On the block implicit LU algorithm for linear systems of equations

Elena Bodon
ON THE BLOCK IMPLICIT LU ALGORITHM FOR LINEAR SYSTEMS OF EQUATIONS

ELENA BODON
University of Bergamo
Piazza Rosate 2, I-24129 Bergamo, Italy
E-mail:emilio@unibg.it

[Received November 30, 2000]

Abstract. We theoretically describe different implementations of the block version of the implicit LU algorithm in the ABS class. We also consider the special block ABS algorithm proposed in [3], showing that it corresponds to a version of the block implicit LU algorithm, with blocksize equal to two, when the coefficient matrix is diagonally dominant. Numerical experiments show that the different block implementations have similar accuracy.

Mathematical Subject Classification: 65F05
Keywords: linear equations, ABS methods, implicit LU algorithm

1. Introduction

Define the linear system of equations

\[ Ax = b , \]

where \( A = (a_1, \ldots, a_m)^T : a_i, x \in \mathbb{R}^n : b \in \mathbb{R}^m : m \leq n. \)

The unscaled version of the ABS algorithm [2] for solving system (1.1) is given by the following procedure.

Algorithm (a)

Let \( x_1 \in \mathbb{R}^n \) be arbitrary, \( H_1 \in \mathbb{R}^{n \times n} \) arbitrary nonsingular.

(●) For \( i = 1 \) to \( m \):

Compute the \( i \)-th component of the residual vector in \( x_i \)

\[ \tau_i = a_i^T x_i - b^T e_i , \]

where \( e_i \) is the unit vector in \( \mathbb{R}^n \), and compute the vector

\[ s_i = H_i a_i . \]

If \( s_i = 0 \) and \( \tau_i = 0 \), set \( x_{i+1} = x_i ; H_{i+1} = H_i \) and go to (●).

If \( s_i = 0 \) and \( \tau_i \neq 0 \) stop, the system is incompatible.
Compute the search vector
\[ p_i = H_i^T z_i , \]  
where \( z_i \in \mathbb{R}^n \) is arbitrary save for the condition \( a_i^T H_i z_i \neq 0 \).

Update the approximation to the solution
\[ x_{i+1} = x_i - \alpha_i p_i , \]  
where the stepsize \( \alpha_i \) is given by
\[ \alpha_i = \frac{\tau_i}{\delta_i} = \frac{a_i^T p_i}{\delta_i} . \]

If \( i = m \) stop, \( x_{m+1} \) is a solution, otherwise update matrix \( H_i \) by
\[ H_{i+1} = H_i - H_i a_i w_i^T H_i , \]  
where \( w_i \in \mathbb{R}^n \) is arbitrary save for the condition \( w_i^T H_i a_i = 1 \).

We list some properties of algorithm (a) which will be used later (see [2] for the proofs and other properties):

(P1) \( H_i a_j = 0, j = 1, ..., i - 1 \) (the vectors \( a_j \) span the null space of \( H_i \)).

(P2) From (P1) we have: \( a_j \) is linearly independent of \( a_1, ..., a_{j-1} \) if \( H_i a_j \) is linearly independent of \( H_i a_1, ..., H_i a_{j-1} \): \( i < j \).

(P3) Define \( A_i = (a_1, ..., a_i) \); \( P_i = (p_1, ..., p_i) \); \( Q_i = (H_i^T w_1, ..., H_i^T w_i) \); \( S_i = (s_1, ..., s_i) \); then the following relations hold: \( A_i^T P_i = L_i' \); \( A_i^T Q_i = L_i' \); \( A_i^T H_i^T = L_i' S_i^T \), where \( L_i' \); \( L_i' \) are lower triangular matrices.

(P4) The arbitrary parameter \( H_1 \) is redundant, since algorithm (a) can be defined in terms of \( H_1 = I \); \( z_i = H_i^T z_i \); \( w_i = H_i^T w_i \), where \( I \) is the unit matrix in \( R^{n \times n} \).

(P5) The arbitrary parameters \( z_i \); \( w_i \), are redundant, since the sequences (1.2) and (1.3) can be defined in terms of only one parameter \( z_i \) or \( w_i \).

(P6) The matrix \( H_{i+1} \) (1.4) can be updated in the form \( H_{i+1} = H_1 - S_i Q_i^T \). From (P3) it holds that \( H_{i+1} = H_1 - H_1 A_i L_i'^{-T} P_i^T \), and from (P4) and (P5) it holds that \( H_{i+1} = I - A_i L_i'^{-T} P_i^T \).

(P7) The solution of the system \( A_i^T X = I_i \), where \( X \in \mathbb{R}^{n \times i} \), \( I_i \) is the unit matrix in \( R^{i \times i} \), given by algorithm (a) can be also written in the form \( X = P_i L_i'^{-1} \).

(P8) The representation of all solutions of the first \( i \) equations has the form
\[ x' = x_{i+1} + H_{i+1}^T t \]  
with \( t \in \mathbb{R}^n \) arbitrary.
2. The block ABS algorithm

The block ABS algorithm, due to Abaffy and Galántai [1] for the scaled ABS class, see also [2], and further developed in several papers by Galántai, is a block form of the ABS algorithm. The block unscaled version of algorithm (a) is given by the following procedure.

Assume for simplicity that $A$ is full rank. Let $m, m_2, ..., m_j$ be positive integers such that $m_1 + m_2 + ... + m_j = m$. Set $A_k = (a_{m_1 + m_2 + ... + m_{k-1} + 1}, a_{m_1 + m_2 + ... + m_k})$, $b_k = (b^{T} e_{m_1 + m_2 + ... + m_{k-1} + 1}, b^{T} e_{m_1 + m_2 + ... + m_k})$, $A_k^T$ is the $k$-th block of $m_k$ rows of $A$, $b_k$ is the $k$-th block of $m_k$ components of $b$.

Let $x_1 \in R^n$ be arbitrary, and $H_1 \in R^{n \times n}$ arbitrary nonsingular.

For $k = 1$ to $j$:

Compute the matrix $P_k \in R^{n \times m_k}$ by

$$ P_k = H_k^T Z_k,$$

where $Z_k \in R^{n \times m_k}$ is arbitrary full rank save for the condition $A_k^T H_k^T Z_k$ nonsingular.

Update the approximation to the solution by

$$ x_{k+1} = x_k - P_k d_k, \quad (2.1) $$

where $d_k$ is the unique solution of the nonsingular system

$$ A_k^T P_k d_k = r_k, \quad (2.2) $$

and $r_k = A_k^T x_k - b_k$ is the $k$-th block of $m_k$ components of the residual in $x_k$.

If $k = j$ stop, $x_{j+1}$ solves system (1.1), otherwise update matrix $H_k$ by

$$ H_{k+1} = H_k - H_k A_k W_k^T H_k, $$

where $W_k \in R^{n \times m_k}$ is arbitrary full rank save for the condition

$$ W_k^T H_k A_k = I_{m_k}. \quad (2.3) $$

The properties of block algorithm, similar to properties of algorithm (a), are given in [2].

If $A$ is rank deficient, from property (P1), the linear dependent rows of $A$ can be deleted from system (1.1) when solving system (2.3).

The choice $Z_k = W_k$ verifies the nonsingularity condition for the matrix $A_k^T H_k^T Z_k$ and implies $d_k = r_k$ in (2.2).

If $m_k = 1, k = 1, ..., j$ ($j = m$), we obtain algorithm (a), while if $m = n$ and $m_k = m_1 = m$ ($j = 1$) and if $H_1 = I$, we obtain the explicit computation of $A^{-1}$ in (2.3) and if $Z_1 = W_1$, the solution (2.1) is computed by $x_2 = x_1 - A^{-1} r_1$. 

3. Alternative formulations of the block ABS algorithm

We reformulate the block ABS algorithm for a general matrix \( A \) in the case \( Z_k = W_k \), \( k = 1, \ldots, j \).

Algorithm (b1)

Give \( x_1, H_1, m_1, m_2, \ldots, m_j \) as above.

(*) For \( k = 1 \) to \( j \):

Compute

\[
S_k = H_k A_k ,
\]

\[
r_k = A_k^T x_k - b_k .
\]

If \( S_k = 0 \) and \( r_k = 0 \), set \( x_{k+1} = x_k, H_{k+1} = H_k \) and go to (*).

If \( S_k = 0 \) and \( r_k \neq 0 \), stop, system (1.1) is incompatible, or if a column \( s_i \) of \( S_k \) and the corresponding element \( \tau_i \) of \( r_k \) are such that \( s_i = 0, \tau_i \neq 0 \), stop, system (1.1) is incompatible.

If \( S_k \neq 0 \), apply algorithm (a) for finding the search vectors of the matrix \( S_k^T \in \mathbb{R}^{m_k \times n} \) deleting the linearly dependent rows if they exist (from property (P2) the corresponding rows in \( A \) are linearly dependent from the previous rows). Set \( S_k' \in \mathbb{R}^{n \times l_k} \) the matrix formed by the linearly independent columns of \( S_k \), \( l_k \leq m_k \).

Determine the solution of the system

\[
S_k'^T W_k = I_{l_k}
\]

by algorithm (a), where \( W_k \in \mathbb{R}^{n \times l_k} \).

Compute the matrix \( P_k \in \mathbb{R}^{n \times l_k} \) by

\[
P_k = H_k'^T W_k .
\]

Update the approximation to the solution by

\[
x_{k+1} = x_k - P_k r_k ,
\]

where \( r_k \) consists of the \( l_k \) components of \( r_k \) corresponding to the \( l_k \) rows of \( S_k'^T \).

If \( l_k < m_k \), compute \( \tau_i = a_i^T x_{k+1} - b_i^T e_i \) for the \( m_k - l_k \) indices \( i \) corresponding to the linearly dependent rows of \( S_k'^T \). If \( \tau_i \neq 0 \) for at least one index \( i \), stop, the system is incompatible.

If \( k = j \), stop, \( x_{j+1} \) is a solution of system (1.1), otherwise update matrix \( H_k \) by

\[
H_{k+1} = H_k - S_k' P_k^T .
\]
If \( m_k = 2, \ k = 1, \ldots, j, \ j = m/2 \), we obtain the following algorithm, also proposed by a different derivation in [3]. Suppose \( m \) even, if \( m \) is odd, the last step for \( k = m/2 + 1 \) corresponds to the last step of algorithm (a). Set \( A_k = (a_{k_1}, a_{k_2}), b_k = (b_{k_1}, b_{k_2}) \).

**Algorithm (b2)**

Let \( x_1, H_1 \) as above.

(\( \bullet \)) For \( k = 1 \) to \( m/2 \):
- Compute the vectors \( S_k = (s_{k_1}, s_{k_2}) \)
  \[
  s_{k_1} = H_k a_{k_1}, \quad s_{k_2} = H_k a_{k_2}.
  \] (3.7)
- Compute the residuals \( r_k = (\tau_{k_1}, \tau_{k_2}) \)
  \[
  \tau_{k_1} = a_{k_1}^T x_k - b_{k_1}, \quad \tau_{k_2} = a_{k_2}^T x_k - b_{k_2}.
  \] (3.8)

If \( s_{k_1} = 0, s_{k_2} = 0, \tau_{k_1} = 0, \tau_{k_2} = 0 \), set \( x_{k+1} = x_k ; H_{k+1} = H_k \) and go to (\( \bullet \)).

If \( s_{k_2} = 0 \) and \( \tau_{k_1} \neq 0 \), or \( s_{k_1} = 0 \) and \( \tau_{k_2} \neq 0 \), stop, the system is incompatible.

If \( s_{k_1} \neq 0 \) and \( s_{k_2} = 0, \tau_{k_2} = 0 \), set \( k_i = k_1 \) and go to (\( \bullet \bullet \)).

If \( s_{k_2} \neq 0 \) and \( s_{k_1} = 0, \tau_{k_1} = 0 \), set \( k_i = k_2 \) and go to (\( \bullet \bullet \)).

If \( s_{k_1} \neq 0 \) and \( s_{k_2} \neq 0 \), verify if \( s_{k_1}, s_{k_2} \) are linearly independent (for example applying algorithm (a) to the matrix \( (s_{k_1}, s_{k_2})^T \)).

If \( s_{k_1}, s_{k_2} \) are linearly independent, choose \( W_k = (w_{k_1}, w_{k_2}), W_k \in R^{n \times 2} \) satisfying system \( S_k^T W_k = I_2 \)
  \[
  s_{k_1}^T w_{k_1} = 1, \quad s_{k_1}^T w_{k_2} = 0, \quad s_{k_2}^T w_{k_1} = 0, \quad s_{k_2}^T w_{k_2} = 1.
  \] (3.9)

Update the approximation to the solution by
  \[
  x_{k+1} = x_k - H_k^T (\tau_{k_1} w_{k_1} + \tau_{k_2} w_{k_2}).
  \] (3.10)

If \( k = m/2 \), stop, \( x_{\frac{m}{2}+1} \) is a solution of system (1.1), otherwise update matrix \( H_k \) by
  \[
  H_{k+1} = H_k - s_{k_1} w_{k_1}^T H_k - s_{k_2} w_{k_2}^T H_k
  \] (3.11)
and go to (\( \bullet \)).

(\( \ast \)) If \( s_{k_2} \) is linearly dependent from \( s_{k_1} \), set \( k_i = k_2 \) (row \( k_2 \) in \( A \) is linearly dependent from the previous rows).

(\( \bullet \bullet \)) Choose \( w_{k_i} \in R^n \) such that
  \[
  s_{k_i}^T w_{k_i} = 1.
  \]
Update the approximation to the solution by
\[ x_{k+1} = x_k - \tau_k H_k^T w_{k_i} . \]

If it is case (*), compute \( \tau_{k_2} = a_{k_2}^T x_{k+1} - b_{k_2} \) and if \( \tau_{k_2} \neq 0 \), stop, the system is incompatible. If \( k = m/2 \), stop, \( x_{m/2 + 1} \) is a solution of system (1.1), otherwise update matrix \( H_k \) by
\[ H_{k+1} = H_k - s_k w_{k_i}^T H_k \]
and go to (●).

We consider two equivalent forms of the block ABS algorithm (b1), other particular cases of block forms can be found in [2].

The first versions of the algorithm (b1) constructs at every step \( k \), the same approximate solution \( x_k \) and the same matrices \( S_k, H_k \) avoiding the explicit computation of \( W_k \). Rewrite (3.5) as \( x_{k+1} = x_k - H_k^T W_k r_k \). Then by multiplying (3.3) by \( r_k \) we have \( S_k^T W_k r_k = r_k \). Set \( t_k = W_k r_k \), then instead of (3.3) we can solve \( S_k^T t_k = r_k \) and compute \( x_{k+1} = x_k - H_k^T t_k \). Rewrite (3.6) as \( H_{k+1} = (I - S_k W_k^T) H_k \). Denote by \( \tilde{H}_k \) the matrix obtained after \( m_k \) iterations of algorithm (a) applied to system \( S_k^T t_k = r_k \). From (3.3) it holds that \( \text{Null}(I - S_k W_k^T) = \text{Null}(\tilde{H}_k) \), hence from properties (P4),(P5),(P6),(P7) we can write \( \tilde{H}_k = I - S_k W_k^T \). Then the update of \( H_k \) in (3.6) is given by \( H_{k+1} = \tilde{H}_k H_k \). The first alternative formulation of algorithm (b1) is given by the following procedure.

Algorithm (c)

Let \( x_1, H_1, m_1, m_2, ..., m_j \) as above.

(●) For \( k = 1 \) to \( j \):

Compute
\[ S_k = H_k A_k , \quad (3.12) \]
\[ r_k = A_k^T x_k - b_k . \quad (3.13) \]

If \( S_k = 0 \) and \( r_k = 0 \) set \( x_{k+1} = x_k, : H_{k+1} = H_k \) and go to (●).

If \( S_k = 0 \) and \( r_k \neq 0 \), stop, the system is incompatible.

If \( S_k \neq 0 \), solve by algorithm (a) the system
\[ S_k^T t_k = r_k . \quad (3.14) \]

Update the approximation to the solution by
\[ x_{k+1} = x_k - H_k^T t_k . \quad (3.15) \]

If \( S_k \) is not full rank and at a step \( i \) system (3.14) is incompatible, then the corresponding equation \( i \) in system (1.1) is incompatible from the previous equations.
If \( k = j \), stop, \( x_{j+1} \) is a solution of system (1.1), otherwise update matrix \( H_k \) by
\[
H_{k+1} = \hat{H}_k H_k , \tag{3.16}
\]
where \( \hat{H}_k \) is the matrix (1.4) obtained after \( m_k \) steps of algorithm (a) applied to system (3.14).

The second version of algorithm (b1) is a modification of algorithm (c). It avoids the computation of the update \( H_{k+1} \) (a matrix-matrix product) but performs transformations on the matrix \( A \).

Define \( \bar{S}_k = H_k \bar{A}_k \) where \( \bar{A}_k \) is the transpose of the matrix formed by the last \( m_{k_1} + \ldots + m_j \) rows of \( A \) (i.e. the last \( j - k \) blocks of \( A \)), \( \bar{r}_k = \bar{A}_k^T x_k - \bar{b}_k \) where \( \bar{b}_k \) is the vector formed by the last \( m_{k_1} + \ldots + m_j \) components of \( b \), and observe that \( S_k = H_k A_k = \hat{H}_{k-1} H_{k-1} \bar{A}_k = \hat{H}_{k-1} H_{k-1} (\bar{A}_{k-1})_{m_k} = \hat{H}_{k-1} (\bar{S}_{k-1})_{m_k} \), \( r_k = A_k^T x_k - b_k = A_k^T x_k - b_k - A_k^T H_k^{-1} t_{k-1} = (A_{k-1}^T)_{m_k} x_{k-1} - (b_{k-1})_{m_k} - (A_{k-1})_{m_k} H_k^{-1} t_{k-1} = (\bar{r}_{k-1})_{m_k} - (S_{k-1}^T)_{m_k} t_k \), where \( (A_{k-1})_{m_k} \) is the submatrix formed by the first \( m_k \) columns of \( A_{k-1} \), \( (\bar{S}_{k-1})_{m_k} \) is the submatrix formed by the first \( m_k \) columns of \( \bar{S}_{k-1} \) and \( (\bar{r}_{k-1})_{m_k} \) is the vector formed by the first \( m_k \) components of \( \bar{r}_{k-1} \). Then algorithm (c) becomes

**Algorithm (d)**

Let \( x_1, H_1, m_1, m_2, \ldots, m_j \) as above.

Set \( \bar{S}_1 = H_1 A, \quad \bar{r}_1 = A x_1 - b \)

(•) For \( k = 1 \) to \( j \):

Set \( \bar{S}_k \) and \( \bar{r}_k \) partitioned as follows:
\[
\bar{S}_k = (S_k, \bar{S}_k) \quad \text{with dimension } (n \times m_k), \quad \bar{S}_k \text{ with dimension } (n \times (m_{k_1} + \ldots + m_j))
\]
\[
\bar{r}_k = (r_k, \bar{r}_k) \quad r_k \text{ with dimension } m_k, \quad \bar{r}_k \text{ with dimension } m_{k_1} + \ldots + m_j
\]

If \( S_k = 0 \) and \( r_k = 0 \), set \( \bar{S}_k = \bar{S}_k, \bar{r}_k = \bar{r}_k, t_k = 0, \cdot \cdot \cdot \bar{H}_k = I \) and go to (•).

If \( S_k = 0 \) and \( r_k \neq 0 \), stop, the system is incompatible.

If \( S_k \neq 0 \), solve by algorithm (a) the system
\[
S_k^T t_k = r_k . \tag{3.17}
\]

If \( S_k \) is not full rank and at a step \( i \) system (3.17) is incompatible, then the corresponding equation \( i \) in system (1.1) is incompatible from the previous equations.

If \( k < j \), compute
\[
\bar{S}_{k+1} = \hat{H}_k \bar{S}_k , \tag{3.18}
\]
\[
\bar{r}_{k+1} = \bar{r}_k - \bar{S}_k^T t_k , \tag{3.19}
\]
where \( \hat{H}_k \) is the matrix (1.4) obtained after \( m_k \) steps of algorithm (a) applied to system (3.17), and go to (•).
Estimate the solution by the following process:

Set $\bar{t}_j = t_j$, for $k = 1$ to $j - 1$ compute

$$\bar{t}_{j-k} = t_{j-k} + \tilde{H}_{j-k}^T \bar{t}_{j-k+1},$$

$$x_{j+1} = x_1 - H_1^T \bar{t}_1.$$

(3.20)

Algorithms (c) and (d) can be also derived from property (P8) if we consider at step $k$ a solution of the $k$-th block and all the possible solutions given by formula (1.5), at step $k + 1$ the solutions of the $(k + 1)$-th block satisfying relation (1.5).

The equivalent forms of algorithms (a), (b1), (b2), (c), (d) can differ in storage, overhead, computational timings, according to the type of ABS method, the type of representation of matrix $H_i$, and the type of implementation (sequential or parallel).

The block form of algorithm (b1) was tested in [4] on three algorithms of the ABS class: the implicit LU algorithm, the modified Huang algorithm, the implicit QR algorithm. The block form of algorithm (d) was tested in [5] on the implicit LU algorithm and on the modified Huang algorithm (using several representations of the matrix $H_k$) on parallel processor. In this paper we consider only the implicit LU method and a modification in [3] used in algorithm (b2).

4. The implicit LU algorithm and some of its block forms

The implicit LU algorithm corresponds to the parameter choices

$$H_1 = I, \quad z_i = e_i, \quad w_i = e_i/e_i^T H_i a_i.$$

It is well defined if $A$ has nonsingular principal submatrices, otherwise column pivoting has to be performed. Matrix $H_{i+1}$ has the structure

$$H_{i+1} = \begin{bmatrix} 0_{i \times i} & 0_{i \times (n-i)} \\ M_i & I_{n-i} \end{bmatrix}.$$

(4.1)

The first $i$ rows are zero, the submatrix given by the last $n - i$ rows and columns is the identity matrix. The search vector $p_{i+1}$ coincides with the $(i + 1)$-th row of $H_{i+1}$, hence the matrix $P_i$ in (P3) is unit upper triangular. The vector $s_{i+1}$ has the first $i$ components equal to zero. If $x_1 = 0$, only the first $i$ components of $x_{i+1}$ can be nonzero. Several tests on the implicit LU method can be found in [6].

The block generalization of the implicit LU method is obtained solving system (3.3), (3.9), (3.14), (3.17), of algorithms (b1), (b2), (c), (d), respectively, by the implicit LU method with zero starting point. Matrix $H_{k+1}$ in (3.6), (3.11), (3.16) is equal to (1.4) after $m_1 + m_2 + .. + m_k$ steps. Matrix $W_k$ can have nonzeros only in $m_k$ rows, which form the inverse of the $m_k \times m_k$ nonsingular submatrix of $S_k^T$
whose columns are chosen by the implicit LU method. Matrix $P_k$ in (3.4), (3.10) has $n - (m_{k+1} + \ldots + m_j)$ rows equal to zero and so $x_{k+1}$ in (3.5), (3.10), (3.15). Matrix $S_k$ in (3.1), (3.7), (3.12), (3.17) has $m_1 + \ldots + m_{k-1}$ rows equal to zero and the partial solution $t_k$ in (3.14), (3.17) can have only $m_k$ components that are nonzero.

The methods tested in our experiments are listed below. For the block algorithms we have used a size $n_b$ equal for all the blocks. The starting points are set to zero.

- **Implicit LU** - the method given above, where, for stability reasons, at every step $i$ we take column $j$ such that $\delta_j = \max(\delta_k = |c_k^T H_i a_i| : k = 1, \ldots, n)$. From the structure of $H_{i+1}$ (4.1), at every step $i$, the $n - i$ elements of $a_i$, not used to update the estimate solution (1.3), are overwritten by the elements of the last column of $M_i$, hence the storage required is the storage of $A$. Then the search vectors $p_i$ (column $i$ of $H_k^T$) are memorized in $A$ and system (1.1) can be solved with different right hand-sides using the vectors $p_i : i = 1, \ldots, n$ previously computed. The method requires $n m^2 - 2m^3/3$ multiplications for $m \leq n$ and $n^3/3$ for $m = n$.

- **Block implicit LU (b)** - the method given by algorithm (b1), where system (3.3) is solved by the implicit LU described above. For this method the $n_b \times n$ block $A_k^T$ is overwritten by the $kn_b \times n_b$ nonzero submatrix of $P_k$ and by the $n_b \times n - kn_b$ submatrix of $S_k^T$ used to update $H_k$ (3.6) and at step $k + i$ by a nonzero part of $H_{k+i+1}$, while for solving (3.3) we use a workspace of dimension $n_b \times n - (k - 1)n_b$. Thus the storage required is the storage of $A$ plus $n_b \times n$. To solve (3.3) $n^2 - (k - 1)n_b) n_b^2 - 2n_b^3/3$ multiplications are needed for computing the search vectors and $2n_b^2/3$ for $W_k$. $(k - 1)n_b^2$ multiplications are needed for the product $H_k^T W_k$ in computing $P_k$ (3.4), $(n - kn_b)kn_b^2$ for the product $S_k P_k^T$ in updating $H_{k+1}$ (3.6), $(k - 1)n_b^2$ for $r_k$ (3.2), $kn_b^2$ for $x_{k+1}$ (3.5) and $(n - (k - 1)n_b)(k - 1)n_b^2$ for $S_k$ (3.1). Hence the total cost for $m = n$ and $n$ multiple of $n_b$ is $n^3/3 + n^2n_b - nn_b^2/3 + O(n^2)$. The number of operations is an increasing function in $n_b$, minimum is $n^3/3$ for $n_b = 1$ (the algorithm corresponds to implicit LU) and maximum is $n^3$ for $n_b = n$ (the algorithm performs one step computing the inverse of $A$ in (3.3)).

- **Block implicit LU (c)** - the method given by algorithm (c), where system (3.14) is solved by the implicit LU described above. For this method the $n_b \times n$ block $A_k^T$ is overwritten by the $n_b \times n - (k - 1)n_b$ nonzero submatrix of $S_k^T$, then $S_k$ is overwritten by the nonzero submatrix of $H_k$ when solving system (3.14) and then $H_k$ is overwritten with a nonzero part of $H_{k+1}$ at next step $k + 1$ and by a nonzero part of $H_{k+i+1}$ at step $k + i$. Hence the memory cost is the storage of $A$. The number of multiplications required is $(n - (k - 1)n_b)n_b^2 - 2n_b^3/3$ for solving system (3.14), $(k - 1)n_b^2$ for $x_{k+1}$ in (3.15), $(k - 1)n_b^2$ for $r_k$ in (3.13), $(n - (k - 1)n_b)(k - 1)n_b^2$ for computing $S_k$ (3.12), $(k - 1)n_b^2(n - kn_b)$ for the product in $H_{k+1}$ in (3.16). The total amount for $m = n$ and $n$ multiple of $n_b$ is $n^3/3 + O(n^2)$. Also in the case of blocks $m_1, m_2, \ldots, m_j$ with different size the total amount is $n^3/3 + O(n^2)$.

- **Block implicit LU (d)** - the method given by algorithm (d), where system (3.17)
is solved by the implicit LU described above. For this method the \((n- kn_b) \times (n- (k - 1) n_b)\) nonzero submatrix of \(S_k\) is overwritten by the \((n- kn_b) \times (n- kn_b)\) nonzero submatrix of \(S_{k+1}\) and the \(n_b \times (n- (k - 1) n_b)\) nonzero submatrix of \(S_k^T\) is overwritten by the \(n_b \times (n- kn_b)\) nonzero submatrix of \(H_k^T\) when solving system (3.17). Hence the memory is the storage of \(A\). The number of multiplications required is \((n- (k - 1) n_b) n_b^2 - 2n_b^3/3\) for solving system (3.17), \((n- kn_b) n_b\) for computing \(r_{k+1}\) (3.19), \(n_b(n- kn_b)^2\) for the product in \(S_{k+1}\) (3.18) and \(n^2/2\) for the process in estimating the solution \(x_{k+1}\) (3.20). The total amount for \(m = n\) and \(n\) multiple of \(n_b\) is \(n^3/3 + O(n^2)\) also for blocks \(m_1, m_2, ..., m_j\) with different size.

Notice that the two versions of block implicit LU (c) and (d) have the same storage and overhead as implicit LU has.

5. The algorithm of Adib et al.

Here we call Algorithm [3] the method described by Adib et al. in [3]. We show that this method is algorithm (b2), where system (3.9) is solved applying twice the implicit LU.

Using implicit LU in (3.9) the search vectors \(p_{k_1}, p_{k_2}\) of the matrix \((s_{k_1}, s_{k_2})^T\) are constructed choosing two columns \(j_1, j_2\) such that

\[
\delta_{j_1} = \max(\delta_i = |s_{k_1(i)}| \quad i = 1, ..., n),
\]

\[
\delta_{j_2} = \max(\delta_i = |s_{k_2(i)} - (s_{k_2(j_1)} s_{k_1(i)})/\delta_{j_1}| \quad i = 1, ..., n),
\]

hence

\[
p_{k_1(i)} = 1, \quad p_{k_1(i)} = 0, \quad i = 1, ..., n \quad i \neq j_1
\]

\[
p_{k_2(j_1)} = -s_{k_2(j_2)}/\delta_{j_1}, \quad p_{k_2(j_2)} = 1, \quad p_{k_2(i)} = 0 \quad i = 1, ..., n \quad i \neq j_1,j_2.
\]

The solutions \(w_{k_1}, w_{k_2}\) can be nonzeros only in the components \(j_1, j_2\)

\[
w_{k_1(j_1)} = 1/\delta_{j_1} + (s_{k_1(j_1)}/(\delta_{j_1}\delta_{j_2}))(s_{k_1(j_2)}/\delta_{j_1}),
\]

\[
w_{k_1(j_2)} = -s_{k_2(j_2)}/\delta_{j_2}, \quad w_{k_1(i)} = 0, \quad i = 1, ..., n \quad i \neq j_1,j_2
\]

\[
w_{k_2(j_1)} = -s_{k_2(j_2)}/(\delta_{j_1}\delta_{j_2}), \quad w_{k_2(j_2)} = 1/\delta_{j_2}, \quad w_{k_2(i)} = 0, \quad i = 1, ..., n \quad i \neq j_1,j_2
\]

and from property (4.1) of the update, rows \(j_1, j_2\) in \(H_{k+1}\) (3.11) become zero.

In the method proposed in [3], the parameters \(w_{k_1}, w_{k_2}\) correspond to the two solutions obtained solving by implicit LU the two following systems

\[
\left\{
\begin{array}{l}
{S_k}^T w_{k_1} = 0 \\
{s_k}^T w_{k_2} = 1
\end{array}
\right..
\]

\[
\left\{
\begin{array}{l}
{s_k}^T w_{k_1} = 0 \\
{s_k}^T w_{k_2} = 1
\end{array}
\right..
\]
The solution \( w_{k_2} \) of the second system (5.5) is equal to the solution (5.3) since the sequence of the rows \( s_{k_2}^T \), \( s_{k_2}^T \) is the same as in (3.9) and so the choice of the pivot indices \( j_1, j_2 \) is (5.1). The solution \( w_{k_1} \) of the first system (5.4) is not (5.2) since in (5.4) the first considered row is \( s_{k_2}^T \), then the columns taken for the pivot can be different from \( j_1, j_2 \). Set

\[
\delta_{j_3} = \max(\delta_i = |s_{k_2(i)}|, \quad i = 1, \ldots, n),
\]

\[
\delta_{j_4} = \max(\delta_i = |s_{k_1(i)} - (s_{k_1(j_3)}s_{k_2(i)})/\delta_{j_3}|, \quad i = 1, \ldots, n).
\]

The search vectors of (5.4) are

\[
p_{k_1(j_3)} = 1, \quad p_{k_1(i)} = 0, \quad i = 1, \ldots, n \quad i \neq j_3
\]

\[
p_{k_2(j_3)} = -s_{k_2(j_4)}/\delta_{j_3}, \quad p_{k_2(j_4)} = 1, \quad p_{k_2(i)} = 0, \quad i = 1, \ldots, n \quad i \neq j_3, j_4
\]

and the solutions \( w_{k_1} \) can be nonzeros only in the components \( j_3, j_4 \)

\[
w_{k_1(j_3)} = -s_{k_2(j_4)}/(\delta_{j_3}\delta_{j_4}), \quad w_{k_1(j_4)} = 1/\delta_{j_4}, \quad w_{k_1(i)} = 0, \quad i = 1, \ldots, n \quad i \neq j_3, j_4.
\]

(5.6)

Since \( H_1 = I \), rows \( j_1, j_2, j_3, j_4 \) in (3.11) can be different from the unit vector, and the process leads to a dense matrix \( H_k \). If \( j_1, j_2 \) coincide with \( j_3, j_4 \), solution (5.6) is equal to (5.2), the step corresponds to a step of the block implicit LU algorithm and rows \( j_1, j_2 \) of \( H_{k+1} \) become zero. The indices \( j_1, j_2 \) can be equal to \( j_3, j_4 \) if matrix \( (s_{k_1}, s_{k_2})^T \) has two dominant elements in two different columns, thus if \( A \) is diagonally dominant, algorithm [3] becomes the block implicit LU (b) algorithm with blocksize \( n_b = 2 \).

The special structure of \( H_k \) depends on the pivot indices \( j_1, j_2, j_3, j_4 \). A zero row appears if \( (s_{k_1}, s_{k_2}) \) has rank 1 (iteration \( k \) is an implicit LU iteration). Also zero rows appear at a step \( k \) when the number of a set of rows taken from the first \( 2k \) is equal to the number of different indices of columns selected as pivot in those rows (in this case all rows with indices in the set become zero in \( H_{k+1} \)).

Algorithm [3] can be generalized for blocks greater than 2, but it becomes very expensive.

The code written for numerical experiments uses an \( n \times n \) array to memorize \( H_k \) since generally it is not known how sparse the matrix can be. The code performs at every iteration a search to find the set of pivot indices, used at previous iterations, which can give zero rows in \( H_k \). If this set exists, these rows are ignored in the multiplications and so the number of operations is reduced. Then the amount of operations may vary according to the values of the elements of \( A \), the minimum being \( n^3/3 \) when at every step the indices \( j_1, j_2 \) coincide with \( j_3, j_4 \), maximum is \( 3n^3/2 \) when at every step the indices \( j_1, j_2 \) are different from \( j_3, j_4 \) and there are no zero rows in \( H_k \).

The following table reports storage requirement and number of multiplications for the above algorithms (only higher order terms are considered).
E. Bodon

storage  overhead

| implicit LU       | $n^2$   | $n^3/3$ |
|-------------------|---------|---------|
| algorithm [3]     | $2n^2$  | min $n^3/3$, max $3n^3/2$ |
| block implicit LU (b) $n_b = 2$ | $n^2$   | $n^3/3$ |
| block implicit LU (c) $n_b = 2$ | $n^2$   | $n^3/3$ |
| block implicit LU (d) $n_b = 2$ | $n^2$   | $n^3/3$ |
| block implicit LU (b) | $n^2 + n_b n$ | $n^3/3 + n^2 n_b - n n_b^2/3$ |
| block implicit LU (c) | $n^2$   | $n^3/3$ |
| block implicit LU (d) | $n^2$   | $n^3/3$ |

6. Numerical testing and results

Codes for testing the above algorithms were written in Fortran 77, run in single precision and tested on square matrices with dimension $n = 100, n = 200$, using the same size $n_b$ for all blocks, with values $n_b = 2, 5, 10, 20, 50$. The following families of problems were considered.

(M1) Matrices with random integer elements, solution vector $x^*$ with random integer elements and right hand-side $b$ computed exactly by $b = Ax^*$.

(M2) Ill conditioned matrices with random integer elements having two or more rows and/or columns equal save for an element $a_{ij}$ set equal to $a_{ij} + 2^{-k}$, $k$ integer, solution vector $x^*$ with random integer elements and the component $i$ (or $j$) equal to $2^k$ so that the right hand-side $b = Ax^*$ has integer elements.

(M3) Ill conditioned matrices formed by a product of a random matrix of family (M1) and an ill conditioned and small dimensioned matrix with integer elements taken from [7] (matrices of Wilson, Rutishauser, Morris, Pascal, Tanabe etc.). The solution vector $x^*$ has random integer elements and $b$ is computed by $b = Ax^*$.

(M4) A set of matrices listed in [3] and [8] having not all integer entries but exactly represented in the machine, the solution vector $x^*$ having all components equal to 1.

The performance of the algorithms was measured by the relative error in the solution and in the residual with Euclidean norm defined respectively by $\|x - x^*\|/\|x^*\|$ , $\|Ax - b\|/\|b\|$. Some of the results are reported in Tables 1, 2, 3, 4. Tables 5, 6 give a comparison of the algorithms on 200 problems. The elements $u_{ij}/v_{ij}$ have the following meaning: $u_{ij}$ indicates how many times the algorithm in row $i$ has a smaller relative error than the algorithm in column $j$, $v_{ij}$ how many times the relative error is equal (only the first two digits are considered). We find that all the algorithms have similar performance in most problems. The block implicit LU (b) seems to have the worst performance for great values of blocksize $n_b$, probably due to the computation of the inverse of a not small dimensioned matrix in (3.3). The original nonblock version gives generally the best performance.

Acknowledgement: This work is partly supported by MURST 1999 Programma di Cofinanziamento.
REFERENCES

[1] Abaffy, J. and Galántai A.: Conjugate direction methods for linear and nonlinear systems of algebraic equations, Colloq. Math. Soc. János Bolyai 50, (1986), 481-502.

[2] Abaffy J. and Spedicato E.: ABS Projection Algorithms: Mathematical Techniques For Linear And Nonlinear Algebraic Equations, Ellis Horwood, Chichester, 1989.

[3] Adibi M. and Mahdavi-Amiri N.: A new ABS-based approach for solving linear systems with a rank two update of the Abaffian. Unpublished manuscript

[4] Bertocchi M. and Spedicato E.: ABS Block Algorithms for Dense Linear Systems in a Vector Processor Environment, Technical Note, University of Bergamo, 1988.

[5] Bodon E.: Numerical Results of the ABS Algorithms for Linear Systems of Equations, Report DMSIA N.9, University of Bergamo, 1993.

[6] Nicolai S. and Spedicato E.: A bibliography of the ABS methods, Optimization Methods and Software, 8, (1997), 171-183.

[7] Spedicato E. and Vespucci M.T.: Variations on the Gram-Schmidt and the Huang algorithms for linear systems: a numerical study, Applications of Mathematics, 2, (1993), 81-100.

[8] Mirnia K.: Iterative Refinement In ABS Methods, Report DMSIA 32/96, University of Bergamo, 1996.
| Method            | n=100                  | Dimension n=200                |
|-------------------|------------------------|--------------------------------|
|                  | Solution error | Residual error | Solution error | Residual error |
| impl.LU           | .24E-05              | .54E-06          | .39E-05        | .83E-06        |
| block impl.LU (b)| n=2                   | .59E-05          | .14E-04        | .12E-05        |
| block impl.LU (c)| n=2                   | .30E-05          | .12E-04        | .11E-05        |
| block impl.LU (d)| n=2                   | .34E-05          | .82E-05        | .25E-05        |
| block impl.LU (b)| n=5                   | .33E-05          | .14E-04        | .11E-05        |
| block impl.LU (c)| n=5                   | .54E-05          | .13E-04        | .13E-05        |
| block impl.LU (d)| n=5                   | .80E-05          | .14E-04        | .22E-05        |
| block impl.LU (c)| n=10                  | .39E-05          | .10E-04        | .14E-05        |
| block impl.LU (d)| n=10                  | .53E-05          | .25E-05        | .12E-05        |
| block impl.LU (b)| n=10                  | .84E-05          | .93E-05        | .19E-05        |
| block impl.LU (c)| n=50                  | .96E-05          | .20E-05        | .14E-05        |
| block impl.LU (d)| n=50                  | .96E-05          | .28E-04        | .15E-05        |
| impl.LU           | .35E-05              | .56E-06          | .84E-05        | .93E-06        |
| algorithm[3]      | .46E-05              | .76E-06          | .13E-05        |                 |
| block impl.LU (b)| n=2                   | .47E-05          | .99E-05        | .21E-05        |
| block impl.LU (c)| n=2                   | .53E-05          | .85E-05        | .21E-05        |
| block impl.LU (d)| n=2                   | .13E-04          | .81E-05        | .12E-05        |
| block impl.LU (b)| n=5                   | .11E-04          | .68E-05        | .11E-05        |
| block impl.LU (d)| n=5                   | .10E-05          | .14E-05        | .30E-05        |
| block impl.LU (b)| n=10                  | .24E-05          | .73E-05        | .14E-05        |
| block impl.LU (c)| n=10                  | .53E-05          | .99E-05        | .13E-05        |
| block impl.LU (d)| n=10                  | .78E-05          | .84E-05        | .19E-05        |
| block impl.LU (b)| n=20                  | .98E-05          | .85E-05        | .13E-05        |
| block impl.LU (c)| n=20                  | .15E-05          | .10E-05        | .13E-05        |
| block impl.LU (d)| n=20                  | .25E-05          | .99E-05        | .13E-05        |
| block impl.LU (b)| n=50                  | .60E-05          | .42E-05        | .16E-05        |
| block impl.LU (c)| n=50                  | .75E-05          | .22E-05        | .58E-05        |
| block impl.LU (d)| n=50                  | .80E-05          | .46E-05        | .15E-05        |
| impl.LU           | .59E-06              | .56E-06          | .46E-05        | .96E-06        |
| algorithm[3]      | .46E-05              | .71E-06          | .13E-03        | .11E-05        |
| block impl.LU (b)| n=2                   | .18E-04          | .60E-04        | .11E-05        |
| block impl.LU (c)| n=2                   | .12E-04          | .12E-03        | .12E-05        |
| block impl.LU (d)| n=2                   | .84E-05          | .86E-04        | .20E-05        |
| block impl.LU (b)| n=5                   | .45E-05          | .13E-03        | .11E-05        |
| block impl.LU (c)| n=5                   | .64E-06          | .11E-03        | .12E-05        |
| block impl.LU (d)| n=5                   | .93E-06          | .50E-04        | .26E-05        |
| block impl.LU (b)| n=10                  | .54E-05          | .15E-03        | .16E-05        |
| block impl.LU (c)| n=10                  | .55E-05          | .10E-03        | .12E-05        |
| block impl.LU (d)| n=10                  | .67E-06          | .13E-03        | .17E-05        |
| block impl.LU (b)| n=20                  | .55E-05          | .23E-04        | .28E-05        |
| block impl.LU (c)| n=20                  | .62E-06          | .10E-03        | .13E-05        |
| block impl.LU (d)| n=20                  | .75E-06          | .61E-04        | .15E-05        |
| block impl.LU (b)| n=50                  | .52E-05          | .19E-03        | .37E-05        |
| block impl.LU (c)| n=50                  | .64E-06          | .77E-04        | .12E-05        |
| block impl.LU (d)| n=50                  | .64E-06          | .33E-04        | .14E-05        |
| impl.LU           | .30E-05              | .41E-06          | .21E-04        | .84E-06        |
| algorithm[3]      | .13E-04              | .69E-06          | .92E-05        | .94E-06        |
| block impl.LU (b)| n=2                   | .13E-04          | .16E-04        | .96E-06        |
| block impl.LU (c)| n=2                   | .18E-04          | .18E-04        | .95E-06        |
| block impl.LU (d)| n=2                   | .51E-05          | .10E-04        | .17E-05        |
| block impl.LU (b)| n=5                   | .71E-05          | .11E-04        | .10E-05        |
| block impl.LU (c)| n=5                   | .57E-06          | .14E-04        | .96E-06        |
| block impl.LU (d)| n=5                   | .82E-06          | .26E-05        | .16E-05        |
| block impl.LU (b)| n=10                  | .97E-05          | .98E-05        | .11E-05        |
| block impl.LU (c)| n=10                  | .50E-06          | .71E-05        | .99E-06        |
| block impl.LU (d)| n=10                  | .79E-06          | .13E-04        | .19E-05        |
| block impl.LU (b)| n=20                  | .46E-05          | .87E-05        | .24E-05        |
| block impl.LU (c)| n=20                  | .67E-05          | .70E-05        | .12E-05        |
| block impl.LU (d)| n=20                  | .50E-06          | .21E-04        | .17E-05        |
| block impl.LU (b)| n=50                  | .20E-04          | .53E-04        | .97E-05        |
| block impl.LU (c)| n=50                  | .62E-06          | .25E-04        | .13E-05        |
| block impl.LU (d)| n=50                  | .62E-06          | .31E-04        | .13E-05        |
| Method          | dimension n=100 |          |          |          | Dimension n=200 |          |          |
|-----------------|-----------------|----------|----------|----------|-----------------|----------|----------|
|                 | Solution error  | Residual error |          |          |                 | Solution error | Residual error |
| block impl.LU (b) n₀ = 2 | .18E-02 | .61E-06 |          |          | .24E-03 | .11E-05 |          |
| block impl.LU (c) n₀ = 2 | .61E-02 | .53E-06 | .10E-03 | .11E-05 |          |          |          |
| block impl.LU (d) n₀ = 2 | .32E-02 | .72E-06 | .16E-03 | .21E-05 |          |          |          |
| block impl.LU (c) n₀ = 5 | .21E-01 | .66E-06 | .94E-04 | .12E-05 |          |          |          |
| block impl.LU (d) n₀ = 5 | .81E-02 | .55E-06 | .15E-03 | .10E-05 |          |          |          |
| block impl.LU (b) n₀ = 20 | .27E-04 | .71E-06 | .12E-03 | .17E-05 |          |          |          |
| block impl.LU (c) n₀ = 10 | .42E-01 | .55E-06 | .16E-03 | .12E-05 |          |          |          |
| block impl.LU (d) n₀ = 10 | .19E-01 | .12E-05 | .16E-03 | .14E-05 |          |          |          |
| block impl.LU (b) n₀ = 20 | .87E-02 | .41E-05 | .20E-03 | .12E-04 |          |          |          |
| block impl.LU (c) n₀ = 20 | .25E-01 | .71E-06 | .51E-04 | .12E-05 |          |          |          |
| block impl.LU (d) n₀ = 20 | .31E-02 | .65E-06 | .52E-04 | .14E-05 |          |          |          |
| block impl.LU (b) n₀ = 50 | .11E-01 | .27E-04 | .21E-03 | .24E-04 |          |          |          |
| block impl.LU (c) n₀ = 50 | .28E-01 | .74E-06 | .47E-04 | .13E-05 |          |          |          |
| block impl.LU (d) n₀ = 50 | .38E-01 | .74E-06 | .55E-04 | .15E-05 |          |          |          |
| block impl.LU (b) n₀ = 2 | .18E-03 | .54E-06 | .79E-03 | .86E-06 |          |          |          |
| block impl.LU (c) n₀ = 2 | .17E-04 | .61E-06 | .60E-03 | .13E-05 |          |          |          |
| block impl.LU (d) n₀ = 2 | .21E-04 | .63E-06 | .10E-02 | .10E-05 |          |          |          |
| block impl.LU (b) n₀ = 5 | .35E-03 | .63E-06 | .24E-03 | .99E-06 |          |          |          |
| block impl.LU (c) n₀ = 5 | .81E-03 | .79E-06 | .84E-04 | .10E-05 |          |          |          |
| block impl.LU (d) n₀ = 5 | .23E-03 | .84E-06 | .52E-03 | .11E-05 |          |          |          |
| block impl.LU (b) n₀ = 10 | .74E-03 | .61E-06 | .65E-03 | .11E-05 |          |          |          |
| block impl.LU (c) n₀ = 10 | .88E-05 | .65E-06 | .22E-03 | .19E-05 |          |          |          |
| block impl.LU (d) n₀ = 10 | .48E-03 | .68E-05 | .61E-03 | .11E-05 |          |          |          |
| block impl.LU (b) n₀ = 10 | .43E-03 | .56E-06 | .31E-03 | .11E-05 |          |          |          |
| block impl.LU (c) n₀ = 20 | .80E-03 | .81E-06 | .50E-03 | .19E-05 |          |          |          |
| block impl.LU (d) n₀ = 20 | .99E-04 | .64E-06 | .61E-03 | .11E-05 |          |          |          |
| block impl.LU (b) n₀ = 20 | .48E-03 | .72E-06 | .40E-03 | .17E-05 |          |          |          |
| block impl.LU (c) n₀ = 50 | .30E-03 | .16E-03 | .22E-03 | .49E-05 |          |          |          |
| block impl.LU (d) n₀ = 50 | .86E-04 | .54E-06 | .44E-04 | .14E-05 |          |          |          |
| block impl.LU (b) n₀ = 50 | .86E-04 | .54E-06 | .75E-04 | .17E-05 |          |          |          |
| block impl.LU (c) n₀ = 10 | .18E-03 | .58E-06 | .14E-03 | .17E-05 |          |          |          |
| block impl.LU (d) n₀ = 10 | .17E-03 | .58E-06 | .13E-03 | .15E-05 |          |          |          |
| block impl.LU (b) n₀ = 2 | .13E-03 | .58E-06 | .14E-03 | .15E-05 |          |          |          |
| block impl.LU (c) n₀ = 2 | .12E-02 | .94E-06 | .46E-03 | .91E-06 |          |          |          |
| block impl.LU (d) n₀ = 2 | .41E-03 | .80E-06 | .87E-04 | .11E-05 |          |          |          |
| block impl.LU (b) n₀ = 5 | .56E-04 | .73E-06 | .56E-03 | .11E-05 |          |          |          |
| block impl.LU (c) n₀ = 5 | .32E-04 | .14E-05 | .13E-04 | .28E-05 |          |          |          |
| block impl.LU (d) n₀ = 5 | .12E-02 | .71E-05 | .29E-03 | .98E-06 |          |          |          |
| block impl.LU (b) n₀ = 10 | .31E-03 | .60E-04 | .14E-03 | .11E-05 |          |          |          |
| block impl.LU (c) n₀ = 10 | .58E-04 | .10E-05 | .12E-03 | .22E-05 |          |          |          |
| block impl.LU (d) n₀ = 10 | .36E-03 | .69E-05 | .25E-03 | .12E-05 |          |          |          |
| block impl.LU (b) n₀ = 20 | .20E-02 | .87E-06 | .17E-03 | .11E-05 |          |          |          |
| block impl.LU (c) n₀ = 20 | .94E-03 | .78E-06 | .42E-03 | .19E-05 |          |          |          |
| block impl.LU (d) n₀ = 20 | .14E-02 | .16E-04 | .28E-03 | .16E-05 |          |          |          |
| block impl.LU (b) n₀ = 20 | .76E-05 | .67E-06 | .48E-04 | .12E-05 |          |          |          |
| block impl.LU (c) n₀ = 20 | .89E-03 | .81E-06 | .22E-03 | .17E-05 |          |          |          |
| block impl.LU (d) n₀ = 20 | .98E-04 | .39E-04 | .18E-03 | .76E-05 |          |          |          |
| block impl.LU (b) n₀ = 50 | .16E-02 | .75E-06 | .14E-03 | .14E-05 |          |          |          |
| block impl.LU (c) n₀ = 50 | .16E-02 | .75E-06 | .13E-03 | .15E-05 |          |          |          |
| block impl.LU (d) n₀ = 50 | .14E-03 | .68E-06 | .39E-03 | .14E-05 |          |          |          |
| block impl.LU (b) n₀ = 50 | .14E-03 | .68E-06 | .40E-03 | .16E-05 |          |          |          |
| Method          | dimension n=100 | Dimension n=200 |
|-----------------|----------------|-----------------|
|                 | Solution error | Residual error  | Solution error | Residual error |
| impl.LU         | .10E-02        | .25E-06         | .10E-02        | .36E-06         |
| algorithm[3]    |                |                 |                |                 |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 2     | .33E-03        | .20E-06         | .33E-03        | .32E-06         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 2     | .72E-03        | .30E-06         | .38E-03        | .27E-06         |
| block impl.LU   |                |                 |                |                 |
| (d) n₀ = 2     | .30E-03        | .54E-06         | .34E-03        | .31E-06         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 5     | .95E-03        | .37E-06         | .37E-03        | .39E-06         |
| block impl.LU   |                |                 |                |                 |
| (d) n₀ = 5     | .12E-02        | .38E-06         | .38E-03        | .26E-06         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 10    | .41E-04        | .11E-06         | .24E-03        | .22E-06         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 10    | .24E-02        | .49E-06         | .96E-03        | .14E-05         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 10    | .14E-02        | .51E-06         | .30E-03        | .41E-06         |
| block impl.LU   |                |                 |                |                 |
| (d) n₀ = 10    | .15E-03        | .21E-06         | .35E-03        | .18E-06         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 20    | .66E-02        | .19E-05         | .39E-03        | .79E-06         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 20    | .54E-04        | .34E-06         | .48E-03        | .24E-06         |
| block impl.LU   |                |                 |                |                 |
| (d) n₀ = 20    | .20E-03        | .48E-06         | .24E-03        | .17E-06         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 50    | .21E-01        | .33E-05         | .23E-02        | .42E-05         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 50    | .11E-02        | .30E-06         | .78E-03        | .74E-06         |
| block impl.LU   |                |                 |                |                 |
| (d) n₀ = 50    | .11E-02        | .30E-06         | .78E-03        | .74E-06         |
| impl.LU         | .10E-02        | .25E-06         | .53E-02        | .39E-06         |
| algorithm[3]    |                |                 |                |                 |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 2     | .92E-03        | .17E-06         | .28E-02        | .30E-06         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 2     | .46E-02        | .24E-06         | .22E-02        | .80E-06         |
| block impl.LU   |                |                 |                |                 |
| (d) n₀ = 2     | .53E-04        | .38E-06         | .86E-02        | .83E-06         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 5     | .46E-04        | .19E-06         | .15E-02        | .18E-06         |
| block impl.LU   |                |                 |                |                 |
| (d) n₀ = 5     | .47E-02        | .49E-06         | .12E-01        | .18E-05         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 10    | .26E-03        | .15E-06         | .14E-02        | .47E-06         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 10    | .37E-02        | .69E-06         | .98E-02        | .25E-05         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 20    | .24E-02        | .29E-06         | .26E-02        | .32E-06         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 0      | .12E-02        | .16E-06         | .20E-02        | .32E-06         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 10    | .18E-02        | .27E-05         | .43E-01        | .37E-05         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 20    | .39E-02        | .23E-06         | .19E-02        | .36E-06         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 20    | .44E-03        | .13E-06         | .21E-02        | .69E-06         |
| block impl.LU   |                |                 |                |                 |
| (c) n₀ = 50    | .47E+00        | .29E-04         | .22E+00        | .29E-06         |
| block impl.LU   |                |                 |                |                 |
| (b) n₀ = 50    | .95E-02        | .32E-06         | .23E-02        | .33E-06         |
| block impl.LU   |                |                 |                |                 |
| (d) n₀ = 50    | .95E-02        | .32E-06         | .23E-02        | .33E-06         |

*Table 3 - Test matrices (M3)*
### Table 4 - Test matrices (M4)

| Method          | Dimension n=100 |          | Dimension n=200 |          |
|-----------------|-----------------|----------|-----------------|----------|
|                 | Solution error  | Residual error | Solution error  | Residual error |
| impl.LU         | .69E-05         | .11E-06  | .19E-04         | .12E-06  |
| block impl.LU   | .13E-04         | .24E-06  | .55E-04         | .47E-06  |
| block impl.LU   | .13E-04         | .16E-06  | .52E-04         | .27E-06  |
| block impl.LU   | .35E-05         | .44E-06  | .46E-05         | .35E-06  |
| block impl.LU   | .12E-04         | .10E-06  | .44E-04         | .33E-06  |
| block impl.LU   | .82E-05         | .16E-06  | .67E-04         | .46E-06  |
| block impl.LU   | .29E-05         | .35E-06  | .50E-05         | .87E-06  |
| block impl.LU   | .20E-04         | .25E-06  | .61E-04         | .65E-06  |
| block impl.LU   | .15E-04         | .14E-06  | .10E-03         | .57E-06  |
| block impl.LU   | .46E-05         | .49E-07  | .78E-05         | .13E-06  |
| block impl.LU   | .20E-04         | .43E-06  | .16E-03         | .95E-06  |
| block impl.LU   | .13E-04         | .21E-06  | .46E-04         | .85E-06  |
| block impl.LU   | .15E-04         | .21E-06  | .18E-04         | .11E-06  |
| block impl.LU   | .28E-04         | .33E-06  | .24E-04         | .90E-07  |
| block impl.LU   | .17E-04         | .29E-06  | .62E-04         | .31E-06  |
| block impl.LU   | .42E-05         | .49E-06  | .57E-04         | .25E-06  |
| block impl.LU   | .13E-04         | .11E-06  | .67E-04         | .26E-06  |
| block impl.LU   | .22E-04         | .28E-06  | .12E-03         | .73E-06  |
| block impl.LU   | .42E-05         | .52E-06  | .59E-05         | .27E-06  |
| block impl.LU   | .12E-04         | .12E-06  | .98E-05         | .40E-06  |
| block impl.LU   | .28E-04         | .27E-06  | .92E-04         | .44E-06  |
| block impl.LU   | .50E-05         | .35E-06  | .88E-05         | .12E-05  |
| block impl.LU   | .16E-04         | .49E-06  | .73E-04         | .72E-06  |
| block impl.LU   | .50E-05         | .29E-06  | .10E-04         | .61E-06  |
| block impl.LU   | .23E-03         | .28E-05  | .97E-04         | .41E-06  |
| block impl.LU   | .13E-03         | .17E-06  | .12E-03         | .15E-06  |
| block impl.LU   | .32E-04         | .30E-06  | .18E-04         | .17E-06  |
| impl.LU         | .68E-03         | .29E-06  | .37E-02         | .35E-06  |
| block impl.LU   | .10E-02         | .17E-06  | .14E-02         | .11E-06  |
| block impl.LU   | .83E-04         | .21E-06  | .78E-02         | .26E-06  |
| block impl.LU   | .28E-03         | .13E-06  | .16E-02         | .20E-06  |
| block impl.LU   | .59E-03         | .15E-06  | .58E-02         | .22E-06  |
| block impl.LU   | .10E-02         | .15E-06  | .63E-02         | .20E-06  |
| block impl.LU   | .33E-03         | .19E-06  | .22E-02         | .26E-06  |
| block impl.LU   | .84E-03         | .19E-06  | .68E-02         | .24E-06  |
| block impl.LU   | .87E-03         | .20E-06  | .54E-02         | .23E-06  |
| block impl.LU   | .32E-03         | .22E-06  | .19E-02         | .30E-06  |
| block impl.LU   | .13E-02         | .50E-06  | .64E-02         | .23E-06  |
| block impl.LU   | .54E-03         | .23E-06  | .42E-02         | .28E-06  |
| block impl.LU   | .32E-03         | .22E-06  | .15E-02         | .22E-06  |
| block impl.LU   | .12E-01         | .25E-05  | .12E-01         | .11E-05  |
| block impl.LU   | .65E-04         | .29E-06  | .41E-02         | .48E-06  |
| block impl.LU   | .65E-03         | .29E-06  | .14E-02         | .53E-06  |
| impl.LU         | .49E-03         | .15E-06  | .20E-02         | .22E-06  |
| block impl.LU   | .10E-02         | .18E-06  | .37E-02         | .26E-06  |
| block impl.LU   | .88E-03         | .16E-06  | .42E-02         | .24E-06  |
| block impl.LU   | .15E-03         | .19E-06  | .48E-03         | .22E-06  |
| block impl.LU   | .12E-02         | .17E-06  | .37E-02         | .25E-06  |
| block impl.LU   | .13E-02         | .18E-06  | .34E-02         | .25E-06  |
| block impl.LU   | .20E-03         | .15E-06  | .88E-03         | .23E-06  |
| block impl.LU   | .14E-02         | .19E-06  | .57E-02         | .26E-06  |
| block impl.LU   | .11E-02         | .17E-06  | .34E-02         | .26E-06  |
| block impl.LU   | .49E-03         | .15E-06  | .11E-02         | .23E-06  |
| block impl.LU   | .28E-01         | .40E-05  | .91E-01         | .37E-05  |
| block impl.LU   | .87E-03         | .21E-06  | .35E-02         | .26E-06  |
| block impl.LU   | .73E-03         | .18E-06  | .15E-02         | .21E-06  |
| block impl.LU   | .32E+00         | .45E-04  | .55E+00         | .29E-04  |
| block impl.LU   | .82E-03         | .15E-06  | .36E-02         | .24E-06  |
| block impl.LU   | .82E-03         | .15E-06  | .25E-02         | .23E-06  |
Table 5 - Comparison of solution error

|                | imp. LU | algo. [3] | Block implicit |
|----------------|---------|-----------|----------------|
|                | \( n_a \) | \( \text{LU(b)} \) | \( \text{LU(c)} \) | \( \text{LU(d)} \) | \( \text{LU(b)} \) | \( \text{LU(c)} \) | \( \text{LU(d)} \) |
| \( \text{imp. LU} \) |          |           |                |
| \( \text{algorithm [3]} \) | 85/9    | 106/9     | 107/9          | 113/10        | 72/10      | 122/9      | 115/11        | 78/10          |
| \( \text{imp. LU(b)} \) | 2 84/9  | 98/8      | 108/8          | 66/6          | 102/11     | 106/6      | 72/7          |                |
| \( \text{imp. LU(c)} \) | 2 77/10 | 92/14     | 54/12          | 99/12         | 92/14      | 62/10      |                |                |
| \( \text{imp. LU(d)} \) | 2 118/10| 94/14     | 48/14          | 96/13         | 92/11      | 59/7       |                |                |
| \( \text{imp. LU(b)} \) | 5 69/9  | 89/12     | 92/13          | 53/12         |            | 95/12      | 55/10         |                |
| \( \text{imp. LU(c)} \) | 5 74/11 | 89/14     | 64/9           | 93/12         |            | 61/11      |                |                |
| \( \text{imp. LU(d)} \) | 5 122/10| 128/12    | 91/10          | 135/10        |            | 128/11     |                |                |
| \( \text{imp. LU(b)} \) | 10 62/8 | 77/10     | 88/9           | 58/6          | 79/10      | 83/9       | 58/10         |                |
| \( \text{imp. LU(c)} \) | 10 73/11| 99/7      | 106/9          | 66/9          | 97/10      | 90/11      | 68/10         |                |
| \( \text{imp. LU(d)} \) | 10 107/11| 119/9     | 126/10         | 121/8         | 67/11      | 123/13     | 119/11        | 73/11          |
| \( \text{imp. LU(b)} \) | 20 50/7 | 97/8      | 63/10          | 43/9          | 59/8       | 61/6       | 42/8          |                |
| \( \text{imp. LU(c)} \) | 20 70/11| 94/7      | 52/10          | 100/12        | 102/13     | 54/11      |                |                |
| \( \text{imp. LU(d)} \) | 20 96/12| 107/7     | 105/8          | 64/7          | 119/10     | 110/9      | 67/12         |                |
| \( \text{imp. LU(b)} \) | 50 29/7 | 42/5      | 30/11          | 35/9          | 20/8       | 36/10      | 39/11         | 19/7           |
| \( \text{imp. LU(c)} \) | 50 79/11| 94/8      | 88/10          | 96/10         | 61/7       | 98/12      | 89/14         | 62/10          |
| \( \text{imp. LU(d)} \) | 50 81/11| 95/9      | 91/10          | 96/10         | 62/7       | 99/12      | 92/12         | 63/10          |

\( \text{Block implicit} \)

|                | \( n_a \) | \( \text{LU(b)} \) | \( \text{LU(c)} \) | \( \text{LU(d)} \) | \( \text{LU(b)} \) | \( \text{LU(c)} \) | \( \text{LU(d)} \) |
|----------------|---------|----------------|----------------|----------------|----------------|----------------|----------------|
| \( \text{imp. LU} \) |          |                |                |                |                |                |                |
| \( \text{alg.[3]} \) | 130/8   | 116/11         | 82/11          | 143/7          | 119/11        | 92/12         | 164/7          | 110/11         | 108/11         |
| \( \text{imp. LU(b)} \) | 2 113/10| 94/7           | 64/10          | 137/5          | 97/7          | 86/7          | 153/5          | 98/8           | 96/9           |
| \( \text{imp. LU(c)} \) | 2 103/9 | 86/9           | 71/8           | 127/10         | 98/9          | 75/7          | 136/9          | 94/10          | 94/10          |
| \( \text{imp. LU(d)} \) | 2 135/6| 96/7           | 122/11         | 148/9          | 138/10        | 129/7         | 172/8          | 132/9          | 131/7          |
| \( \text{imp. LU(b)} \) | 5 111/10| 94/10          | 64/13          | 133/8          | 88/12         | 71/10         | 154/10         | 90/12          | 89/12          |
| \( \text{imp. LU(c)} \) | 5 108/9 | 99/11          | 70/11          | 133/6          | 85/13         | 81/9          | 150/11         | 97/14          | 96/12          |
| \( \text{imp. LU(d)} \) | 5 132/10| 122/10         | 116/11         | 150/8          | 135/11        | 121/12        | 174/7          | 128/10         | 127/10         |
| \( \text{imp. LU(b)} \) | 10 79/14| 63/8           | 126/9          | 89/11          | 72/9          | 131/8         | 82/10          | 81/10          |                |
| \( \text{imp. LU(c)} \) | 10 107/14| 60/10         | 187/8          | 88/14          | 80/11         | 130/10        | 79/14          | 79/14          |                |
| \( \text{imp. LU(d)} \) | 10 127/8| 115/16         | 140/8          | 115/15         | 112/15        | 166/6         | 117/15         | 117/15         |                |
| \( \text{imp. LU(b)} \) | 20 65/9 | 65/8           | 52/8           | 62/7           | 50/9          | 138/12        | 56/8           | 57/8           |                |
| \( \text{imp. LU(c)} \) | 20 109/11| 98/14         | 70/15          | 131/7          | 65/16         | 161/6         | 88/13          | 87/13          |                |
| \( \text{imp. LU(d)} \) | 20 119/9| 105/11         | 76/18          | 141/9          | 119/16        | 161/6         | 115/14         | 113/14         |                |
| \( \text{imp. LU(b)} \) | 50 81/8 | 41/10          | 28/6           | 50/12          | 33/6          | 33/6          | 32/10          | 31/10          |                |
| \( \text{imp. LU(c)} \) | 50 108/10| 108/13        | 68/15          | 136/8          | 99/13         | 71/14         | 158/10         | 5/187          |                |
| \( \text{imp. LU(d)} \) | 50 109/10| 108/13        | 68/15          | 135/8          | 100/13        | 73/14         | 159/10         | 8/187          |                |
Table 6 - Comparison of residual error

| Algorithm [3] | LU(b) | LU(c) | LU(d) | LU(b) | LU(c) | LU(d) |
|---------------|-------|-------|-------|-------|-------|-------|
| \( n_b \)     | 2     | 5     | 2     | 2     | 5     | 2     |
| 10            | 149/14 | 138/16 | 146/8 | 162/9 | 133/10 | 144/11 |
| 50            | 150/10 | 143/12 | 150/8 | 167/9 | 139/10 | 150/11 |
| 100           | 151/10 | 145/12 | 152/8 | 168/9 | 140/10 | 151/11 |
| 500           | 152/10 | 149/12 | 154/8 | 170/9 | 141/10 | 152/11 |

| Algorithm [3] | LU(b) | LU(c) | LU(d) | LU(b) | LU(c) | LU(d) |
|---------------|-------|-------|-------|-------|-------|-------|
| \( n_b \)     | 2     | 5     | 2     | 2     | 5     | 2     |
| 10            | 149/14 | 138/16 | 146/8 | 162/9 | 133/10 | 144/11 |
| 50            | 150/10 | 143/12 | 150/8 | 167/9 | 139/10 | 150/11 |
| 100           | 151/10 | 145/12 | 152/8 | 168/9 | 140/10 | 151/11 |
| 500           | 152/10 | 149/12 | 154/8 | 170/9 | 141/10 | 152/11 |

| Algorithm [3] | LU(b) | LU(c) | LU(d) | LU(b) | LU(c) | LU(d) |
|---------------|-------|-------|-------|-------|-------|-------|
| \( n_b \)     | 2     | 5     | 2     | 2     | 5     | 2     |
| 10            | 149/14 | 138/16 | 146/8 | 162/9 | 133/10 | 144/11 |
| 50            | 150/10 | 143/12 | 150/8 | 167/9 | 139/10 | 150/11 |
| 100           | 151/10 | 145/12 | 152/8 | 168/9 | 140/10 | 151/11 |
| 500           | 152/10 | 149/12 | 154/8 | 170/9 | 141/10 | 152/11 |