ORIGINAL PAPER

A PARALLEL SOLUTION OF TRIDIAGONAL LINEAR SYSTEMS BY CONTINUED FRACTIONS

DUMITRU FANACHE

Valahia University of Targoviste, Faculty of Science and Arts, 130024, Targoviste, Romania

Abstract. *In paper it is report the LU decomposition of tridiagonal matrix to evaluate continued fractions. Application of parallel suffix while applying parallel prefix products leads us to an optimal algorithm for LU decomposition that runs in* $O(\log n)$ *parallel time with* $O\left(\frac{n}{\log n}\right)$ *processors, where* n *is the size of the tridiagonal matrix.*

Keywords: *tridiagonal system, LU decomposition, continued fraction, prefix product, suffix product.*

1. INTRODUCTION

Tridiagonal systems of equations arise frequently in the resolving of partial differential equations are given for the various solutions on parallel architectures. As is known, for example, *Black Scholes equation* is used to calculate the value of an option. In many cases, for example, a European-type option, this equation gives us the exact solution, but for other, more complex, it is necessary to apply numerical methods to obtain an approximate solution and application leads us precisely to resolve such particular linear systems [4].

We present an algorithm for calculating the first n convergents of a general continued fraction using this opportunity for his representation with a product of the matrices 2×2 type mentioned in [1] and a parallel prefix similar to that given in [2, 3] for effectively resolve these dot-matrix products.

LU decomposition algorithm is one of the most efficient existing sequential algorithms for the solution of linear systems with nonsymmetric tridiagonal matrix. To identify parallel opportunities of *LU* decomposition, we determine the convenient recurrence relations between elements of decomposition matrices, so that through them to achieve in fact a method of calculating the same as that conducted to the convergents evaluation of continuous fractions.

It operates so that the product of a matrix operation is associative, so perfect parallelization.

2. THE RELATIONSHIP BETWEEN *LU* DECOMPOSITION AND CONTINUED FRACTIONS

To consider linear system under a more general form as follows:

$$Ax = d \tag{1}$$

where A is a tridiagonal matrix of order n by form (2), $x = (x_1, x_2, \cdots, x_n)^T$ is unknowns vector $d = (d_1, d_2, \cdots, d_n)^T$ is free terms vector, both size n:

Corresponding author: dfanache@gmail.com

$$A=\begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & a_n & b_n \end{pmatrix} \tag{2}$$

where $a_1 = 0$ şi $c_n = 0$.

In the LU factorization, matrix A is decomposed into a product of two bidiagonal matrices L and U. So $A = LU$, where:

$$L=\begin{pmatrix} 1 & & & & \\ e_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & e_{n-1} & 1 & \\ & & & e_n & 1 \end{pmatrix} \quad \text{and} \quad U=\begin{pmatrix} f_1 & c_1 & & & \\ & f_2 & c_2 & & \\ & & \ddots & \ddots & \\ & & & f_{n-1} & c_{n-1} \\ & & & & f_n \end{pmatrix} \tag{3}$$

For example, in (1), matrix A and respectively free terms vector d are:

$$A=\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \qquad d=\begin{pmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}^T \tag{4}$$

find the vector solution $x=\begin{pmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{pmatrix}^T$. LU algorithm for to solve system (1) consists in determining the vector y from system $Ly = d$ and then solving the system $Ux = y$.

More precisely, this algorithm for solving system (1) consists of the following steps:

Step I. Calculate the LU decomposition of the matrix A as follows:

$$f_1 = b_1$$
$$e_i = a_i / f_{i-1}, \quad 2 \le i \le n$$
$$f_i = b_i - e_i * c_{i-1}, \quad 2 \le i \le n$$

```
f(1)=b(1);
for i=2:n
    e(i)=a(i)/f(i-1);
    f(i)=b(i)-e(i)*c(i-1);
end
```

Step II. Determines y from system $Ly = d$ using:

$$y_1 = d_1$$
$$y_i = d_i - e_i * y_{i-1}, \quad 2 \le i \le n$$

```
y(1)=d(1);
for i=2:n
   y(i)=d(i)-e(i)*y(i-1);
end
```

Step III. Determines x by solving system $Ux = y$ using:

$$x_n = y_n / f_n$$
$$x_i = \left(y_i - c_i * x_{i+1}\right) / f_i, \quad n-1 \ge i \ge 1$$

```
x(n)=y(n)/f(n);
for i=n-1:-1:1
   x(i)=(y(i)-c(i)*x(i+1))/f(i);
end
[L U]=lu(A);
```

Let's consider first the parallelization LU decomposition part of the LU algorithm for solving the system (1), ie *Step* I above. Once the diagonal values $f_1, f_2, \cdots, f_n$ of U have been calculated, $e_2, e_3, \cdots, e_n$ can then be obtained in one parallel step with $n-1$ processors. We will focus first on determining f_i values. Let

$$\alpha_i = b_{n-i+1}, \qquad \text{for } 1 \le i \le n \tag{5}$$

and

$$\beta_i = -a_{n-i+2} * c_{n-i+1}, \qquad \text{for } 2 \le i \le n \tag{6}$$

According to (5) and (6), f_i satisfy the non-linear recurrence relation

$$f_1 = \alpha_n$$
$$f_i = \alpha_{n-i+1} + \beta_{n-i+2} / f_{i-1}, \quad 2 \le i \le n \tag{7}$$

```
f(1)=alfa(n);
for i=2:n
 f(i)=alfa(n-i+1)+beta(n-i+2)/f(i-1);
end
```

Thus, the system given by (4), we obtain the results in Table 1.

Table 1. Elements of matrices L respectively U after decomposition of the matrix given in (4)

α	2	2	2	2	2	2	2	2
β	1	-1	-1	-1	-1	-1	-1	-1
e_i	0	-0.50000	-0.6667	-0.7500	-0.8000	-0.8333	-0.8571	-0.8750
f_i	2.0000	1.5000	1.3333	1.2500	1.2000	1.1667	1.1429	1.1250

Determining matrices L and U involves determining e_i, respectively f_i, as matrix factorization given by (4) is:

$$L = \begin{pmatrix} 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.5000 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.6667 & 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.7500 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -8.0000 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.8333 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.8571 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.8750 & 1.0000 \end{pmatrix}$$

respectively

$$U = \begin{pmatrix} 2.0000 & -1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.5000 & -1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.3333 & -1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.25000 & -1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.2000 & -1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.1667 & -1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.1429 & -1.0000 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.1250 \end{pmatrix}$$

It follows that if the general case of system given by (1) gets extended continued fraction $f_i, i=\overline{1,n}$ by form:

$$f_1 = \alpha_n ; f_2 = \alpha_{n-1} + \frac{\beta_n}{\alpha_n}; f_3 = \alpha_{n-2} + \cfrac{\beta_{n-1}}{\alpha_{n-1} + \cfrac{\beta_n}{\alpha_n}}; \cdots ; f_n = \alpha_1 + \cfrac{\beta_2}{\alpha_2 + \cfrac{\beta_3}{\ddots \cfrac{\vdots}{\alpha_{n-1} + \cfrac{\beta_n}{\alpha_n}}}} \tag{8}$$

Therefore, we aim to find a possibility of parallel evaluation continued fractions $f_i, i=\overline{1,n}$ given by (8).

3. THE RELATIONSHIP BETWEEN CONTINUED FRACTIONS AND PARALLEL PREFIX/ SUFFIX ALGORITHMS

Let a sequence $a[1\cdots n]$ with elements of T type and a binary associative operation $\oplus : T\times T \to T$, elements $Pr[1\cdots n]$ with $Pr[k] = \oplus_{i=1}^{k} a_i \ (\forall) 1\le k\le n$ are called *prefixes* of $a[1\cdots n]$ sequence and elements $Suf[1\cdots n]$ with $Suf[k] = \oplus_{i=1}^{k} a_{n-i+1} \ (\forall) 1\le k\le n$ are called suffixes of same sequence. To consider the reduced continued fraction of n order:

$$\frac{p_n}{q_n} = \alpha_1 + \cfrac{\beta_2}{\alpha_2 + \cfrac{\beta_3}{\ddots \cfrac{\vdots}{\alpha_{n-1} + \cfrac{\beta_n}{\alpha_n}}}} = \alpha_1 + \frac{\beta_2}{\alpha_2 +}\frac{\beta_3}{\alpha_3 +}\cdots\frac{\beta_n}{\alpha_n} \tag{9}$$

for any n. Under the matrix formulation given by Milne-Thomson [1], relation (9) can be written as:

$$\begin{bmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{bmatrix} = \begin{bmatrix} \alpha_1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \alpha_2 & 1 \\ \beta_2 & 0 \end{bmatrix}\cdots\begin{bmatrix} \alpha_n & 1 \\ \beta_n & 0 \end{bmatrix}$$

or

$$\begin{bmatrix} p_n \\ q_n \end{bmatrix} = \begin{bmatrix} \alpha_1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \alpha_2 & 1 \\ \beta_2 & 0 \end{bmatrix}\cdots\begin{bmatrix} \alpha_n & 1 \\ \beta_n & 0 \end{bmatrix}\cdot\begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{10}$$

Next we define:

$$A_k = \begin{bmatrix} \alpha_k & 1 \\ \beta_k & 0 \end{bmatrix} \text{ for } 1 \le k \le n \qquad (11)$$

with $\beta_1 = 1$, with (11), we can write (10) more concise, so:

$$\begin{bmatrix} p_n \\ q_n \end{bmatrix} = A_1 A_2 A_3 \cdots A_n . \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad (12)$$

Using Matlab sequence below, we determine the f_c vector, identical with the f vector, which argues the result of [1] that the f_i can be calculated as a prefix product matrix ([2]) according to relations (10).

```
for i=1:n
  for k=1:i
if(k==1) x=[alfa(k) 1; beta(k) 0];
else
  y=[alfa(k) 1;beta(k) 0];
  x=x*y;
end
end
x=x*[1 0]';
  fc(i)=x(1)/x(2);
end
```

Therefore all convergents (reduceds) $\dfrac{p_i}{q_i}$ for $i = 1,2,\cdots,n$ of general continued fraction can be calculated in $O(\log n)$ parallel time [3, 5] using $O\left(\dfrac{n}{\log n}\right)$ processors, having only needed the prefix product:

$$A_1;\ A_1 A_2; \cdots; A_1 A_2 \cdots A_n \qquad (13)$$

of matrices A_i by 2×2 order for $1 \le i \le n$ and a parallel prefix algorithm can be used to calculate the products (13). However the f_i of (8) are not the convergents a single continuous fractions [1]. In fact, the direct application of this fast parallel evaluation the calculation of finite continued fractions for $f_1, f_2 \cdots, f_n$ of (8) requires:

$$O\left(\sum_{k=2}^{n} \frac{k}{\log k}\right) = O\left(\int_2^n \frac{x}{\log x} dx\right) = O\left(\frac{n^2}{\log n}\right)$$

processors and requires $O\left(\sum_{k=2}^{n} \log k\right) = O\left(\log^2 n\right)$ parallel time. Such the prefixes in (13) calculated continued fractions in the wrong direction as far as f_i are concerned.

Further, we can show the products of suffix [2]:

$$A_n;\ A_{n-1}A_n;\cdots;\ A_2A_3\cdots A_n \tag{14}$$

can be used effectively to determine the f_i values in parallel. To do this, write each f_i of (8) as:

$$\begin{aligned} f_1 &= \alpha_n \\ f_i &= \alpha_{n-i+1} + \frac{r_i}{s_i}, \quad 2 \le i \le n \end{aligned} \tag{15}$$

We consider also continued fraction defined in (9):

$$\alpha_1 + \frac{\beta_2}{\alpha_2 +}\frac{\beta_3}{\alpha_3 +}\cdots\frac{\beta_n}{\alpha_n} \tag{16}$$

Then

$$\frac{r_i}{s_i} = \frac{\beta_{n-i+1}}{\alpha_{n-i+1} +}\frac{\beta_{n-i+2}}{\alpha_{n-i+2} +}\cdots\frac{\beta_n}{\alpha_n} \quad \text{for} \quad 1 \le i \le n-1 \tag{17}$$

From (15), once the values of r_i and s_i for $2 \le i \le n$ are known $f_2, f_2, \cdots, f_n$ can be calculated with 2 parallel arithmetic operations using $n-1$ processors. To calculate of r_i and s_i, we argued that:

$$\begin{bmatrix} s_i \\ r_i \end{bmatrix} = \begin{bmatrix} \alpha_{n-i+1} & 1 \\ \beta_{n-i+1} & 0 \end{bmatrix}\begin{bmatrix} \alpha_{n-i+2} & 1 \\ \beta_{n-i+2} & 0 \end{bmatrix}\cdots\begin{bmatrix} \alpha_n & 1 \\ \beta_n & 0 \end{bmatrix}\cdot\begin{bmatrix} 1 \\ 0 \end{bmatrix} = A_{n-i+1}A_{n-i+2}\cdots A_n \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ 1 \le i \le n-1 \tag{18}$$

The relationship (18) can be easily demonstrated by induction and using that:

$$\frac{\beta_{i-1}}{\alpha_{i-1} + \dfrac{r_i}{s_i}} = \frac{\beta_{i-1} * s_i}{r_i + \alpha_{i-1} * s_i} \tag{19}$$

For system given by (4) obtain the values in Table 2, the vectors $\begin{bmatrix} s_i \\ r_i \end{bmatrix} i = 2, \cdots, n$

Table 2. Evaluation reports r_i / s_i by suffix algorithm

r_i	2	3	4	5	6	7	8
s_i	-1	-2	-3	-4	-5	-6	-7

Using *Matlab sequence* below to determine the f_s vector for system given by (4), identical with the f vector, which argues the result of [1], that items can be calculated with a suffix product of matrices according relationship (15) and (18).

```
fs(1)=alfa(n);
for i=2:n
  x=[alfa(n) 1; beta(n) 0];
  for k=2:1:i-1
```

```
        y=[alfa(n-k+1) 1; beta(n-k+1) 0];
        x=y*x;
      end
   x=x*[1 0]';
   fs(i)=alfa(n-i+1)+x(2)/x(1);
  end
```

Assume that continued fraction (9) for $n = 4$ is:

$$f_4 = 1 + \cfrac{2}{3 + \cfrac{4}{5 + \cfrac{6}{7}}} = \alpha_1 + \cfrac{\beta_2}{\alpha_2 + \cfrac{\beta_3}{\alpha_3 + \cfrac{\beta_4}{\alpha_4}}} \tag{20}$$

Then $\alpha_1 = 1, \alpha_2 = 3, \alpha_3 = 5, \alpha_4 = 7$ and $\beta_2 = 2, \beta_3 = 4, \beta_4 = 6$. From the expressions given in (8) we deduce:

$$\begin{gathered} f_1 = 7; f_2 = 5 + \frac{6}{f_1} = 5 + \frac{6}{7} = \frac{41}{7} \\ f_3 = 3 + \frac{4}{f_2} = 5 + \frac{28}{41} = \frac{151}{41}; f_4 = 1 + \frac{2}{f_3} = 1 + \frac{82}{151} = \frac{233}{151} \end{gathered} \tag{21}$$

We have

$$A_2 = \begin{bmatrix} 3 & 1 \\ 2 & 0 \end{bmatrix}, A_3 = \begin{bmatrix} 5 & 1 \\ 4 & 0 \end{bmatrix}, A_4 = \begin{bmatrix} 7 & 1 \\ 6 & 0 \end{bmatrix}$$

Thus we get

$$A_4 = \begin{bmatrix} 7 & 1 \\ 6 & 0 \end{bmatrix}; A_3 A_4 = \begin{bmatrix} 41 & 5 \\ 28 & 4 \end{bmatrix}; A_2 A_3 A_4 = \begin{bmatrix} 151 & 19 \\ 82 & 10 \end{bmatrix} \tag{22}$$

From first of columns of matrices in (22) we obtain:

$$\begin{bmatrix} s_1 \\ r_1 \end{bmatrix} = \begin{bmatrix} 7 \\ 6 \end{bmatrix}, \begin{bmatrix} s_2 \\ r_2 \end{bmatrix} = \begin{bmatrix} 41 \\ 28 \end{bmatrix}, \begin{bmatrix} s_3 \\ r_3 \end{bmatrix} = \begin{bmatrix} 151 \\ 82 \end{bmatrix}$$

which are identical with fractions $6/7, 28/41$ and $82/151$ that appear in (21).

4. *LU* DECOMPOSITION ALGORITHM BY CONTINUED FRACTIONS AND EXPERIMENTAL RESULTS

Summarizing the steps involved in the calculation of the parallel LU decomposition of matrix A of system (1) through continued fractions calculation, we deduce:
Input: A trydiagonal matrix by n order from system (1)
Output: L and U matrices of the decomposition LU of A matrix
Step I. Let $f_1 := b_1$ and $\alpha_i = b_{n-i}$ for $1 \le i \le n$.

For $2 \le i \le n$ *computation* $\beta_i = -a_{n-i+1} * c_{n-i}$

Step II. *For* $2 \le i \le n$ *assign* $A_i := \begin{bmatrix} \alpha_i & 1 \\ \beta_i & 0 \end{bmatrix}$ and computation suffix products $A_i A_{i+1} \cdots A_n$.

Step III. *For* $2 \le i \le n$ *computation* $f_1 = \alpha_n ; f_i := \alpha_{n-i+1} + \dfrac{r_i}{s_i}$

where $[s_{n-i+1}, r_{n-i+1}]^T$ is first column of the product $A_i A_{i+1} \cdots A_n$ calculated in *Step* II.
Step IV. For $1 \le i \le n-1$ computation $e_i := a_i / f_{i-1}$

Theorem. *Parallel algorithm for LU decomposition by continued fractions*, *calculated LU factorization* of *tridiagonal matrix type A by* $n \times n$ *order in* $O(\log n)$ *parallel arithmetic operations using* $O\left(\dfrac{n}{\log n}\right)$ *processors*.

Proof. Any steps I, III and IV can be calculated with each $n-1$ in a time of order $O(1)$, or with
$O\left(\dfrac{n}{\log n}\right)$ processors in $O(\log n)$ time. All of the suffixes of the matrix products in Step III can be calculated using a convenient modification of the parallel prefix algorithm in $O(\log n)$ using $O\left(\dfrac{n}{\log n}\right)$ processors. Therefore the total time required LU parallel algorithm by continued fractions is $O(\log n)$ using $O\left(\dfrac{n}{\log n}\right)$ □

For different sizes of the linear system equations (1), implementation of this algorithm on a parallel network processors, leads us to the results in Table 3, where time is expressed in ticks and metrics are: $S = T_s / T_p$; $E = S / p$; $C = p * T_p$; $T_o = C - T_s$

Table 3. Evolution of several metrics to solve system (1) by LU decomposition

n	T_s	T_p	S Speed-up	E Efficiency	C Parallel cost	$T_o = C - T_s$ Overhead
8	735	605	1,21	20,17%	3630	2895
16	1465	1053	1,39	23,16%	6318	4853
32	2883	1839	1,57	26,17%	11034	8151
64	5693	3341	1,70	28,33%	20046	14353
128	11287	6239	1,81	30,17	37434	26147

Increase efficiency with increasing size of the problem is illustrated in Fig. 1 and the use of processors is given in Fig. 2.

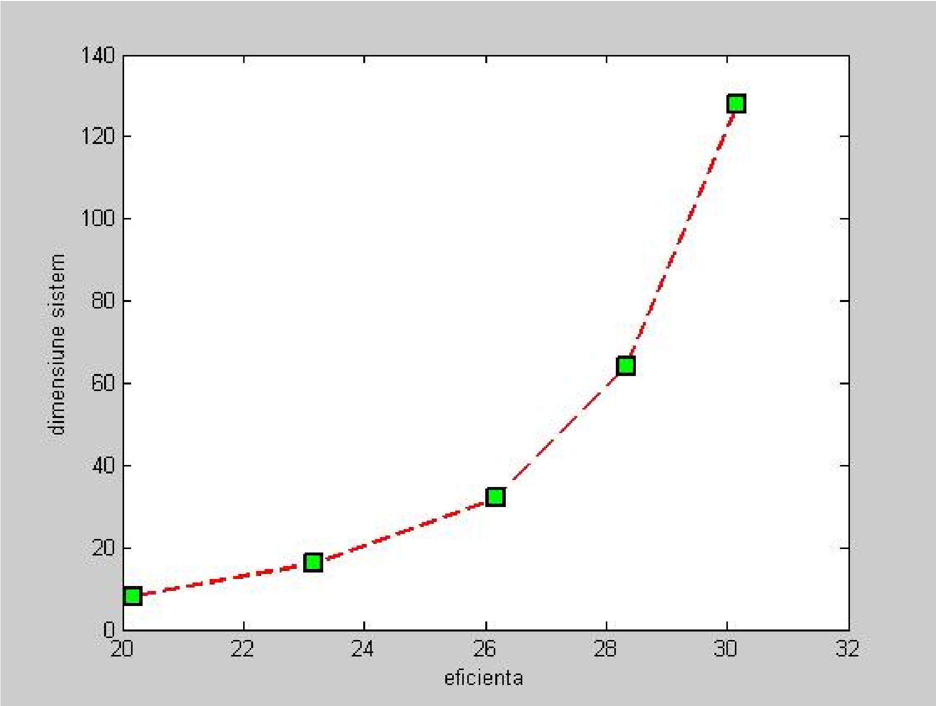

Fig. 1. Algorithm efficiency increases with the size problem.

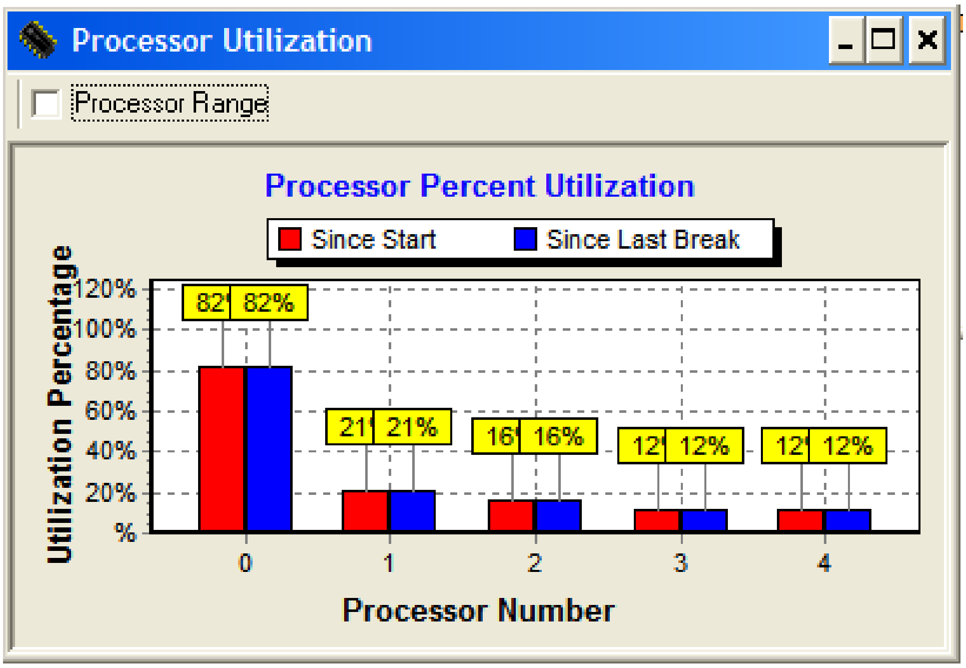

Fig. 2. How to use the processor to solve the system (1).

REFERENCES

[1] Milne-Thomson, L.M., *The Calculus of Finite Differences*, 2nd Unaltered Edition, Chelsea Publishing Company, New York, 1981.
[2] Croitoru, C., *Introducere în proiectarea algoritmilor paraleli*, Ed. MatrixRom, Bucureşti, 2002.
[3] Egecioglu, Ö., Koç, Ç.K., Laub, A., *Journal of Computational and Applied Mathematics*, **27**(1-2), 95, 1989.
[4] Smeureanu, I., Fanache, D., *Revista de Informatică Economică*, **3**(47), 115, 2008.
[5] Pacheco, S.P., *Parallel Programming with MPI*, Morgan Kaufmann Publishers, Inc, San Francisco, An Imprint of Elsevier, 1997.

Manuscript received: 12.09.2010
Accepted paper: 23.11.2010
Published online: 01.02.2011