Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

For further information on
Serdica Bulgaricae Mathematicae Publicationes
and its new series Serdica Mathematical Journal
visit the website of the journal http://www.math.bas.bg/~serdica
or contact: Editorial Office
Serdica Mathematical Journal
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Telephone: (+359-2)9792818, FAX:(+359-2)971-36-49
e-mail: serdica@math.bas.bg

SERDICA — Bulgaricae
mathematicae publicationes
20 (1994) 193-207

BACKWARD STABILITY OF UL-DECOMPOSITION FOR LINEAR SYSTEMS ARISING FROM THE NONLINEAR INTEGRO–DIFFERENTIAL EQUATIONS OF KIRCHOFF TYPE

PL. YALAMOV, V. PAVLOV

Abstract. The backward stability of a special type of linear systems arising from integro-differential equations of Kirchoff type is studied. The estimates of the equivalent perturbations depend on the product $|U||L|$. Then an algorithm is proposed so as to evaluate how the equivalent perturbations of inputs influence on the final solution. An implementation on a network of three processors is described as well. Numerical expirements with random matrices are presented.

1. Introduction. When solving nonlinear problems of mathematical physics and mechanics the differential equations are approximated by systems of algebraic equatons (difference schemes). Usually these systems are solved by iteration methods [2]. The nonlinear algebraic systems arising from the discretization of the integro-differential equations of Kirchoff type [3]

$$u_{tt} - \varphi(\int_0^1 u_x^2\,dx)u_{xx} = f(x) \tag{1}$$

are very complex and have no band structure. The solution process demands the construction of stable algorithms. For simplicity instead of (1) in [1] the model problem

$$y'' - (\int_0^1 y^2\,dx)y' = f(x), \tag{2}$$

with boundary conditions

$$y(0) = \gamma_0,\ y(1) = \gamma_1 \tag{3}$$

is considered. Different approaches to the difference scheme obtained from (2) and (3) are tested. The result is that only Newton's iterations are convergent for a linear system whose iterations are solved by the algorithm described and examined more precisely in the present paper. The algorithm utilizes a special structure of the linear systems.

The direct discretization of (2) and (3) gives a nonlinear algebraic system with dense structure. After some computations, ommitted here, a nonlinear system, which is strongly sparse, is obtained [1]. The linearization of this system gives a linear system of the following form:

$$Ax = f, \tag{4}$$

where A is an $(m \times m)$-matrix,

$$A = \begin{pmatrix} b_{11} & b_{12} & a_1 & a_2 & \cdots & a_{m-2} \\ b_{21} & b_{22} & b & 0 & \cdots & 0 \\ c_1 & d_1 & q_1 & r_1 & & 0 \\ c_2 & d_2 & p_2 & q_2 & r_2 & \\ \vdots & \vdots & & \ddots & \ddots & \\ c_{m-2} & d_{m-2} & 0 & & p_{m-2} & q_{m-2} \end{pmatrix}$$

and f is an m-vector. It is clear that matrix A is strongly sparse, and we should make advantage of this property. In the present paper we shall consider only the solution of system (4), and we shall present round-off error analysis of the algorithm described in Section 2. For this purpose the method proposed in [4] is used and estimates of backward analysis [6] are obtained. These estimates serve to construct an algorithm for computing the estimate of the error in the solution x simultaneously with x. This algorithm does not result in a great increase of the running time.

2. Description of the algorithm. The algorithm for solving system (4) consists of three stages:

1. Calculation of the factors L and U, where $A = UL$, and

$$U = \begin{pmatrix} 1 & 0 & a_1^* & a_2^* & \cdots & a_{m-2}^* \\ & 1 & b & & & \\ & & \Delta_1 & r_1 & & 0 \\ & & & \ddots & \ddots & \\ & & & & \Delta_{m-3} & r_{m-3} \\ 0 & & & & & \Delta_{m-2} \end{pmatrix}, L = \begin{pmatrix} b_{11}^* & b_{12}^* & & & & \\ b_{21}^* & b_{22}^* & & & & 0 \\ c_1^* & d_1^* & 1 & & & \\ c_2^* & d_2^* & \alpha_2 & 1 & & \\ \vdots & \vdots & & \ddots & \ddots & \\ c_{m-2}^* & d_{m-2}^* & 0 & & \alpha_{m-2} & 1 \end{pmatrix}$$

$$\begin{aligned} \Delta_{m-2} &= q_{m-2}, \\ \alpha_{m-2} &= p_{m-2}/q_{m-2}, \\ c_{m-2}^* &= c_{m-2}/\Delta_{m-2}, \\ d_{m-2}^* &= d_{m-2}/\Delta_{m-2}, \\ \Delta_k &= q_k - r_k\alpha_{k+1}, \\ \alpha_k &= p_k/\Delta_k,\ p_1 = 0, \end{aligned} \tag{5}$$

$$\begin{aligned} c_k^* &= (c_k - r_k c_{k+1}^*)/\Delta_k, \\ d_k^* &= (d_k - r_k d_{k+1}^*)/\Delta_k, \\ a_{m-2}^* &= a_{m-2}, \\ a_k^* &= a_k - \alpha_{k+1} a_{k+1}^*, \quad k = m-3, \ldots, 1, \\ b_{11}^* &= b_{11} - (c^*, a^*), \\ b_{12}^* &= b_{12} - (d^*, a^*), \\ b_{21}^* &= b_{21} - b c_1^*, \\ b_{22}^* &= b_{22} - b d_1^*. \end{aligned}$$

2. Calculation of f^*, where $Uf^* = f$, and

$$\begin{aligned} f_m^* &= f_m/\Delta_{m-2}, \\ f_{k+2}^* &= (f_{k+2} - r_k f_{k+3}^*)/\Delta_k, \ k = m-3, \ldots, 1, \\ f_2^* &= f_2 - b f_3^*, \\ f_1^* &= f_1 - (f^*, a^*). \end{aligned} \tag{6}$$

3. Calculation of x, where $Lx = f^*$, and

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (B^*)^{-1} \begin{pmatrix} f_1^* \\ f_2^* \end{pmatrix}$$

$$x_{k+2} = f_{k+2}^* - c_k^* x_1 - d_k^* x_2 - \alpha_k x_{k+1}, \ k = 1, \ldots, m-2. \tag{7}$$

Here the first two components x_1 and x_2 of the solution x can be computed in different ways. In the numerical experiments we have used Gaussian elimination with pivoting. By c^*, d^*, a^* we have denoted the $(m-2)$-vectors, whose components are entries of L and U, and $f^* = (f_3^*, \ldots, f_m^*)^t$. The inner product of two arbitrary $(m-2)$-vectors y and z is denoted by (y, z).

3. Round-off error analysis. The method proposed in [4, 5] is used so as to obtain the estimates of backward analysis. This method is based on the graph of the algorithm and its parallel structure. It is a generalization of forward and backward analysis. In contrast to Wilkinson's backward analysis, the notion of equivalent perturbation is introduced for every piece of data (input,intermediate and output). Then a linear system with respect to ε,

$$B\varepsilon = \eta$$

is obtained, where ε is a first order approximation of the vector of all the equivalent perturbations, η is the vector of all the local absolute round-off errors, and B is a strongly sparse matrix, consisting of Frechet-derivatives of all the local operations and of elements which are equal either to 0, or to -1. The backward analysis of the three stages of the algorithm will be done separately. For this purpose we shall need two

theorems, which are proved in [5]. In these theorems we need the notion of replication. All of the arcs in the graph carry results but some of them may be replicas of the others. In this case we say that the arc (or equivalently the result) is replicated. We say also that we have replication in the corresponding vertex.

Theorem 1. *Let the following conditions be fulfilled:*
(i) the operands in all the additions and substractions are not replicated;
(ii) at least one operand in every multiplication or division is not replicated.
(iii) the algorithm consists of arithmetic operations only.
Suppose that a fixed parallel form of the graph is given and the equivalent perturbations of outputs are equal to zero. Then for the equivalent perturbations of every input a we have

$$\varepsilon_a = a \sum_{j=1}^{r(a)} \rho_j, \tag{8}$$

where $|\rho_j| \le 0.5p^{-t+1}$ *(p is the radix of the number system, t is the number of mantissa digits), and* $r(a)$ *is the length of the minimal path leading from the vertex where a is stored to the subset* V_m *consisting of vertices in which the outputs are obtained and of vertices the results of which are replicated.*

Corollary 1. *If b is an output, and* $\varepsilon_b = b\sum_j \rho_b^{(j)}$, *then for every input a, which is an argument of b (i. e. there is a path between the corresponding vertices* v_a *and* v_b*) and we have*

$$\varepsilon = a(\sum_{j=1}^{r(a)} \rho_j + \sum_j \rho_b^{(j)}). \tag{9}$$

If $r(a)+1$ *is less than the length of the path connecting* v_a *and* v_b *the second sum in (9) is missing.*

Theorem 2. *Suppose that at least one arc runs to every inner vertex from an input vertex, the corresponding input a is not replicated and* $|\partial f_a/\partial a| \ge c > 0$, *where* f_a *is the scalar function for which a is an argument. Then we have*

$$\varepsilon_a = (\eta_a + \varepsilon_y - \sum_{i=1}^{n} \frac{\partial f_a}{\partial x_i}\varepsilon_{x_i})/\frac{\partial f_a}{\partial a}, \tag{10}$$

where y is the result of f_a, $x_i, i = 1, \ldots, n$, *are outputs,* ε_{x_i} *are the corresponding equivalent perturbations,* ε_a *is the equivalent perturbation of a, and* η_a *is the absolute round-off error when calculating the value of* f_a. *In the case when* $\varepsilon_{x_i} = 0$, *we have*

$$|\varepsilon_a| \le (|\eta_a| + |\varepsilon_y|)/c. \tag{11}$$

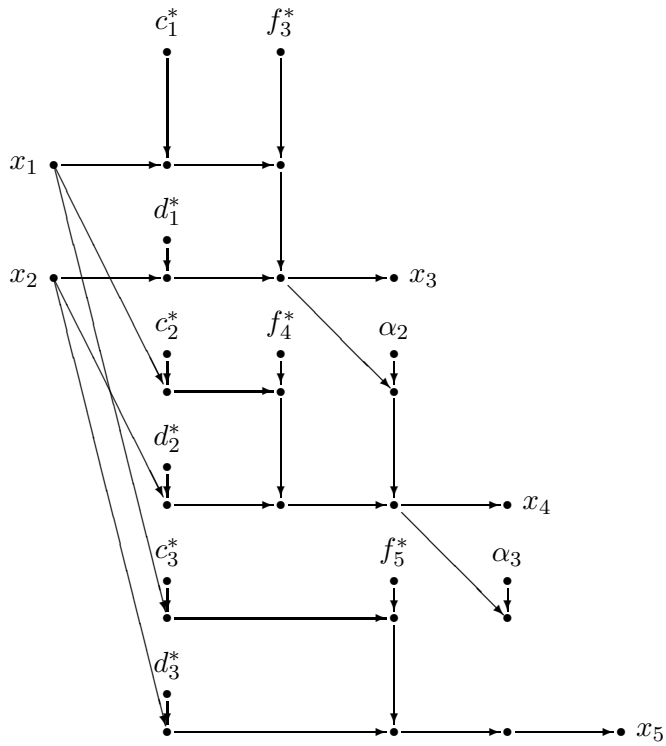

Figure 1: The graph of Stage 3 for m=5

Next we shall assume that x_1, x_2 are computed in double precision, i.e. we shall not analyse the computation of these components. The analysis of their computation will not change the final estimates considerably.

In order to apply Theorem 1 and Theorem 2 we need the graphs of the corresponding stages. The graphs of stages 2 and 3 are shown in Fig. 1 and Fig.2 respectively for the case $m = 5$. Evidently these graphs satisfy the conditions of Theorem 1. In the following we shall denote by ε_s the equivalent perturbation of s, where s can be a number , a vector or a matrix. So, from Theorem 1 for Stage 3 (7) we can write straightforward that

$$\varepsilon_{c_k^*} = c_k^* \sum_{l=1}^{3} \rho_l^{(k)}, \tag{12}$$

$$\varepsilon_{d_k^*} = d_k^* \sum_{l=1}^{4} \tau_l^{(k)},$$

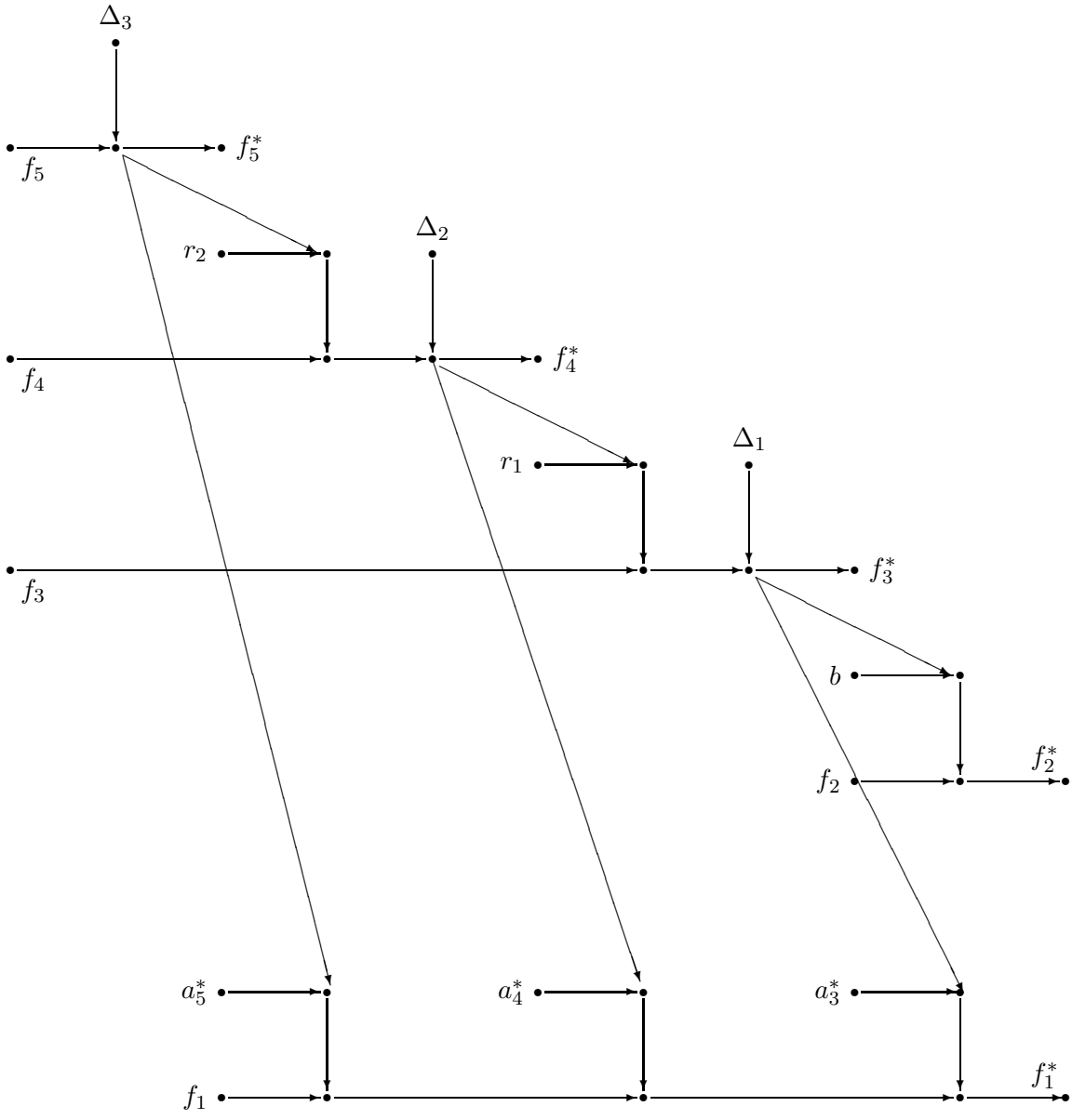

Figure 2: The graph of Stage 2 ($Uf^* = f$)

$$\begin{aligned} \varepsilon_{\alpha_k} &= \alpha_k \sum_{l=1}^{4} \tau_l^{(k)}, \\ \varepsilon_{f^*_{k+2}} &= f^*_{k+2}\mu^{(k+2)}, \end{aligned}$$

$$|\rho_l^{(k)}|, |\sigma_l^{(k)}|, |\tau_l^{(k)}|, |\mu^{(k+2)}| \le \varepsilon_0,$$

$$k = 1, \ldots, m-2,$$

where ε_0 is the relative machine precision (e.g., for symmetric rounding we have $\varepsilon_0 = 0.5p^{-t+1}$, p is the radix of the number system, t is the number of the mantissa digits). From (12) it follows that

$$\begin{aligned} |\varepsilon_{c^*_k}| &\le 3|c^*_k|\varepsilon_0, \\ |\varepsilon_{d^*_k}| &\le 4|d^*_k|\varepsilon_0, \\ |\varepsilon_{\alpha_k}| &\le 4|\alpha_k|\varepsilon_0, \\ |\varepsilon_{f^*_{k+2}}| &\le |f^*_{k+2}|\varepsilon_0, \\ k &= 1, \ldots, m-2. \end{aligned} \tag{13}$$

Similarly by using Corollary 1 for Stage 2 (6) we have

$$\begin{aligned} |\varepsilon_{a^*_k}| &\le (k+2)|a^*_k|\varepsilon_0, \quad k = 1, \ldots, m-2, \\ |\varepsilon_b| &\le 3|b|\varepsilon_0, \\ |\varepsilon_{r_k}| &\le 4|r_k|\varepsilon_0, \quad k = 1, \ldots, m-2, \\ \varepsilon_{\delta_k} &= 0, \quad k = 1, \ldots, m-2, \\ |\varepsilon_{f_1}| &\le (m-1)|f_1|\varepsilon_0, \\ |\varepsilon_{f_2}| &\le 2|f_1|\varepsilon_0, \\ |\varepsilon_{f_k}| &\le 3|f_k|\varepsilon_0, \quad k = 3, \ldots, m. \end{aligned} \tag{14}$$

It is easy to check that Stage 1 satisfies the conditions of Theorem 2. We shall not need the graph of this stage explicitly for this case. For the computation of each entry of L or U there is an argument, which is an input for the whole algorithm. Let us consider, for example, that the computation of a^*_k is done for a vertex. Then Theorem 2 gives

$$\varepsilon_{a_k} = \eta_{a_k} + \varepsilon_{a^*_k} + \varepsilon_{\alpha_{k+1}} a^*_{k+1} + \varepsilon_{a^*_{k+1}} \alpha_{k+1}.$$

Here η_{a_k} is a first order approximation of the local absolute round-off error which is defined by the following expression [6]:

$$\eta_{a_k} = a_k \nu_1^{(k)} - \alpha_{k+1} a^*_{k+1} (\nu_2^{(k)} + \nu_1^{(k)}),$$

$$|\nu_i^{(k)}| \le \varepsilon_0, i = 1, 2,$$

and hence,

$$|\eta_{a_k}| \le (|a_k| + 2|\alpha_{k+1}a^*_{k+1}|)\varepsilon_0. \tag{15}$$

In the following by $|Z|$, where Z is an arbytrary vector or matrix, we denote the corresponding vector or matrix, which entries are the absolute values of the entries in Z. Now, from (13), (14), (15) we have

$$|\varepsilon_{a_k}| \le [(k+2)|a^*_k| + (k+9)|\alpha_{k+1}a^*_{k+1}| + |a_k|]\varepsilon_0 = [(k+9)|U_1||L_{k+2}| + |a_k|]\varepsilon_0.$$

Here and further on U_i and L_i, $i = 1, \ldots, m$, the i-th row of U and the i-th column of L are denoted, respectively.

The other estimates of the rest of the remaining inputs are obtained in a similar way from (5),(13),(14) and we shall give them directly:

$$\begin{aligned}
|\varepsilon_{c_k}| &\le (3|c^*_k\Delta_k| + 10|r_k c^*_{k+1}| + 2|c_k|)\varepsilon_0 \le (10|U_{k+2}||L_1| + 2|c_k|)\varepsilon_0, \\
|\varepsilon_{d_k}| &\le (4|d^*_k\Delta_k| + 11|r_k d^*_{k+1}| + 2|d_k|)\varepsilon_0 \le (11|U_{k+2}||L_2| + 2|d_k|)\varepsilon_0, \\
|\varepsilon_{b_{11}}| &\le \sum_{k=1}^{m-2}((k+6)|a^*_k c^*_k| + |b_{11}|)\varepsilon_0 \le ((m+6)|U_1||L_1| + |b_{11}|)\varepsilon_0, \\
|\varepsilon_{b_{12}}| &\le \sum_{k=1}^{m-2}((k+7)|a^*_k d^*_k| + |b_{12}|)\varepsilon_0 \le ((m+7)|U_1||L_2| + |b_{12}|)\varepsilon_0, \\
|\varepsilon_{b_{21}}| &\le (8|bc^*_1| + |b_{21}|)\varepsilon_0 = (8|U_2||L_1| + |b_{21}|)\varepsilon_0, \\
|\varepsilon_{b_{22}}| &\le (9|bd^*_1| + |b_{22}|)\varepsilon_0 = (9|U_2||L_2| + |b_{22}|)\varepsilon_0, \\
|\varepsilon_{p_k}| &\le 5|p_k|\varepsilon_0 = 5|U_{k+2}||L_{k+1}|\varepsilon_0, \\
|\varepsilon_{q_k}| &\le (10|\alpha_{k+1}r_k| + |q_k|)\varepsilon_0, \\
|\varepsilon_{r_k}| &\le 4|r_k|\varepsilon_0 = 4|U_{k+2}||L_{k+1}|\varepsilon_0.
\end{aligned} \tag{16}$$

These estimates can be rewritten in a more compact form as follows:

$$|\varepsilon_a| \le [C_1(m)|U||L| + C_2|A|]\varepsilon_0, \quad |\varepsilon_f| \le C_3(m)|f|\varepsilon_0 \tag{17}$$

where $C_1(m)$ and $C_3(m)$ are piecewise linear functions of m, and C_2 does not depend on m. It is clear that the algorithm described in Section 2 is backward stable, if the product $|U||L|$ is not very large.

4. An algorithm for estimating the round-off error. The inequalites (17) depend on the product $|U||L|$. If matrix A has some special properties (e.g. positive definite, M-matrix, etc) this product can be expressed further on by the entries of A. The matrices arising from the Newton's method for equation (2) are more general. For this reason in the next section we propose an algorithm which computes the estimate

of the round-off error in the solution x together with x. This algorithm is based on the following note. By definition of backward analysis [6] we have

$$(A + \varepsilon_A)\tilde{x} = f + \varepsilon_f, \tag{18}$$

where $\tilde{x} = x + \delta x$ is the solution computed with round-off errors, and δx is the error. Now we can derive from (18) that

$$x - \tilde{x} = \delta x = -A^{-1}\varepsilon_A\tilde{x} + A^{-1}\varepsilon_f. \tag{19}$$

Having in mind that $|A^{-1}| \leq |L^{-1}||U^{-1}|$, from (17) and (19) for an arbitrary norm we have

$$\|\delta x\| \leq [\| \, |L^{-1}||U^{-1}|(C_1(m)|U||L| + C_2|A|)\| \, \|\tilde{x}\| + \| \, |L^{-1}||U^{-1}|C_3(m)|f| \, \|]\varepsilon_0. \tag{20}$$

It is difficult to compute estimate (20). That is why we introduce the following operator F which maps an arbitrary matrix D onto a vector as follows:

$$F(D) = (\|D_1\|_1, \|D_2\|_1, \ldots, ||D_m\|_1)^T, \tag{21}$$

where D_i, $i = 1, \ldots, m$, is the i-th row of D, and $\|z\|_1 = \sum_{k=1}^{m} |z_k|$ for an arbitrary vector z.

Lemma 1. *Let D and E be a square $(m \times m)$ matrix with nonnegative entries, i.e. $D \geq 0, E \geq 0$. Then we have*

$$\|DE\|_\infty = \|DF(E)\|_\infty. \tag{22}$$

P r o o f. Let $E = \{e_{ij}\}_{i,j=1}^m, D = \{d_{ij}\}_{i,j=1}^m$. Then (22) is proved by the following equalities, which follow from the definition of F (21):

$$\|DE\|_\infty = \max_i \sum_j d_{ik}e_{kj} = \max_i \sum_k d_{ik}\|E^{(k)}\|_1 = \|DF(E)\|_\infty,$$

where $E^{(k)}$ is the k-th column of E. □

In (20) we have $D = |L^{-1}||U^{-1}|, E = C_1(m)|U||L| + C_2|A|$. Further on we shall use the infinity norm. So by Lemma 1 instead of $\|DE\|_\infty$ in (20) we shall consider $\|DF(E)\|_\infty$. The following Lemma gives an algorithm to find an estimate of $\|DF(E)\|_\infty$.

Lemma 2. *Let $T = \{t_{ij}\}_{i,j=1}^m$ be an upper triangular nonsingular $(m \times m)$ matrix, and let $f \geq 0$ be an m-vector. Let $h = |T^{-1}|f$ and*

$$\begin{aligned} h_k^* &= f_k + \sum_{j=k+1}^{m} |t_{kj}|h_j^*, k = 1, \ldots, m-1, \\ h_m^* &= h_m. \end{aligned}$$

Then $h_k^* \geq h_k, k = 1, \ldots, m.$

P r o o f. We shall use induction on k to prove the lemma. Evidently, for $k = m$ the statement is true. Let us suppose that $h_j^* \geq h_j, j = k+1, \ldots, m$. Denote by t_{ij}^{-1} the entries of the inverse T^{-1}. Then we have

$$\begin{aligned} h_k^* &= [f_k + \sum_{j=k+1}^{m} |t_{kj}| h_j^*]/|t_{kk}| \geq \\ &\geq |t_{kk}^{-1}| f_k + [\sum_{j=k+1}^{m} |t_{kj}| \sum_{i=k+1}^{m} |t_{ji}^{-1}| f_i]/|t_{kk}| = \\ &= |t_{kk}^{-1}| f_k + [\sum_{i=k+1}^{m} f_i \sum_{j=k+1}^{m} |t_{kj} t_{ji}^{-1}|]/|t_{kk}|. \end{aligned} \tag{23}$$

From the equality $TT^{-1} = I$, where I is the coresponding identity matrix, if follows that

$$\sum_{j=k+1}^{i} |t_{kj} t_{ji}| \geq |\sum_{j=k+1}^{i} t_{kj} t_{ji}^{-1}| = |\sum_{j=k}^{i} t_{kj} t_{ji}^{-1} - t_{kk} t_{ki}^{-1}| = |t_{kk} t_{ki}^{-1}|. \tag{24}$$

Now (23) and (24) give

$$h_k^* \geq |t_{kk}^{-1}| f_k + \sum_{i=k+1}^{m} |t_{ki}^{-1}| f_i = h_k. \quad \square$$

Now we can formulate the following algorithm for estimating the expression in the square brackets of (20).

1. Calculate the entries of $E = C_1(m)|U||L| + C_2|A|$ from (16).
2. Calculate the entries of the vectors $v = F(E)$ and $w = C_3(m)|f|$ from (14).
3. Calculate an estimate of $v^* = |u^{-1}|v$ and $w^* = |u^{-1}|w$ according to Lemma 2.
4. Calculate an estimate of $v^{**} = |L^{-1}|v^*$ and $w^{**} = |L^{-1}|w^*$ according to Lemma 2.
5. Calculate the infinity norms of $v^{**}, \tilde{x}$ and w^{**} and estimate δx from (20).

Let us note that this algorithm will take $(59m - 82)$ arithmetic operations, which is about 2.27 times the aritmetic operations of algorithm (5), (6), (7) ($(26m-44)$ arithmetic operations), and the number of operations of the two algorithms is of the same order.

5. A parallel implementation. Algorithm (5)-(7) is easily implemented on a network of 3 processors with local memory. For our aims we shall give the graph of one step at both Stages 1 and 2 together (fig 3.). Here

$$b_{11}^{(m-2)} = b_{11},$$

δx	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-13}
err	10^{-5}	10^{-6}	10^{-5}	10^{-5}	10^{-6}	10^{-7}	10^{-7}	10^{-12}	10^{-13}	10^{-13}	10^{-13}	10^{-16}
			10^{-6}	10^{-6}	10^{-7}		10^{-13}	10^{-13}	10^{-14}	10^{-14}	10^{-14}	
			10^{-7}	10^{-7}				10^{-14}	10^{-15}	10^{-15}	10^{-15}	
N	1	1	11	17	9	1	5	7	32	64	13	1

Table 1: Matrices with random entries.

$$
\begin{aligned}
b_{12}^{(m-2)} &= b_{12}, \\
f_1^{(m-2)} &= f_1, \\
b_{11}^{(k)} &= b_{(k+1)} - c_{k+1}^* a_{k+1}^*, \\
b_{12}^{(k)} &= b_{12}^{(k+1)} - d_{k+1}^* a_{k+1}^*, \\
f_1^{(k)} &= f_1^{(k+1)} - f_{k+1}^* a_{k+1}^*, \\
b_{11}^* &= b_{11}^{(0)}, \\
b_{12}^* &= b_{12}^{(0)}, \\
f_1^{(0)} &= f_1^*.
\end{aligned}
$$

Now we show that the algorithm has a good load balance, and all the processors are busy at almost each step. It is clear that the three processors should perform as a pipeline, for which the input data flow is $\alpha_{k+1}, c_{k+1}^*, d_{k+1}^*, f_{k+3}^*, a_{k+1}^*$ for each k, and the output data flow is $\alpha_k, c_k^*, d_k^*, f_{k+2}^*, a_k^*$, in the given order. The operations lying on one and the same dashed line (Fig. 3) are executed at one and the same time. The inputs r_k should be in the memory of processor 1 ($p1$), the inputs q_k, c_k, d_k, f_{k+2} should be in the memory of processor 2 ($p2$), and p_k and a_k in the memory of processor 3 ($p3$). When each processor conpletes the computation of the k-th step , it begins to execute the operations of the next step. So, only processor $p1$ does not work for one tact on each cycle of 5 tacts, and processors $p2$ and $p3$ are fully loaded. The graph in Fig.3 shows that the intermediate data are transferred only in one direction in the network, and at each phase of communication only one number is transferred.

Now let us proceed with Stage 3. If we consider that each vertex contains an operation of the form $a + bc$, then the graph of Stage 3 is given in Fig.4. Here we assume that x_1 and x_2 are computed somehow (by Cramer's rule, for example).

This graph maps naturally along the k-axis on a graph consisting of three vertices (excluding the input and output vertices). The latter is the graph of the computing architecture. Again the data flow is only in one direction. The vectors f^* and c^* should be stored in the memory of $p1$, the record d^* should be stored in the

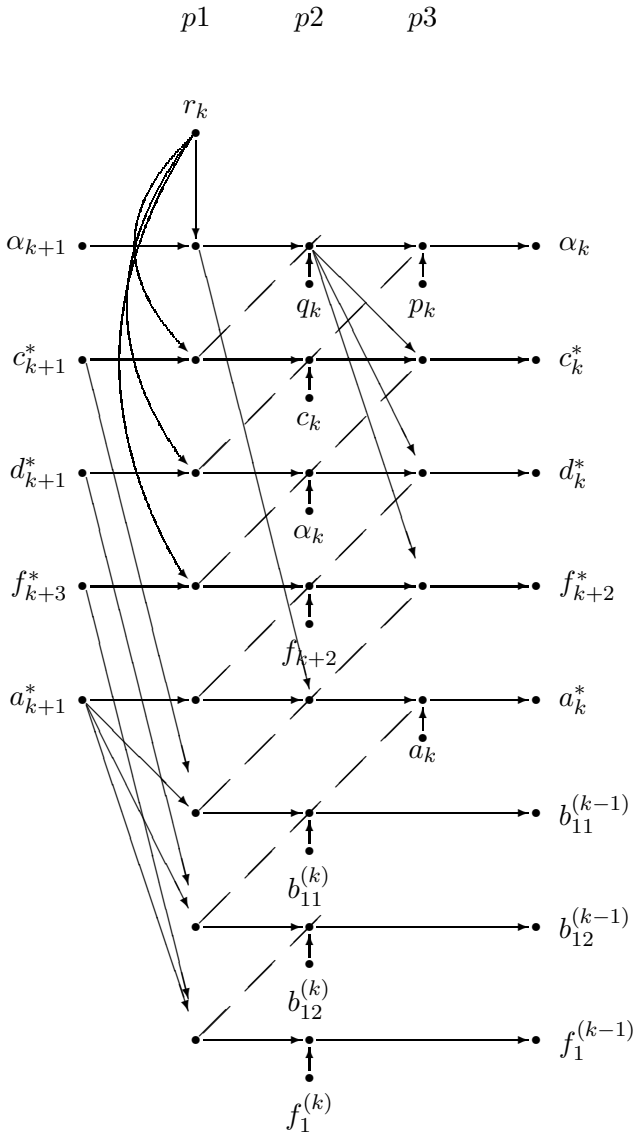

Figure 3: The graph of steps 1 and 2

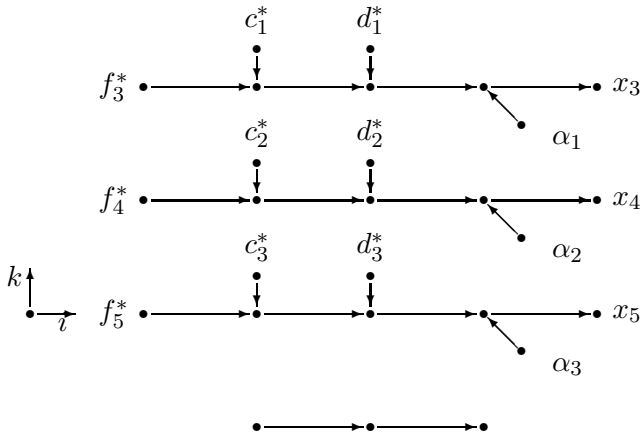

Figure 4: The graph of stage 3 for $m = 5$

δx	10^{22}	10^{60}	10^{95}	10^{129}	10^{163}	10^{198}
err	10^{11}	10^{30}	10^{49}	10^{68}	10^{85}	10^{103}

Table 2: Ill-conditioned matrices with random entries.

m	50	100	200	500	750
δx	0.5×10^{-9}	6.3×10^{-8}	9.1×10^{-7}	3.3×10^{-5}	1.7×10^{-4}
err	3.6×10^{-11}	5.8×10^{-10}	3.4×10^{-9}	2.9×10^{-7}	2.9×10^{-7}

Table 3: The results for the matrix given in Case 1.

m	50	100	200	500	750
δx	1.5×10^{-11}	3.8×10^{-11}	9.8×10^{-11}	3.9×10^{-10}	$7.7 \times s10^{-10}$
err	1.5×10^{-13}	2×10^{-13}	1.4×10^{-13}	1.8×10^{-11}	3.2×10^{-11}

Table 4: The results for the matrix given in Case 2.

m	50	100	200	500	750
δx	1×10^{-11}	2.6×10^{-11}	6.7×10^{-11}	2.8×10^{-10}	5.7×10^{-10}
err	1.2×10^{-13}	1.5×10^{-13}	3.6×10^{-13}	6×10^{-12}	2.5×10^{-11}

Table 5: The results for the matrix given in Case 3.

memory of $p2$, and $\alpha_i, i = 1, \ldots, m-2$ in the memory of $p3$. The solution is obtained in the memory of $p3$.

If we consider that the time for each arithmetic operation is one and the same, and this is the time unit and neglect the communication time, then the time cost for the parallel implementation of the algorithm is $10m$-14, and, as we noticed above, the time cost for the sequential implementation is 26m-44, hence, the speedup is $S_3 \approx 2.6$ on the network of three processors. Obviously, the efficiency is approximately $E_3 \approx 0.86$.

6. Numerical experiments. The code for the numerical experiments has been written in Fortran 77 and $\varepsilon_0 \approx 10^{-16}$. Table 1 shows the results of the experiments with matrices for which $p_i = p, q_i = q, r_i = r, p+r = q$ (p, r - random numbers), because this the most important case in the applications. The right hand side is chosen so that the exact solution is $x = (1, 1, \ldots, 1)^T$. Each entry of the first row of Table 1 shows the magnitude of δx, the entry below shows the magnitude of the real errors, and the entry below them shows how many matrices have been obtained as a result. Equal number of matrices of order $m = 50, 100, 200, 500$ was generated.

Table 2 shows some of the results for very ill-conditioned matrices. The results are analogous and δx is large where the error is large.

In the last three examples (Tables 3,4,5) the matrices are different for the entries p_i and q_i:

$$b_{11} = (2m^3 - 9m^2 + 10m + 12)/12,$$

$$b_{12} = b_{21} = -1,$$

$$b_{22} = m + 1 - \frac{1}{m+1},$$

$$a_i = c_i = d_i = r_i = -1, i = 1, \ldots m-2.$$

The entries p_i and q_i for the different tables are as follows:

Case 1: Table 3 - $p_i = -1, q_i = 2$.

Case 2: Table 4 - $p_i = -1 - 1/9, q_i = 2 + 1/9$.

Case 3: Table 5 - $p_i = -1 - 1/7, q_i = 2 + 1/7$.

The right hand side f in all the examples are chosen so that the exact solution is $x = (1, 1, \ldots, 1)^T$.

Let us note that in all the examples the method bounds the real error, and in some of the examples δx is very close to err (Tables 4 and Tables 5). For very ill-conditioned matrices δx is far from err, but nevertheless shows that the error is large.

REFERENCES

[1] P. Y. Yalamov, V. T. Pavlov. On the stability of algorithms for numerical solution of Kirchoff type integro-differential equations. *Mathematics and Education in Mathematics*, **21** (1992), 173–179.

[2] J. Ortega, W. Rheinnboldt. Iterative solution of nonlinear equations in several variables. Academic press, New York and London, 1970.

[3] G. Kauderer. Nonlinear mechanics. Inostrannaya literatura, Moscow, 1961 (in Russian).

[4] V. V. Voevodin, P. Y. Yalamov. A new method of round-off error estimation. Proc. Workshop on Parallel and Distributed Processing (ed. K. Boyanov), March 27-29, 1990, Sofia., Elsevier, Amsterdam, 1990, 315-333.

[5] P. Y. Yalamov. Round-off error and graphs. Preprint N4, November 1991, Technical University, Rousse.

[6] J. H. Wilkinson. The algebraic eigenvalue problem. Clarendon Press, Oxford, 1965.

Centre of Mathematics
Technical University
7017 Rousse
BULGARIA

Received 28.10.1992
Revised 08.07.1994