank degeneracy"). There is no guarantee, however, that the rank degeneracy will always be detected (though, deducing from our experiments, it may be expected that this will very often happen). Note that, a perfectly reliable determination of the numerical rank can be obtained only by the singular value decomposition (proposed by Golub and Kahan [23] in 1965; see also Björck [5], Stewart [40] and Wilkinson [46]), while the simpler factorizations are unreliable in exposing near rank degeneracy (see Wilkinson [45], p. 19). A computing procedure which performs a singular value decomposition for dense matrices is given by Golub and Reinsch [24]. Unfortunately, we can not see how the ideas discussed in Section 2 can be applied in an attempt to perform efficiently the singular value decomposition in the case where A is large and sparse.

6.5. On the problems (1.2) with large residual vectors.

Consider the case where the residual vector s (see (3.1)) satisfies: $s \neq 0$ and $\| s \|$ is large. Then it may be more profitable to attempt to correct both the residual vector s and the solution vector y of (1.2), see e.g. Björck [4]. The device discussed in this paper, Step 3 , appears also in Golub and Wilkinson [25]. This device has been chosen because in this way a simple treatment of the two cases, $s=0$ and $s \neq 0$, can be performed simultaneously.

R E F E R E N C E S

1. Å. Björck: Iterative refinement of linear least squares problems I . BIT 7(1967), 257-278.

2. Å. Björck: Iterative refinement of linear least squares problems II . BIT 8(1968), 1-30.

3. Å. Björck: Methods for sparse linear least squares problems. In: "Sparse Matrix Computations" (J. Bunch and D. Rose, eds), 179-199. Academic Press, New York, 1976.

4. Å. Björck: Comment on the iterative refinement of least squares solutions. J. Amer. Stat. Assoc. 73(1978), 161-166.

5. Å. Björck: Numerical algorithms for linear least squares problems. Report No. 2/78, Department of Mathematics, University of Trondheim, Trondheim, Norway, 1978.

6. Å. Björck and T. Elfving: Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. BIT 19(1979), 145-163.

7. Å. Björck and G. H. Golub: ALGOL programming, contribution No. 22: "Iterative refinement of linear least squares solutions by Householder transformations". BIT 7(1967), 322-337.

8. R. K. Brayton, F. G. Gustavson and R. A. Willoughby: Some results on sparse matrices. Math. Comp. 24(1970), 937-954.

9. R. J. Clasen: Techniques for automatic tolerance in linear programming. Comm. ACM 9(1966), 802,803.

10. A. K. Cline, C. B. Moler, G. W. Stewart and J. H. Wilkinson: An estimate for the condition number of a matrix. SIAM J. Numer. Anal. 16(1979), 368-375.

11. A. R. Curtis and J. K. Reid: The solution of large sparse unsymmetric systems of linear equations. J. Inst. Math. Appl. 8(1971), 344-355.

12. A. R. Curtis and J. K. Reid: On automatic scaling of matrices for Gaussian elimination. J. Inst. Math. Appl. 10(1972), 118,124.

13. I. S. Duff: Pivot selection and row ordering in Givens reduction of sparse matrices. Computing 13(1974), 239-248.

14. I. S. Duff: A survey of sparse matrix research. Report No. CSS28, A.E.R.E., Harwell, England, 1976.

15. I. S. Duff: MA28 - a set of FORTRAN subroutines for sparse unsymmetric matrices. Report R8730, A.E.R.E., Harwell, England, 1977.

16. I. S. Duff and J. K. Reid: A comparison of some methods for the solution of sparse overdetermined systems of linear equations. J. Inst. Math. Appl. 17(1976), 267-280.

17. I. S. Duff and J. K. Reid: Some design features of a sparse

matrix code. ACM Trans. Math. Software 5(1979), 18-35.

18. I. S. Duff, J. K. Reid, N. Munksgaard and H. B. Nielsen: Direct solution of sets of linear equations whose matrices are sparse, symmetric and indefinite. J. Inst. Math. Appl. 23(1979), 235-250.

19. D. K. Fadeev and V. N. Fadeeva: Computational methods of linear algebra. Freeman, San Francisco and London, 1963.

20. G. E. Forsythe, M. A. Malcolm and C. B. Moler: Computer methods for mathematical computations. Prentice-Hall, Englewood Cliffs, N. J., 1977.

21. G. E. Forsythe and C. B. Moler: Computer solution of linear algebraic equations. Prentice-Hall, Englewood Cliffs, N. J., 1967.

22. D. R. Fulkerson and P. Wolfe: An algorithm for scaling matrices. SIAM Review 4(1962), 142-146.

23. G. H. Golub and W. Kahan: Calculating the singular values and pseudo inverse of a matrix. J. SIAM Numer. Anal., Ser. B, 2(1965), 205-224.

24. G. H. Golub and C. Reinsch: Singular value decomposition and least squares solutions. Numer. Math. 14(1970), 403-420.

25. G. H. Golub and J. H. Wilkinson: Note on the iterative refinement of least squares solution. Numer. Math. 9(1966),

139-148.

26. F. G. Gustavson: Some basic techniques for solving sparse systems of linear equations. In: "Sparse Matrices and Their Applications" (D.J.Rose and R.A.Willoughby, eds), 41-52. Plenum Press, New York, 1972.

27. F. G. Gustavson: Two fast algorithms for sparse matrices: multiplication and permuted transposition. ACM Trans. Math. Software 4(1978), 250-269.

28. R. W. Hamming: Introduction to applied analysis. McGraw-Hill, New York, 1971.

29. H. M. Markowitz: The elimination form of the inverse and its applications to linear programming. Management Sci. 3(1957), 255-269.

30. E. H. Moore: On the reciprocal of the general algebraic matrix. Bull. Amer. Math. Soc. 26(1919-1920), 394-395.

31. H. B. Nielsen: Iterative refinement. Report No. 76-02, Institute for Numerical Analysis, Technical University of Denmark, Lyngby, Denmark, 1976.

32. R. Penrose: A generalized inverse for matrices. Proc. Cambridge Phil. Soc. 51(1955), 506-513.

33. G. Peters and J. H. Wilkinson: The least squares problem and pseudoinverses. Computer J. 13(1970), 309-316.

34. J. K. Reid: Sparse matrices and decomposition with application to the solution of algebraic and differential equations. In: "Decomposition of Large-Scale Problems" (D.M.Himmelblau, ed.), 57-69. North-Holland Publishing Company, Amsterdam, 1973.

35. J. K. Reid: Fortran subroutines for handling sparse linear programming bases. Report R8269, A.E.R.E., Harwell, England, 1976.

36. J. K. Reid: Solution of linear systems of equations: direct methods (general). In: "Sparse Matrix Technique" (V.A.Barker, ed.), Lecture Notes in Mathematics 572, 102-129. Springer, Berlin, 1977.

37. K. Schaumburg, J. Wasniewski and Z. Zlatev: On the use of sparse matrix technique in the numerical integration of linear ordinary differential equations. Report No. 79/9, The Regional Computing Centre at the University of Copenhagen (RECKU), Copenhagen, Denmark, 1979.

38. R. D. Skeel: Gaussian elimination and numerical instability. Report No. UIUCDCS-R-77-862, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1977.

39. R. D. Skeel: Iterative refinement implies numerical stability for Gaussian elimination. Manuscript, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1978.

40. G. W. Stewart: Introduction to matrix computations. Academic Press, New York, 1973.

41. R. P. Tewarson: Sparse matrices. Academic Press, New York, 1973.

42. V. V. Voevodin: Computational bases of the linear algebra. Nauka, Moskow, 1977 (in Russian).

43. J. H. Wilkinson: Rounding errors in algebraic processes. Prentice-Hall, Englewood Cliffs, N.J., 1963.

44. J. H. Wilkinson: The algebraic eigenvalue problem. Oxford University Press, London, 1965.

45. J. H. Wilkinson: Some recent advances in numerical linear algebra. In: "The State of the Art in Numerical Analysis" (D.A.H.Jacobs, ed.), 3-53. Academic Press, New York, 1977.

46. J. H. Wilkinson: Singular value decomposition - basic aspects. In: "Numerical Software - Needs and Availability" (D.A.H.Jacobs, ed.), 109-135. Academic Press, New York, 1978.

47. P. Wolfe: Error in the solution of linear programming problems. In: "Error in Digital Computation" (L.B.Rall, ed.), Vol. 2, 271-284. Wiley, New York, 1965.

48. 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 Univer-

sity , Copenhagen, Denmark, 1979.

49. Z. Zlatev: On some pivotal strategies in Gaussian elimination by sparse technique. SIAM J. Numer. Anal. 17(1980), No. 1 (to appear).

50. Z. Zlatev: On testing a subroutine for computing the pseudo-inverse solution of large linear problems by direct methods, (in preparation).

51. Z. Zlatev and H. B. Nielsen: Preservation of sparsity in connection with iterative refinement. Report No. 77-12, Institute for Numerical Analysis, Technical University of Denmark, Lyngby, Denmark, 1977.

52. Z. Zlatev and H. B. Nielsen: SIRSM - a package for the solution of sparse systems by iterative refinement. Report No. 77-13, Institute for Numerical Analysis, Technical University of Denmark, Lyngby, Denmark, 1977.

53. Z. Zlatev and H. B. Nielsen: Least squares solution of large linear problems. Report No. 79-06, Institute for Numerical Analysis, Technical University of Denmark, Lyngby, Denmark, 1979.

54. Z. Zlatev and P. G. Thomsen: ST - a FORTRAN IV subroutine for the solution of large systems of linear algebraic equations with real coefficients by use of sparse technique. Report No. 76-05, Institute for Numerical Analysis, Technical University of Denmark, Lyngby, Denmark, 1976.

55. Z. Zlatev and P. G. Thomsen: Application of backward differentiation methods to the finite element solution of time dependent problems. Int. J. num. Meth. Engng 14(1979), 1051-1061.

56. Z. Zlatev and J. Wasniewski: Package Y12M - solution of large and sparse systems of linear algebraic equations. Preprint Series No. 24 1978, Mathematics Institute, University of Copenhagen, Copenhagen, Denmark, 1978.