Багато фізичних процеси і явища, з огляду на свою складність, не можуть бути описані аналітично. У таких випадках застосовують емпіричне моделювання. Для побудови емпіричних моделей оптимальної складності, яка має вигляд полінома заданого степеня, в роботі використаний метод, в основі якого лежить генетичний підхід. Реалізація розробленого методу вимагає багаторазового розв'язування системи лінійних алгебраїчних рівнянь. Розв'язування системи лінійних алгебраїчних рівнянь здійснюється шляхом приведення відповідної матриці до верхньої діагональної форми з одиницями на головній діагоналі. Аналіз алгоритму приведення матриці до верхнього діагонального вигляду показав, що така процедура володіє внутрішнім паралелізмом. На основі створеної моделі обчислювального процесу у вигляді мережі Петрі розроблено стратегію побудови паралельного алгоритму для розв'язування системи лінійних алгебраїчних рівнянь. Суть стратегії в тому, що обчислення здійснюються на декількох паралельних процесорах. Одному з них присвоєні координуючі функції, і він названий майстром. Інші процесори – робітники – знаходяться в підпорядкуванні майстра. Поділ обсягу обчислень такий, що кількість рядків матриці, з якими оперує майстер, більша не менше ніж на одиницю, за відповідну кількість рядків, відведених робітникові. Для запропонованої стратегії оцінена ефективність паралельного алгоритму за критерієм сумарної кількості арифметичних операцій. Запропонована стратегія є складовою частиною процесу синтезу емпіричної моделі оптимальної складності на основі генетичних алгоритмів. Поділ обчислювального навантаження між паралельно працюючими процесорами (майстром і робочими) забезпечує прискорення обчислювального процесу в п'ять і більше разів

Ключові слова: емпірична модель, генетичний алгоритм, паралелізм, мережа Петрі, число операцій

UDC 519.684.4
DOI: 10.15587/1729-4061.2019.171632

MODELING THE PARALLELISM OF EMPIRICAL MODELS OF OPTIMAL COMPLEXITY USING A PETRI NET

M. Gorbiychuk
Doctor of Technical Sciences, Professor*
E-mail: gorb@nung.edu.ua
O. Bila
Postgraduate student*
E-mail: ksm@nung.edu.ua
T. Humeniuk
PhD*
E-mail: tarasksm@gmail.com
*Department of computer systems and networks
Ivano-Frankivsk National Technical University of Oil and Gas
Karpatska str., 15,
Ivano-Frankivsk, Ukraine, 76019

1. Introduction

If there are comprehensive data about a certain system (object), it is possible to obtain a model that will adequately predetermine the behavior of such a system at assumptions and at specified parameters of the environment the system interacts with. Such models are called deterministic and are used to solve a variety of problems in different areas of human activity.

In most cases, systems (objects) are rather complicated, because there are many components that interact in complex ways. An example of such systems can be both technical and environmental objects [1, 2]. Empirical modeling is used in order to describe the interaction between the parameters of the system and the environment that directly influences the system and causes a change in its state. Mathematical models obtained as a result of empirical modeling are widely used for solving such problems as recognition of objects, prediction, automatic classification, and optimal control.

In paper [3], the method for synthesis of models of optimal complexity based on the principles of genetic algorithms was developed. Implementation of the method showed that determining the structure and parameters of the model requires using significant computational resources. That is why development and studying a parallel algorithm using a Petri net with the view to accelerating the computational process is a relevant scientific problem.

2. Literature review and problem statement

At present, to construct empirical models, one uses such methods as the classical least squares method (LSM), the Bayesian approach, the method of supporting vectors and the method of non-parametric nuclear estimation.

According to LSM, having the structure of a model, it is necessary, using observation of both input and output variables, to determine the parameters of the model by the best approximation method, in the capacity of which the sum of squares of deviations of experimental data from calculation data is used. The specified criterion is internal [4] and its application leads to a controversial result: the more complex the model, the more accurate it is. As a rule, an observed output magnitude of an object is overlapped by an obstacle, which distorts an actually existing functional dependence between the input and output of an object.

The authors of papers [4, 5] use an external addition, which is a certain part from the obtained sample of experimental data in order to select the best model from the

assigned class of models (as a rule, regressive). The method for selection of the best approximation by external addition was called the inductive method of self-organization of models.

Regarding the problem of synthesis of an empirical model, the use of an external criterion gives an opportunity to select unambiguously the structure of the model from the assigned class of models.

An implementation of inductive method of self-organization of models requires significant computational resources. That is why in order to reduce such consumption of resources, the method for selection of the structure of a model from the assigned class of models based on the ideas of genetic algorithms was proposed in paper [3]. In this case, it was possible to reduce considerably the consumption of computer time, although this consumption is still quite high.

Currently, the possibilities of increasing the effectiveness of computational processes due to increasing the clock frequency of processors are almost exhausted or economically inappropriate. One of the ways to solve the problem of increasing the effectiveness of the computational process is its parallelization [6, 7].

Development of parallel computations led to the emergence of a series of scientific works [7–9], where the parallel algorithms of solving the problems that require a significant amount of computer time were developed. The examples include the problems of digital filtering, assessment of the quality of functioning of complex dynamical systems, solution of systems of linear equations with rarefied matrices. Paper [10] gave the estimation of efficiency of parallel computational processes in solving the above tasks, based on the condition that the processors are loaded evenly. This situation occurs very rarely. Generally, the distribution of matrix blocks between processors is uneven. Thus, during the synthesis of empirical models of optimal complexity, the problem of development of the algorithm and studying its effectiveness for the case where computational powers are unevenly distributed between processors is unresolved.

3. The aim and objectives of the study

The aim of this study is to develop a parallel algorithm for the synthesis of empirical models of optimal complexity, which will make it possible to achieve the acceleration of the computational process compared to linear computations.

To accomplish the set aim, the following tasks were set:

– to algorithmize the problems of synthesis of empirical models of optimal complexity on the principles of genetic algorithms;

– to develop the strategy of interaction of parallel processors using Petri nets;

– to assess the effectiveness of the developed parallel algorithm for synthesis of empirical models of optimal complexity.

4. The empirical polynomial model of optimal complexity based on genetic algorithms

The method for construction of empirical models based on genetic algorithms, like the methods of group accounting of arguments (MGAA) is based on external criteria for regularity and shifting [5].

The use of the external criteria assumes that the sample of experimental data is split into two parts – learning N_A and checking. Learning set N_A is used for determining the parameters of the model, which in most cases has a polynomial structure:

$$y = \sum_{i=0}^{M-1} a_i \prod_{j=1}^{n} x_j^{s_{ji}}, \quad (1)$$

where n is the number of independent variables (input magnitudes), on which the value of output of research object depends; a_i, $i=\overline{0,M-1}$ is the coefficients of the model (1); s_{ji} – powers of arguments on which constraint is put:

$$\sum_{j=1}^{n} s_{ji} \le m, \forall i. \quad (2)$$

The number of terms in polynomial model (1) is calculated from formula [6]:

$$M = \frac{(m+n)!}{m!n!}. \quad (3)$$

The empirical model (1) that is optimal in relation to the structure is selected on test set N_B by regularity criterion [5]:

$$\Delta^2(B) = \frac{\sum_{i=1}^{N_B} (Y^{(i)}(B) - y^{(i)}(B))^2}{\sum_{i=1}^{N_B} Y^{(i)}(B)^2} \quad (4)$$

or shift

$$\Delta^2(A,B) = \frac{\sum_{i=1}^{N} \left(y^{(i)}(A) - y^{(i)}(B)\right)^2}{\sum_{i=1}^{N} \left(Y^{(i)}\right)^2}, \quad (5)$$

where $y^{(i)}(A)$ and $y^{(i)}(B)$, $i=\overline{1,N_A(N_B)}$ are the values of output of model (1), computed on sets N_A and N_B; $Y^{(i)}(B)$, $i=\overline{1,N_A}$ are the experimental values of the output magnitudes, referred to the set N_B; $Y^{(i)}$, $i=\overline{1,N}$ is the sample of data on observations of original magnitude Y.

In contrast to the algorithms of MGAA, where the basic functions in the form of regresses are included in the model (1) by the sort-out procedures, in the method of synthesis of empirical models of optimal complexity based on genetic algorithms (GA-method), the selection of the regressions of the model (1) is carried out using the mechanisms of natural selection based on the criteria of selection (4) or (5). This approach implies the creation of an orderly sequence of unities and zeros. If unity is in the i-th place ($a_i \neq 0$), the i-th regress is included in model (1). In the case when the i-th regression is discarded ($a_i=0$), zero will be in the i-th place. Such an orderly sequence, the elements of which are unities and zeros, in the theory of genetic algorithms is called a chromosome and its atomic element is gene. The set of chromosomes forms a population. Now the task of synthesis of empirical models of optimal complexity can be stated in terms of genetic algorithms. Sequentially choosing the «best» chromosomes from the entire population, it is necessary to find the one, for which the fitness function (4) or (5) will take minimum values. The chromosome chosen in this way will determine the structure of the empirical model, which will be optimal in the class of models (1).

Formally, the algorithm of the GA method can be described as follows:

$$GA = \langle I, CP, SA, SE, NP \rangle. \quad (6)$$

A tuple of operators (6) determines the sequence of their execution.

Operator I randomly generates the initial population with I individuals, each of which is a chromosome of length M.

Today, there are no theoretical prerequisites for choosing the number of chromosomes I in the population [11–13]. At a small number of chromosomes in the population, the algorithm quickly finishes its work and there is a danger that the fitness function reaches its minimum value. An increase in the number of chromosomes in a population causes an increase in computer time for implementation of the genetic algorithm. That is why value I is chosen following intuition of a researcher and based on the machine experiments.

Operator CP evaluates the adaptability of a chromosome in the population. For each chromosome, the criterion of selection (4) or (5) is calculated as follows: The original matrix F, the elements of which are regresses of coefficients a_i, $i=\overline{0,M-1}$, calculated on the set of values of the vector of input variables of an object, is divided into two separate matrices F_A and F_B of sizes $N_A \times M$ and $N_B \times M$, The i-th column is removed from matrices F_A and F_B, if there is zero on the i-th position of the chromosome; if there is unity, the corresponding column remains unchanged. As a result, we obtain matrices $\tilde{F}_A$ and $\tilde{F}_B$, from which c columns (according to the number of zeros in the chromosome) were removed.

The size of such matrices – $N_A \times (M-c)$ and $N_B \times (M-c)$. On the set of points N_A, non-zero coefficients a_{Aj}, $j=\overline{0,M-c-1}$ of model (1) are calculated by solving normal Gaussian equation:

$$\tilde{M}_{F,A}\bar{a}_A = \tilde{F}_A^{\mathrm{T}}\bar{Y}_A, \quad (7)$$

where $\bar{a}_A = (a_{A0}, a_{A1}, \ldots, a_{A,M-c-1})^{\mathrm{T}}$ is the vector of non-zero parameters of the model, which is associated with the next chromosome; $\tilde{M}_{F,A} = \tilde{F}_A^T \tilde{F}_A$ is the Fisher matrix; $\bar{Y}_A = (Y^{(1)}, Y^{(2)}, \ldots, Y^{(N_A)})$ is the vector of experimental values of the object output on set N_A.

Depending on what criterion of selection (4) or (5) will be used, by the derived coefficients a_{Aj}, $j=\overline{0,M-c-1}$ of the polynomial model (1) on the set of points N_A and N_B, the value is calculated:

$$\bar{y}(A) = \tilde{F}_A \bar{a}_A \quad (8)$$

or

$$\bar{y}(\mathbf{B}) = \tilde{F}_B \bar{a}_A. \quad (9)$$

Knowing $\bar{y}(A)$ or $\bar{y}(\mathbf{B})$, the value of fitness function $\Delta_j^2(A,B)$ or $\Delta_j^2(\mathbf{B})$, $j=\overline{1,I}$ is calculated for every chromosome from the original population from formula (4) or (5).

Operator *SA* checks the conditions of finishing the work of the algorithm of the GA-method. For each chromosome from the formed population I, selection criterion (4) or (5) is calculated. The selection of the best chromosome from population I, which will determine the structure of the empirical polynomial model of optimal complexity, is carried out according to the following rule. One determines:

$$\Delta_m^2(\mathbf{B}) = \min_j \Delta_j^2(\mathbf{B}) \quad (10)$$

or

$$\Delta_m^2(A,B) = \min_j \Delta_j^2(A,B), \quad j=\overline{1,I}. \quad (11)$$

If the minimum value (10) or (11) of the selection criteria (4) or (5) does not exceed an assigned value, computation stops. Computation can be stopped if, as a result of the algorithm implementation, there is no significant decrease in the fitness function or when the assigned number of iterations are performed.

After satisfaction of one of the three conditions, chromosome ch^*, for which the condition (10) or (11) is satisfied, is chosen from the next population. This chromosome assigns the structure of the model of optimal complexity and forms matrix F^* in such a way that the columns, which are associated with zero values of the corresponding genes of chromosome ch^*, are removed from the original matrix F. Recalculation of parameters of model (1) is carried out on the entire set of points using the LSM.

When none of the enumerated three conditions is met, the execution of the next operator SE takes place.

Operator SE carries out the selection of chromosomes. By calculated values of fitness function, operator *CP* performs selection from population I of those chromosomes that will participate in the creation of descendants for a new population. Such selection is carried out according to the principle of natural selection, where the chromosomes with the best values of fitness function (4) or (5) have the best chances for creation of a new population. The most common method for selection is the roulette method and the tournament method [11]. The roulette method can be used when the fitness function is positive, which makes it suitable only for maximization problems. The tournament method can be used both in maximization problems and in minimization problems.

During tournament selection, all chromosomes are divided into sub-groups of z_0 individual each. The best chromosome according to the selection criterion (4) or (5) is chosen from each subgroup. As a result, we obtain a new population of chromosomes, which forms the parent pool $I(k)$. The number of individuals I in the population remains unchanged. Subgroups can have an arbitrary size, but most often the population is divided into subgroups of 2–3 individuals each.

The number of chromosomes, which is generated by operator I must be multiple of the number of individuals z_0 in the subgroups. This algorithm used the tournament method.

Operator NP generates new population of descendants from the chromosomes selected by operator *SE*. A new population of chromosomes is generated using the process of crossover and mutation. The probability of crossover P_h is selected from the interval [0.5; 1] and the probability of mutation is within [0; 0.1].

Each chromosome selected by operator *SE*, which belongs to the parent pool, is subject to the crossover process. To do this, a pair of chromosomes is selected randomly from the population of individuals. Random number P_c is generated from the interval [0.5; 1] and if its value does not exceed P_h, the pair of chromosomes is subject to crossover. Otherwise, a pair of chromosomes remains unchanged. If there is crossover of chromosomes, the position of a gene (locus), which determines the point of crossover, is played for each pair. The chromosome of each parent has M genes and the point of crossover L_c is a natural number smaller than M. That is why the fixation of the point of crossover is reduced to a random choice of integer L_c from the interval [1, M–1].

The crossover process results in creation of a new pair of chromosomes from a pair of parents. The first descendant in the pair of chromosomes on positions from 1 to L_c consists of genes of the first parent, and on positions L_c+1 to M – of the genes of the other parent. The second descendant in the pair of chromosomes on positions from 1 to L_c consists of the genes of the second parent, and on positions L_c+1 to M – from the genes of the first parent.

The probability of mutation P_m can be assigned by selection of a random number from the interval [0; 1] for each gene. The genes, for which the played number is less than or equal to P_m, are subject to mutation (by replacing unity with zero and vice versa). Mutation can be carried out both over a pool of parents and the pool of descendants.

After execution of operator *NP*, there is transition to operator *CP*.

Implementation of the algorithm of the GA-method [2] showed a significant gain in time compared to the sort-out MGAA algorithm, although at increased dimensionality of a problem, computer time consumption to implement it increases.

5. Studying the algorithm of synthesis of the empirical model of optimal complexity using a Petri net

Analysis of the algorithm of GA-method showed [14] that the parallel form, which, along with other operations, includes the operation of solution of normal Gauss equation (7), corresponds to the value of fitness criterion (4) or (5), which is calculated for each chromosome. The dimensionality of the latter is determined by power of polynomial (1) and by the number of input variables.

Thus, in paper [15], the empirical model (1), in which $n=7$ and $m=4$, was synthesized In accordance with formula (3), the maximum number of model parameters (1) M=330. Such a large number of parameters lead to the fact that the solution of equation (7) (especially at the final stages of the algorithm operation) will require additional consumption of computer time.

It is possible to reduce consumption of time to implement the algorithm of the GA-method if we parallelize additionally the Gaussian algorithm of the solution of the system of linear algebraic equations (7) in each parallel form.

Normal equation (7) will be written down in the following form:

$$\tilde{A}\bar{a}_A = \tilde{b}, \tag{12}$$

where $\tilde{A} = \tilde{M}_{F,A}$; $\tilde{b} = \tilde{F}_A^T \bar{Y}_A$.

Equations (12) will be solved using the method of Gaussian exception, reducing extended matrix $A_e = [\tilde{A} \,\vdots\, \tilde{b}]$ to the upper diagonal form according to the following formulas [16]:

$$\tilde{a}_{i\bullet}^{(i)} = \frac{\tilde{a}_{i\bullet}^{(i-1)}}{\tilde{a}_{ii}^{(i-1)}}, \quad i = \overline{1,\, M-c-1}; \tag{13}$$

$$\tilde{a}_{k\bullet}^{(i)} = \tilde{a}_{k\bullet}^{(i-1)} - \tilde{a}_{i\bullet}^{(i)}\tilde{a}_{ki}^{(i-1)}, \quad k = \overline{i+1, M-c}, \tag{14}$$

where $\tilde{a}_{i\bullet}^{(i)}$, $\tilde{a}_{k\bullet}^{(i)}$ are the i-th and k-th rows of extended matrix A_e; $\tilde{a}_{i\bullet}^{(0)} = \tilde{a}_{i\bullet}$, $i = \overline{1, M-c-1}$; $\tilde{a}_{k\bullet}^{(0)} = \tilde{a}_{k\bullet}$, $k = \overline{i+1, M-c}$. In formula (13) $\tilde{a}_{ii}^{(i-1)}$ is called the leading element.

In the computational process, the cycle by index k is internal in relation to the external cycle by index i. The result of applying iterative procedures (13) and (14) is the upper diagonal matrix with unities on the main diagonal:

$$U = \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1,M-c} & u_{1,M-c+1} \\ 0 & 1 & u_{23} & \cdots & u_{2,M-c} & u_{2,M-c+1} \\ 0 & 0 & 1 & \cdots & u_{3,M-c} & u_{3,M-c+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 & u_{M-c,M-c+1} \end{bmatrix},$$

where $u_{ij} = \tilde{a}_{ij}^{(i)}$, $i = \overline{1,\, M-c}$, $j = \overline{i+1,\, M-c+1}$.

Thus, equation (12) is converted into the equivalent system of linear algebraic equations:

$$\tilde{U}\,\bar{a}_A = \bar{b}_A, \tag{15}$$

where $\tilde{U}$ is the square matrix of dimensions of M–c, formed from the matrix by withdrawal of the last column; $b_{A,i}$= $= u_{i,M-c+1}$, $i = \overline{1, M-c}$.

Equation (12) is solved by the method of reverse run, starting with the last equation. It is clear that:

$$a_{A,M-c} = b_{A,M-c}. \tag{16}$$

Other values of a_A are calculated by the iterative procedure:

$$a_{A,i} = b_{A,i} - \sum_{j=i+1}^{M-c} u_{ij} a_{A,j}, \quad i = \overline{M-c-1,\, 1}. \tag{17}$$

Unlike the classical Gaussian method [17], in the process of reducing the system of equations (12) to the form of (15), there is no operation of division by u_{ii} to calculate a_A. As it is known, execution of the division operation takes more computer time compared to other algebraic operations. So, calculation of parameters of model (4) by recurrent ratios (16) and (17) gives tangible savings of computer time compared with the classical Gauss method.

All the above is also true for the *LU*-method, according to which matrix $\tilde{A}$ is represented as the product of two matrices L and U_R. The former is the lower diagonal with unities on the main diagonal, and the latter is the upper diagonal matrix. Then triangular system $L\bar{\gamma} = \bar{b}_A$, $U_R\bar{a}_A = \bar{\gamma}$ are solved by direct and reverse run.

Because, in the process of solving equations (12), most computer time is consumed for reducing matrix A_e to upper diagonal form, it is appropriate to parallelize the corresponding algorithm in order to reduce these costs.

Let us assume that there are q parallel processes. One of them, called the master, performs the function of control of a parallel computing process. Other q–1 processes (slaves) are subordinated to the master.

The essence of the algorithm of parallelization is that at every computation step, the matrix, which is in the workspace of the master, is split into q sub-matrices with further computation of the values of their elements from formulas (13) and (14). As a result, sub-matrix A, which is located in workspace of the master, is reduced to the upper diagonal form with unities on the main diagonal. At the same time with the master, the slaves recalculate the elements of their sub-matrices from formula (14), and after finishing another cycle of computations forward them to their master. The master unites the sub-matrices received from the slaves into a single matrix that will be called a joint matrix and denoted as $A_U^{(i)}$, $i = 1,2,\ldots$

Let us choose the procedure of dividing the joint matrix $A_U^{(i)}$, which is in workspace of the master, into separate sub-matrices. Let us assume that at each stage of calculation, the rows of joint matrix $A_U^{(i)}$, $i=1,2,\ldots$ are divided into number q without residue.

Before the first phase of computation, we will divide matrix $A_U^{(1)}=A_e$ into q equal parts. Then all q sub-matrices will have the same number of rows:

$$\tilde{n}_1 = z/q, \tag{18}$$

where $z=A_e$.

After the end of the first stage of computation, the number of rows of the joint matrix $A_U^{(1)}$ got decreased by magnitude $\tilde{n}_1$. That is why joint $A_U^{(2)}$ with the following number of rows: $z_1=z-\tilde{n}_1$. will be in the workspace of the master.

Taking into account value $\tilde{n}_1$, which is determined from formula (18), we will get:

$$z_1 = z - \frac{z}{q} = z\frac{q-1}{q}.$$

At the beginning of the computation stage 2, the master divides joint matrix $A_U^{(2)}$ again into q equal parts, which leads to the following result: $\tilde{n}_2=z_1/q$.

Considering value z_1, we will obtain:

$$\tilde{n}_2 = z\frac{q-1}{q^2}. \tag{19}$$

The result of finishing computation cycle 2 will be joint matrix $A_U^{(2)}$ with z_2 rows $z_2=z-\tilde{n}_1-\tilde{n}_2$.

With regard to values $\tilde{n}_1$ and $\tilde{n}_2$, which were determined from formulas (18) and (19), we will obtain:

$$z_2 = z\frac{(q-1)^2}{q^2}.$$

At the beginning of computation stage 3, the master again divides joint matrix $A_U^{(3)}$ with number of rows z_2 into q equal parts. As a result of such division in workspace of the master and slaves, there will be sub-matrices with the number of rows:

$$\tilde{n}_3 = z\frac{(q-1)^2}{q^3}. \tag{20}$$

After completion of computation stage 3, we will obtain matrix $A_U^{(4)}$ with the number of rows: $z_4=z-\tilde{n}_1-\tilde{n}_2-\tilde{n}_3$. Taking into account $\tilde{n}_1$, $\tilde{n}_2$ and $\tilde{n}_3$, we will obtain:

$$z_4 = \frac{z}{q^3}(q-1)^3.$$

If we divide matrix $A_U^{(4)}$ into q equal parts,

$$\tilde{n}_4 = \frac{z}{q^4}(q-1)^3. \tag{21}$$

Continuing the process of dividing the joint matrix after finishing the r-stage of calculations, we will obtain matrix $A_U^{(r)}$ with the following number of rows:

$$z_r = \frac{z}{q^r}(q-1)^r.$$

To continue the operation of the algorithm on the $r+1$ cycle, the master divides matrix $A_U^{(r)}$ into q equal parts, so that:

$$\tilde{n}_r = \frac{z}{q^r}(q-1)^{r-1}, \quad r=\overline{1,\,N_z-1}, \tag{22}$$

where N_z the total number of computation stages.

In general case, the values of $\tilde{n}_r$, $r=\overline{1,N_z-1}$ are non-natural numbers.

For each of $q-1$ workplaces, the capacity of each part (the number of rows) of the joint matrix $A_U^{(r)}$, $r=\overline{1,\,N_z-1}$ will be determined as a integer part of number $\tilde{n}_r$, $r=\overline{1,N_z-1}$. So,

$$n_{w,r} = \left[\frac{z}{q^r}(q-1)^{r-1}\right], \quad r=\overline{1,\,N_z-1},$$

where $[\cdot]$ is the integer part of the number.

The number of rows $n_{f,r}$ of the sub-matrix which is in workspace of the master, will be calculated as the difference between the number of rows z_r, $r=\overline{1,N_z-1}$ of joint matrix $A_U^{(r)}$ and the number of rows, which were totally delegated by $q-1$ slave:

$$n_{f,r} = z_r - n_{w,r}(q-1), \quad r=\overline{1,\,N_z-1}. \tag{23}$$

If in the process of such division of joint matrices $A_U^{(r)}$ it will appear that $n_{f,r}<n_{w,r}$, then $n_{w,r}$ will gradually decrease by unity until the condition is satisfied:

$$n_{f,r} \geq n_{w,r}. \tag{24}$$

In this case, after each such decrease, the value of $n_{f,r}$ is calculated from formula (23).

Bearing in mind the requirement (24), the number of rows in the sub-matrix that is in the workspace of each slave will be calculated from formula:

$$n_{w,r} = \left[\frac{z}{q^r}(q-1)^{r-1}\right] - v_r, \quad r=\overline{1,N_z-1}, \tag{25}$$

where v_r is the number of unities, by which magnitude $n_{w,r}$ decreased and other condition was met (24).

As a result of the chosen strategy of splitting joint matrices, the sub-matrices with the number of rows $n_{f,r}$, will be located in the workspace of the master, and each slave will get the sub-matrix with the number of rows $n_{w,r}$. The values of $n_{f,r}$ and $n_{w,r}$ are calculated from formulas (23) and (25) on condition that inequality (24) is satisfied.

Division of the joint matrices by the master continues until the size of the next layer that should be sent will not be larger than the number of processes:

$$z_s \leq q. \tag{26}$$

Then $n_{f,s}=z_s$, and in the workspace of each slave $n_{w,s}=0$.

Condition (26) determines the end of the process of reducing the original matrix A_e to the upper diagonal form with unities on the main diagonal.

After satisfying condition (26), the master reduces the remaining sub-matrix to the upper rectangular form with unities on the main diagonal. The final step is to unite all sub-matrices that were saved in the workspace of the master.

Let us evaluate the number of computational stages N_z, needed to implement the algorithm of reducing matrix A_e to the upper diagonal form with unities on the main diagonal.

Let us assume that at the last stage of run of the computational algorithm, condition (26) will be the following:

$$n_{f,s} = q. \tag{27}$$

Capacity z_r of joint matrix $A_U^{(r)}$ at each stage of computation decreases by magnitude $n_{f,j}$. That is why at the r-th computation stage we will obtain:

$$z_r = z - n_{f1} - n_{f2} - \cdots - n_{f,r} = z - \sum_{j=1}^{r} n_{f,j}. \quad (28)$$

After completing the N_z-1 stage of computations, capacity of joint matrix $A_U^{(N_z)}$ will be $n_{f,s}$. Based on condition (27) and formula (28), we obtain:

$$n_{f,s} = z - \sum_{j=1}^{N_z-1} n_{f,j}.$$

Since there is assumption (27), then:

$$z - \sum_{j=1}^{N_z-1} n_{f,j} = q. \quad (29)$$

In accordance with the chosen strategy of splitting the joint matrix into q sub-matrices, between powers of sub-matrices of the master and each of the slaves, ratio (25) is true, based on which it can be argued that:

$$n_{f,j} - n_{w,j} = R_j, \quad j = \overline{1, N_z - 1}, \quad (30)$$

where $0 \le R_j \le q$ is an integer.

Powers of sub-matrices that are in the workspace of slaves will be represented as a whole part of number $\frac{z}{q^j}(q-1)^{j-1}$, taking into consideration that condition (25) should be satisfied:

$$n_{w,j} = \frac{z}{q^j}(q-1)^{j-1} - r_j - v_j, \quad (31)$$

where $0 \le r_j < 1$ is the discarded fractional part of number $\tilde{n}_j$.

In follows from formulas (30) and (31) that:

$$n_{f,j} = \frac{z}{q^j}(q-1)^{j-1} + \delta_j, \quad (32)$$

where $\delta_j = R_j - (r_j + v_j)$, $j = \overline{1, N_z - 1}$.

Substituting the obtained value $n_{f,j}$ in formula (29), we will obtain:

$$z\left(1 - \sum_{j=1}^{N_z-1} \frac{1}{q^j}(q-1)^{j-1}\right) - \Delta = q,$$

where $\Delta = \sum_{j=1}^{N_z-1} \delta_j$.

Expression in the brackets will be represented in the following form:

$$1 - \frac{1}{q}\sum_{j=1}^{N_z-1}\left(\frac{q-1}{q}\right)^{j-1}.$$

Magnitude $\sum_{j=1}^{N_z-1}\left(\frac{q-1}{q}\right)^{j-1}$ is the sum of geometric progression, in which the first term is unity, and denominator is $(q-1)/q$; the number of terms of geometric progression is N_z-1. After calculating the sum of the geometric progression, we obtain:

$$1 - \frac{1}{q}\sum_{j=1}^{N_z-1}\left(\frac{q-1}{q}\right)^{j-1} = \left(\frac{q-1}{q}\right)^{N_z-1}.$$

The calculated value of the sum allows us to write down formula (32) as follows:

$$z\left(\frac{q-1}{q}\right)^{N_z-1} - \Delta = q. \quad (33)$$

From ratio (33), we will determine the number of stages to complete in order to reduce matrix A_e to the upper diagonal form:

$$N_z = 1 + \ln\left(\frac{q+\Delta}{z}\right) \Big/ \ln\left(\frac{q-1}{q}\right). \quad (34)$$

Value of N_z also takes into account the last stage, which will take place after satisfaction of condition (26). At the final stage, the master reduces the last part of matrix A_e to the upper diagonal form, and slaves have already finished their work.

Formula (34) determines the number of stages required to complete the work of the algorithm with a certain error. The error occurs because the work of the algorithm ends when at the last computation stage, the matrix of capacity q will be united in the workspace of the master. In fact, the algorithm stops running after condition (26) is satisfied.

Fig. 1 shows the result of determining the number of calculation stages necessary to reduce matrix A_e to the upper diagonal form. The following input data were used: dimensions of square matrices $A_e(i)$, $i = \overline{1,8}$ were formed as a vector –

$$\bar{z} = [10000\ 8000\ 6000\ 4000\ 2000\ 1000\ 500\ 300];$$

the number of processors $q=6$. For each matrix $A_e(i)$, $i = \overline{1,8}$. The number of stages was calculated from formula (34) and the number of stages as a result of satisfaction of condition (26) was recorded.

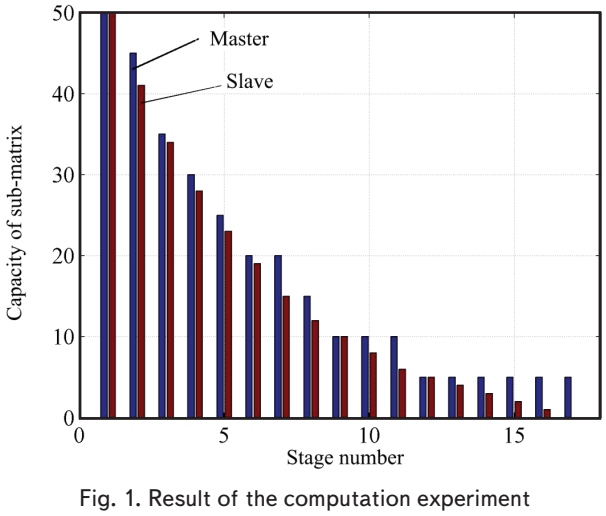

Fig. 1. Result of the computation experiment

Fig. 2 helps visualize the distribution of capacities of sub-matrices of the master and of the slaves when completing the i, $(i = \overline{1,8})$-th computation stage. It shows that condition (26) was always satisfied, and the following original data were used in this case: $z=300$; $q=6$.

To assess the effectiveness of the computational process, first and foremost, we need the working model that adequately characterize the structural and dynamic properties

of the process and reflect the parallelism of the algorithm of reducing matrix A_e to the upper diagonal form.

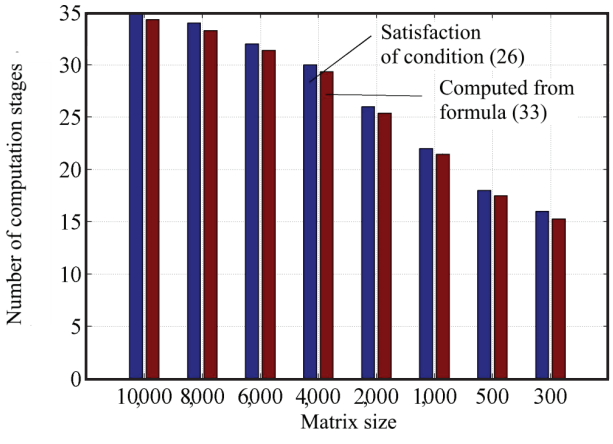

Fig. 2. Distribution of capacities of sub-matrices of the master and of slaves by calculation stages

The Petri nets [18, 19], which adequately reflect the interaction of the basic parts of the process and the information exchange between them, became widely used in modeling parallel computational processes.

Fig. 3 shows the Petri net, which is the model of the computational process of reducing matrix A_e to the upper diagonal form.

Position p_s contains the marker that activates transition t_s, causing the completion of calculation stage 1. Position $p_A^{(0)}$ is identified as the process of sending sub-matrices $A_0^{(i)}$, $i=\overline{1, q-1}$ by the master to the slaves. Sub-matrices $A_0^{(i)}$, $i=\overline{1, q-1}$ have identical sizes and the number of their rows is determined from formula (25). Sub-matrix $A_0^{(0)}$ of the size of $n_{f1}\times(M-c+1)$ remains in the workspace of the master, where the number of rows of the sub-matrix is calculated from formula (23).

Subsequently, computations will be performed like at stage 1. After the k-th step of calculations, sub-matrix $A_k^{(0)}$ with the number of rows, calculated from formula (23), will be in the workspace of the master, and sub-matrices $A_k^{(i)}$, $i=\overline{1, q-1}$ will have the size $n_{w1}\times(M-c+1)$, in which the number of zero columns $P_{w,k}=\sum_{j=1}^{k} n_{f,k}$. After calculation by the master of the components of vector $\tilde{a}_{k\bullet}$ and sending out their values to slaves, transitions t_i^k, $i=\overline{0, q-1}$ are activated. Transformation of sub-matrices $A_k^{(i)}$, $i=\overline{1, q-1}$ on the k-th step of calculations is modeled by positions $p_k^{(i)}$, $i=\overline{1, q-1}$.

Completion of computation stage 1 occurs on condition that the master reduced his matrix $A_m^{(0)}$ to the upper diagonal form. Then matrices $A_m^{(i)}$ are united into joint matrix:

$$A_U^{(1)}=\bigcup_{i=1}^{q-1} A_m^{(i)},$$

which is mapped by position p_{sum}.

Positions p_D, p_c and p_f generate the conditions to complete the work of the algorithm of reducing matrix A_e to the upper diagonal form

Position p_e has a constant marker. Transition t_{sz} will work in the case when the number of rows of sub-matrix $A_m^{(s)}$, which was reduced to the upper diagonal form (a maker will appear in position p_{sz}), will not exceed the value of q. After that, all sub-matrices that are in workspace of the master are joint to matrix U (position p_U) and the process of reducing the matrix to the upper diagonal form is completed.

In case condition (26) is met, transition t_F is activated. A marker appears in position p_c, which leads to the start of transition t_{sum} and, as a result, the marker transfers to position p_s.

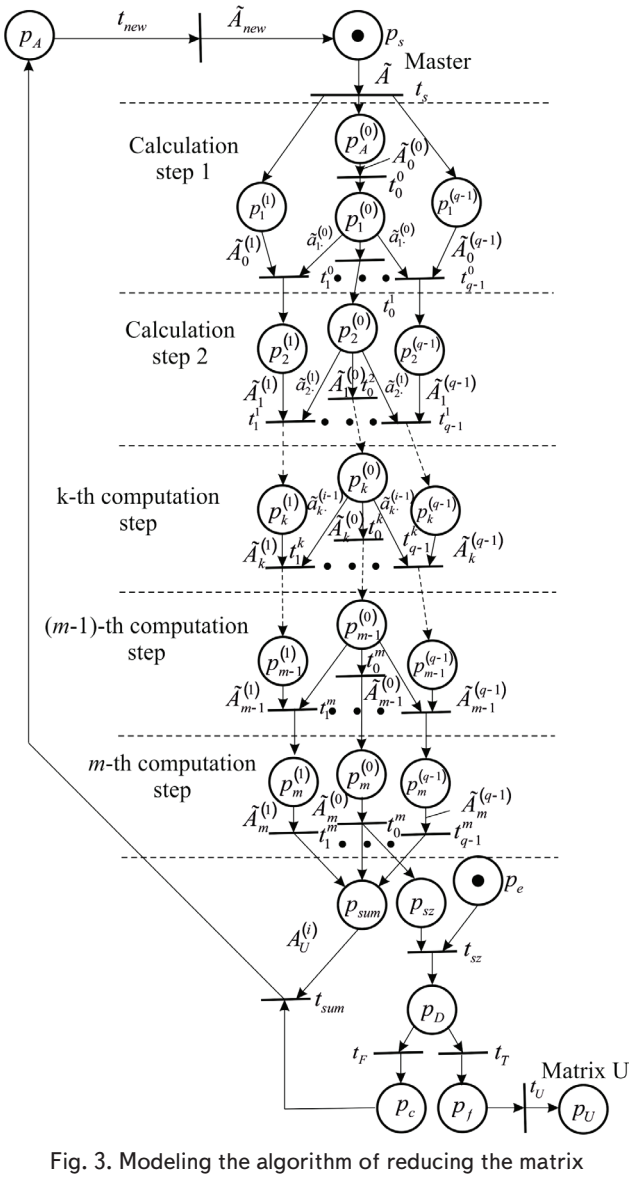

Fig. 3. Modeling the algorithm of reducing the matrix to the upper diagonal form using a Petri net

In the process of reducing the original matrix A_e to the upper diagonal form, the master and slaves work in parallel, with the ratio of (25). The time consumed to solve the original problem will be determined by the number of arithmetic operations, performed by the master and slaves, both at each stage of computation, and on the whole until the problem is solved.

For the case where distribution of computational load between the master and the slaves ae arbitrary, the problem of finding the number of arithmetic operations at each step of the algorithm is solved in [20].

At first, we will calculate the number of operations at each step of the computation executed by the master, assuming the number of rows of the sub-matrix in the workspace of the master are calculated from formula (23).

Execution of the operation of normalizing the first raw (Fig. 3) of sub-matrix $\tilde{A}_0^{(0)}$ according to formula (13) by the master demands $z=M$ with division operations (diagonal elements of sub-matrix $\tilde{A}_0^{(0)}$ are assigned the value of unity. $(n_{f1}-1)z$ operations of multiplication and $(n_{f1}-1)z$ operations of subtraction will be executed to enumerate the elements in the rest n_{f1} rows of sub-matrix $\tilde{A}_0^{(0)}$ from formula (14). The total number of operations performed by the master at stage 1 made up $N_{f1}=(2n_{f1}-1)z$.

As a result of completion of computation step 1, original sub-matrix $\tilde{A}_0^{(0)}$ is modified to sub-matrix $\tilde{A}_1^{(0)}$, in which the first column contains zero elements, except for the first one, which is equal to unity. Sub-matrices $\tilde{A}_0^{(i)}$, $i=\overline{1,q-1}$, that are in workspaces of slaves will be transformed to sub-matrices $\tilde{A}_1^{(i)}$, $i=\overline{1,q-1}$ with zero columns.

At computation step 2 (Fig. 3), the master normalizes the second row of sub-matrix $\tilde{A}_1^{(0)}$, after executing $z-1$ operations of division. Other rows of matrix $\tilde{A}_1^{(0)}$ are recalculated from formula (14). In this case, $(n_{f1}-2)(z-1)$ operations of multiplication will be performed and the same number of operations of subtraction. So, at calculation stage 2, the master will execute $N_{f2}=2(n_{f1}-2)(z-1)+(z-1)=2(n_{f1}-3)(z-1)$ arithmetic operations. In this case, modified matrix $\tilde{A}_2^{(0)}$ will be obtained (Fig. 3).

At computation stage 3, the master norms the third row of matrix $\tilde{A}_2^{(0)}$ (Fig. 3), having executed $z-2$ operations of division. Starting from the fourth row, the master enumerates all the elements of sub-matrix $\tilde{A}_2^{(0)}$ by formula (14), spending the following number of operations of division, multiplication and subtraction: $N_{f3}=3(n_{w1}-5)(z-2)$. As a result, modified matrix $\tilde{A}_3^{(0)}$ will be in the workspace of the master (Fig. 3).

Generalizing the obtained results, it is possible to argue that at the k-th computation step, the master will execute the following number of operations of multiplication, division, and subtraction.

$$N_{f1,k}=2((n_{f1}-k)+0.5)(z-k+1). \qquad (35)$$

Stage 1 of parallel computations is over when condition $k=n_{f1}$, is satisfied, i. e., in formula (35) $k=\overline{1,n_{f1}}$.

Thus, after computation stage 1, we will obtain the upper rectangular matrix $\tilde{A}_{n_{f1}}^{(0)}$ with unities on the main diagonal in the workspace of the master. The size of such matrix is $n_{f1}\times(z+1)$.

After uniting $q-1$ sub-matrices by the master, sub-matrices $\tilde{A}_{n_{f1}}^{(0)}$ and united matrix $A_U^{(1)}$ (Fig. 3), in which there are n_{f1} zero columns, will be stored in its memory. The size of the latter is $(z-n_{f1})\times(z-n_{f1}+1)$.

The master splits the united matrix $A_U^{(1)}$ into q layers according to formulas (23) and (25). Sub-matrices $\tilde{A}_{\Sigma,1}^{(0)}$ with n_{f2} rows will be in the workspace of the master, and slaves will obtain sub-matrices $\tilde{A}_{\Sigma,1}^{(1)}$, each of which will have n_{w2} rows that are calculated from formula (25) at $r=2$.

The number of operations of division, multiplication, and subtraction, which the master executes at step 2 of the algorithm run will be calculated using formula (35), given that the number of nonzero columns of the sub-matrix, located in the workspace of the master is equal to $z-n_{f1}+1$. This means that in formula (35), it is necessary to substitute z with $z-n_{f1}$, and n_{f1} with n_{f2}. As a result of such substitution, we will obtain:

$$N_{f1,k}=2((n_{f2}-k)+0.5)(z-n_{f1}-k+1).$$

The second stage of calculations will finish when condition $k=n_{f2}$.is satisfied.

After completion of calculation stage 2, rectangular matrix $A_{\Sigma,2}^{(U)}=A_{n_{f1}}^{(0)}\cup A_{\Sigma,1}^{(0)}$ of size $(n_{f1}+n_{f2})(z+1)$, on the main diagonal of which there will be unities, will be stored in memory of the master.

At the beginning of calculation cycle 3, the master unites the $q-1$ sub-matrix in one matrix $A_U^{(2)}$ of the size of $(z-n_{f1}-n_{f2})\times(z-n_{f1}-n_{f2}+1)$, which does not contain zero elements. The master splits the obtained matrix into q sub-matrices. The first sub-matrix $A_{\Sigma,3}^{(0)}$, which has n_{f3} rows, remains in the workspace of the master and other sub-matrices $A_{\Sigma,3}^{(1)}$ with n_{w3} rows are sent to $q-1$ slaves.

By analogy with stage 2, the number of operations of division, multiplication and subtraction executed by the master at stage 3 will be calculated from formula (35), after substitution of z with $z-n_{f1}-n_{f2}$. In this case, it is necessary to take into account that capacity of sub-matrix $A_{\Sigma,3}^{(0)}$ makes up n_{f3} rows. Thus,

$$N_{f3,k}=2((n_{f3}-k)+0.5)(z-n_{f1}-n_{f2}-k+1). \qquad (36)$$

Stage 3 is completed on condition that $k=n_{f3}$.

Let us assume that r stages of calculation were completed. As a result, a rectangular matrix $A_{\Sigma,r}^{(0)}$ of size $\left(\sum_{j=1}^{r-1}n_{fj}\right)\times(z+1)$, with unities on the main diagonal will be stored in the memory of the master. In the workspace of the master there will be joint matrix $A_U^{(r)}$ of the size of $\left(z-\sum_{j=1}^{r-1}n_{fj}\right)\times\left(z-\sum_{j=1}^{r-1}n_{fj}+1\right)$ with non-zero elements, which the master splits into q layers. As a result, we will obtain sub-matrix $A_{\Sigma,r}^{(0)}$ of capacity of $n_{f,r}$, which is found in the working space of the master. Each of the sub-matrices of the slaves $A_{\Sigma,r}^{(1)}$ has capacity $n_{w,r}$.

Based on the results of formula (36), we can assert that at the k-th step of the r-th stage of calculations the master will perform the following number of operations of multiplication, division, and subtraction:

$$N_{fr,k}=2\left(\left(n_{f,r}-k\right)+0.5\right)\left(z-\sum_{j=1}^{r-1}n_{f,j}-k+1\right). \qquad (37)$$

The finish of the r-th stage is meeting the condition $k=n_{f,r}$.

If in formula (37) we take into account value n_{fj} from formula (32), we obtain:

$$N_{fr,k}=2\left(\left(n_{f,r}-k\right)+0.5\right)\times$$
$$\times\left(z\left(1-\frac{1}{q}\sum_{j=1}^{r-1}\left(\frac{q-1}{q}\right)^{j-1}\right)-\Delta_{r-1}-k+1\right),$$

where $\Delta_{r-1}=\sum_{j=1}^{r-1}\delta_j$.

After calculating $\sum_{j=1}^{r-1}\left(\frac{q-1}{q}\right)^{j-1}$ from formula of geometric progression, we derive:

$$N_{fr,k}=2\left(\left(n_{f,r}-k\right)+0.5\right)\left(z\left(\frac{q-1}{q}\right)^{r-1}-\Delta_{r-1}-k+1\right). \qquad (38)$$

Now let us calculate the number of division, multiplication, and subtraction operations executed by the master during the r-stage of calculations:

$$N_{M,r}=\sum_{k=1}^{n_{f,r}}N_{fr,k},\quad r=\overline{1,N_z-1}. \tag{39}$$

If we consider operations on sending $\tilde{a}_{ij}^{(v)}$ element at each k-th computation step, the maximum number of parallel arithmetic operations, which rut the r-th stage will be:

$$N_r=N_{M,r}+n_{f,r},\quad r=\overline{1,N_z-1}. \tag{40}$$

Let us calculate the value $N_{M,r}$. If we consider formula (38), expression (39) will take the following form:

$$N_{M,r}=2\sum_{k=1}^{n_{f,r}}\left(\left(n_{f,r}-k\right)+0.5\right)\left(z\left(\frac{q-1}{q}\right)^{r-1}-\Delta_{r-1}-k+1\right).$$

Taking into account [21], that:

$$\sum_{k=1}^{n_{f,r}}k=\frac{n_{f,r}}{2}(n_{f,r}+1)\ \text{ and }\ \sum_{k=1}^{n_{f,r}}k^2=\frac{n_{f,r}}{6}(n_{f,r}+1)(2n_{f,r}+1),$$

we will obtain:

$$N_{M,r}=2n_{f,r}\left(\begin{array}{l}\left(n_{f,r}+0.5\right)\left(z\alpha^{r-1}-\Delta_{r-1}-\frac{1}{2}\left(n_{f,r}-1\right)\right)-\\-\left(n_{f,r}+1\right)\left(\frac{1}{2}\left(z\alpha^{r-1}-\Delta_{r-1}\right)-\frac{1}{3}\left(n_{f,r}-1\right)\right)\end{array}\right), \tag{41}$$

where $\alpha=(q-1)/q$.

To calculate the total number of arithmetic operations executed by the master, it is necessary to take the sum by all operations for each of r computation stages:

$$N_{sum}=\sum_{r=1}^{N_z-1}N_{M,r}+\sum_{r=1}^{N_z-1}n_{f,r}.$$

It is clear that:

$$\sum_{r=1}^{N_z-1}n_{f,r}=z-n_{f,s}.$$

That is why:

$$N_{sum}=\sum_{r=1}^{N_z-1}N_{M,r}+z-n_{f,s}, \tag{42}$$

where $n_{f,s}$ is the capacity of the last joint matrix in the working space of the master which is not subject to subsequent division among the slaves.

At the last stage when condition (26) is satisfied, the master reduces sub-matrix $\tilde{A}_{\Sigma,s}^{(0)}$ of the size of $\left(z-\sum_{j=1}^{r-1}n_{f,j}\right)\times\left(z-\sum_{j=1}^{r-1}n_{f,j}+1\right)$ to the upper diagonal form according to formulas (13) and (14).

We will calculate the number of operations of multiplication, division and subtraction performed by the master after the last computation. We will introduce the following designations:

$$z_\alpha=z-\sum_{j=1}^{r-1}n_{f,j}.$$

Matrix $A_U^{(s)}$ can be regarded as a joint square matrix of size z_α, to which a column of free members of the system of algebraic linear equations is attached. Reduction of such a matrix to the upper diagonal form with unities on the main diagonal requires [22]:

$$N_{f,s}=\frac{4z_\alpha^3+3z_\alpha^2-z_\alpha}{6}, \tag{43}$$

operations of division, multiplication, and subtraction.

Considering formula (43), the number of operations executed by the master in parallel computation, will be as follows:

$$N_{sum}=\sum_{r=1}^{N_z-1}N_{M,r}+\frac{4z_\alpha^3+3z_\alpha^2-z_\alpha}{6}+z-n_{f,s}. \tag{44}$$

Formula (44) allows analyzing the effectiveness of the parallel algorithm in comparison with the sequential algorithm of reducing a square matrix to the upper diagonal form with unities on the main diagonal.

After reducing the diagonal matrix to the upper diagonal form directly from formulas (13) and (14), it is necessary to execute the number of arithmetic operations that is computed from formula (43) after substituting z_α with z.

To assess the effectiveness of the developed algorithm of reducing a square matrix to the upper diagonal form, the number of arithmetic operations was calculated from formulas (41) and (44) and number of N_s operations from formula (43) (after substituting z_α with z). The efficiency of the algorithm was estimated as the ratio of $K_e=N_s/N_{sum}$.

Fig. 4 shows the result of this assessment. The following source data were taken:

$$\overline{z}=\left[10000\ 8000\ 6000\ 4000\ 2000\ 1000\ 500\ 300\right],$$

the number of processors $q=6$. The analysis of obtained results shows that the application of the parallel algorithm allows reducing the number of arithmetic operations needed to reduce the original matrix A_e to the upper diagonal form by more than nine times. It should be noted that the efficiency of the developed algorithm slightly decreases at a decrease in the size of matrix A_e.

All N_r operations in the r-th computation cycle are performed in parallel. Suppose that all operations of multiplication, division and subtraction at the r-th computation stage take τ_r units of time and the sending operations take $\tau_{t,r}$ units of time. Then the overall time consumption to implement the parallel algorithm of reducing matrix A_e to the upper diagonal form should be computed from the following formula:

$$T=\sum_{r=1}^{N_z-1}\left(N_{M,r}\tau_r+\tau_{t,r}n_{f,r}\right)+T_f, \tag{45}$$

where T_f is the time spent by the master to reduce his matrix to the upper diagonal at the last computation stage:

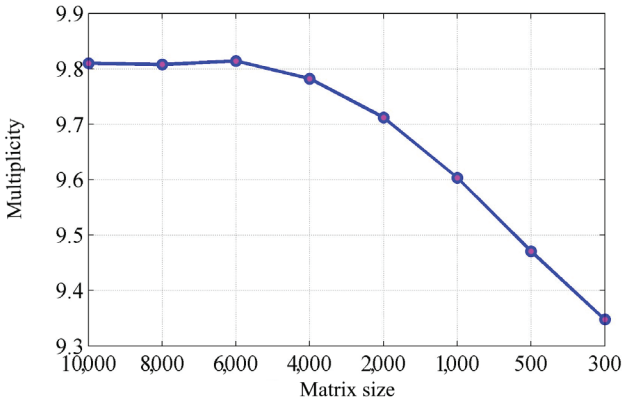

Fig. 4. Dependence of coefficient K_e on the size of matrix A_e

Given the value of $N_{f,s}$, we will write down formula (45) in the following form:

$$T = \sum_{r=1}^{N_z-1}\left(N_{M,r}\tau_r + \tau_{t,r}n_{f,r}\right) + \tau_f N_{f,s},$$

where τ_f is the time consumed to execute arithmetic operations at the final stage of computations.

After reducing matrix A_e to the upper diagonal form with unities on the main diagonal, parameters of model (1) are determined in the reverse order from formulas:

$$a_{A,z} = u_{z,z+1}, \quad a_{A,i} = u_{i,z+1} - \sum_{j=i+1}^{z} u_{ij}a_{A,j}, \quad i = \overline{z-1, 1}. \tag{46}$$

Let us analyze algorithm (46) of solution of the system of linear algebraic equations with triangular matrix U by the method of reverse substitution. The physical sense of the problem of synthesis of the optimum structure of the empirical model is that U is not a degenerated matrix.

Formula (46) does not determine the algorithm unambiguously, since the order of finding the sum is not determined. Consider a sequential operation of determining the sum held in the right-hand side of ratio (46). The corresponding recurrent procedure will be as follows:

$$a_{A,z}^{(0)} = a_{A,z} = u_{z,z+1},$$

$$a_{A,z-i}^{(j)} = a_{A,z-i}^{(j-1)} - u_{z-i,z-j+1}a_{A,z-j+1}^{(j-1)}, \quad a_{A,z-i}^{(0)} = u_{z-i,z+1},$$

$$a_{A,z-i} = a_{A,z-i}^{(i)}, \quad i = \overline{1, z-1}, \quad j = \overline{1, i}. \tag{47}$$

The procedure of finding solutions to equation (15) from formulas (47) has internal parallelism. Therefore, in order to save computer time, we will synthesize the corresponding parallel algorithm.

Loads between the master and the slaves will be distributed by the principle that was used when reducing matrix A_e to the upper diagonal form.

At computation stage 1, the last but one z column of matrix U, which has $z-1$ elements that are different from zero (not considering element $u_{zz}=1$), will be divided equally between q processors so that the master will have $h_{f,1}$ elements and the slaves will have $h_{w,1}$ elements each. From formula (47), we will calculate the value of $a_{A,z-i}^{(1)}$, where $i = \overline{1, h_{f,1}}$ for the master; $i = \overline{h_{f,1}+1, h_{w,k}^{(1)}}$ for $k=1$ and

$$i = \overline{h_{w,k}^{(1)}+1, h_{w,k+1}^{(1)}}, \quad k = \overline{1, q-2}$$

for the slaves. The slaves sent their values $a_{A,z-i}^{(1)}$ to the master, where the set of values $a_{A,z-1}$ and $a_{A,z-i}^{(1)}$, $i = \overline{2, z-1}$, which are stored in the workspace of the master, are generated.

At stage 2, $z-2$ elements of the $z-1$ column will be equally divided between q processors. As a result of this division the master obtained $h_{f,2}$ elements, and the slaves – $h_{w,k}^{(2)}$, elements. After this the master, including himself, sends the corresponding number of elements $a_{A,z-1}$ and $i = \overline{2, z-1}$, to each processor. Subsequently, from formula (47) we calculate the values of $a_{A,z-i-1}^{(2)}$, where $i = \overline{1, h_{f,2}}$ is for the master; $i = \overline{h_{w,k}^{(2)}+1, h_{w,k+1}^{(2)}}$, $k = \overline{1, q-2}$ – for processors, in this case $a_{A,z-2} = a_{A,z-2}^{(2)}$. Stage 2 is completed by generation in the workspace of the master of the set of values $a_{A,z-2}$ and $a_{A,z-i}^{(2)}$, $i = \overline{3, z-1}$.

Subsequent computation steps will be according to the presented circuit and at the r-th step, the master divides $z-r$ elements of the $z-r+1$ column between q processors in the following way: the master will have $h_{j,r}$ elements and the slaves will have $h_{w,k}^{(r)}$, $k = \overline{1, q-1}$ elements each. From formula (47), the values $a_{A,z-i-r+1}^{(r)}$, where $i = \overline{1, h_{f,r}}$ is for the master; $i = \overline{h_{w,k}^{(r)}+1, h_{w,k+1}^{(r)}}$, $k = \overline{1, q-1}$ are for the slaves. As a result of calculation, the workspace of the master, there will be the set of values $a_{A,z-r}$ and $a_{A,z-i}^{(r)}$, $i = \overline{r+1, z-1}$.

The process of solving the algebraic equation (15) by the method of reverse run with the use of the parallel algorithm is completed by the last computation step when condition $z-r=q$ is satisfied. Then the master calculates all values of $a_{A,z-i}$, where $i = \overline{r, z-1}$ from formula (47).

In the general case, the size of each column of matrix U can appear to be not multiple of the number of processors. That is why as before, the number of elements that will be in the workspace of slaves will be determined as the integer portion of the number:

$$h_{w,k}^{(i)} = \left[\frac{z-i}{q}\right], \quad i = \overline{1, r}, \quad k = \overline{1, q-1}. \tag{48}$$

Because each slave at the i-th computation step gets the same number of elements, then:

$$h_{w,1}^{(i)} = h_{w,2}^{(i)} = \cdots = h_{w,q-1}^{(i)} = h_{w,i}.$$

The number of elements of the $z-i+1$ column that the master obtains at the i-th step will be calculated from the following formula:

$$h_{f,i} = z - i - h_{w,i}(q-1), \quad i = \overline{1, z-1}. \tag{49}$$

After condition $z-i=q$ is satisfied, the values $h_{w,j}$, $j = \overline{1, q-1}$ are reset and the values $h_{f,j}$ should be calculated according to the following recurrent procedure:

$$h_{f,j+1} = h_{f,j} - 1, \; j = \overline{z-q, z-1}, \tag{50}$$

where $h_{f,z-q}=q$.

We will show that the chosen method of distribution of the computational load between the master and the slaves always ensured the satisfaction of inequity:

$$h_{f,i} \geq h_{w,i}. \tag{51}$$

The value of magnitude $h_{w,i}$, which is calculated fron formula (48), will be represented in the following form:

$$h_{w,i}=\frac{z-i}{q}-\rho_i,\ \ i=\overline{1,r}, \tag{52}$$

where $0\le\rho_i<1$ is the discarded fractional part of number $(z-i)/q$.

Substituting the obtained value $h_{w,i}$ in (49), we will have:

$$h_{f,i}=\frac{z-i}{q}+(q-1)\rho_i.$$

Since $h_{f,i}-h_{w,i}=q\rho_i\ge 0,$ there is also ratio (51).

Therefore, the distribution of elements between the master and the slaves of $z-i+1$ column is carried out in accordance with formulas (48) and (49). In this case, condition (50) is always satisfied.

We will calculate the number of the operations of multiplication and subtraction, which are carried out as a result of the implementation of the parallel algorithm of calculation according to formula (47). At the r-th computation step, the master will perform $N_0^{(r)}=2h_{f,r}$, and every i-th slave will perform $N_i^{(r)}=2h_{w,r}$ operations of multiplication and subtraction.

Since parallel computations are carried out by columns of matrix U, $r=\overline{1,z}$, and there is ratio (51), the total number of operations when implementing the parallel algorithm is:

$$N=\sum_{r=1}^{z-q}N_0^{(r)}+N_f,$$

where N_f is the number of the operations of multiplication and subtraction at the last stages of computations.

At transition from one computation step to another, the number of the elements of the next column is decreased by unity, so at a certain calculation stage, there will be condition $z-i=q$. Given the condition of completing of the calculation process, as well as formula (51), we come to the conclusion:

$$N=2\sum_{r=1}^{z-q}h_{f,r}+N_f. \tag{53}$$

Now let us calculate magnitude N_f. Final computation steps begin with condition $h_{f,s}=q$. The master performs further calculations of the parameters of the model according to formula (47). That is why the total number of operations of multiplication and subtraction at the last computation stage will be as follows:

$$N_f=2\sum_{i=0}^{q-1}(q-i)=q(q+1).$$

Considering the value of N_f, formula (53) will take the following form:

$$N=2\sum_{r=1}^{z-q}h_{f,r}+q(q+1).$$

Since between the loads of the master and that of every slave there is ratio (51),

$$h_{f,i}-h_{w,i}=q\rho_i. \tag{54}$$

it follows from equation (54) that $h_{f,i}=h_{w,i}+q\rho_i$. Considering formula (52), the value:

$$h_{f,i}=\frac{z-i}{q}+(q-1)\rho_i,\ \ i=\overline{1,\ z-q}.$$

Knowing $h_{f,i}$, we can calculate the value of N from formula (54):

$$N=2\sum_{r=1}^{z-q}\frac{z-r}{q}+\Lambda+q(q+1),$$

where

$$\Lambda=(q-1)\sum_{r=1}^{z-q}\rho_r.$$

Having calculated the value of $\sum_{r=1}^{z-q}(z-r)$, we come to conclusion:

$$N=\frac{z-q}{q}(z+q-1)+q(q+1)+\Lambda.$$

Realization of the sequential algorithm of computation of coefficients of mathematical model (1) after having received matrix U requires [22] $\tilde{N}=z(z-1)$ operations of multiplication and subtraction.

Fig. 5 gives an idea of efficiency of the parallel algorithm of computation of coefficients of empirical models (1) from formula (47) compared to the sequential algorithm.

Efficiency of the parallel algorithm was evaluated by magnitude of multiplicity $K_s=\tilde{N}/N$. Analysis of the obtained results shows that at an increase in dimensionality of matrix A_e, efficiency of the parallel algorithm falls almost exponentially. For example, at $z=100$ and $q=6$, multiplicity $K_s=5.3$. Because in the algorithm of the synthesis of empirical models that are optimal by structure based on genetic algorithms, the procedure of solving the system of equations must be repeated $M-c$ times, the gain in computation time will be noticeable even at relatively small sizes of matrix A_e.

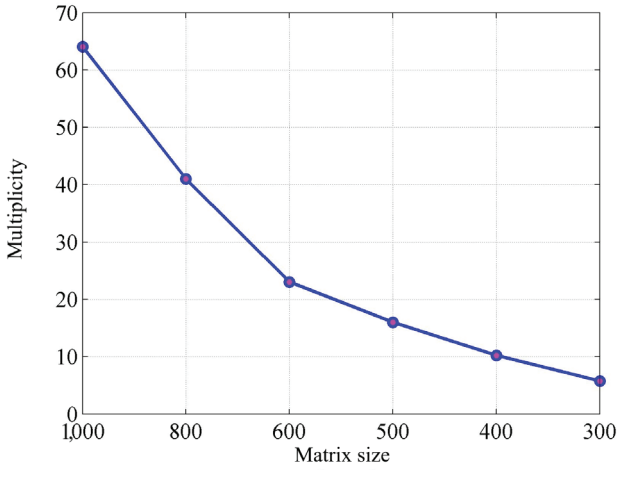

Fig. 5. Efficiency of the parallel algorithm at $q=6$

The total number of arithmetic operations that are executed during the implementation of the parallel algorithm will be determined by the number of operations to reduce matrix $\tilde{A}$ to the upper triangular form and number of operations to solve equation (15).

6. Discussion of results of studying the parallel algorithm using a Petri net

Construction of empirical models of optimal complexity based on the principles of genetic algorithms requires multiple solution of the system of linear algebraic equations with subsequent calculation of the model output. This procedure requires significant consumption of time. That is why there arose the need to develop an algorithm of parallel computation, which greatly accelerates the process of synthesis of the empirical model of optimal complexity. Previously, the stated task was solved on condition that the computational loads between processes are distributed evenly, which is rare in practice.

We proposed the new procedure for distribution of the computational load between the coordinating processor (master) and other processors (slaves), when computational load of the master is somewhat more that the computational load of each slave. For uneven load between parallel computational processes, the acceleration of the computational process as compared to the conventional algorithm was assessed.

The efficiency of the parallelized algorithm increases at an increase in dimensionality of a problem. In fact, the efficiency of the parallel algorithm will be slightly below the theoretical one due to the fact that operations of sending and interruptions were not taken into consideration.

To assess the real efficiency of the proposed parallel algorithm for synthesis of models of optimal complexity, it is necessary to conduct additionally a whole number of computational experiments.

7. Conclusions

1. Unlike the inductive method for self-organization of models, the authors propose to synthesize the empirical model in the form of a polynomial of a given degree. «1» or «0» is matched to each coefficient of the model (1 – the coefficient exists in the model; 0 – coefficient was withdrawn from the model). As a result, the structure of the model is encoded by the sequence, consisting of unities and zeros, called a chromosome. A set of chromosomes is generated – the population, the number of chromosomes of which is determined by the structure of a complete polynomial. The sequence of operations, the main of which are crossing and mutation, are performed on the obtained population. The use of the adjustment function allows selecting the most «adapted» chromosome, which determined the structure of an empirical model of optimal complexity.

2. Implementation of the method for synthesis of empirical models of optimal complexity indicated that, compared to the inductive method for self-organization, the proposed method requires less CPU time. For complex models with a large number of variables and a high polynomial power ($m \geq 3$), the consumption of computer time may be significant (up to one hour or more). To identify the ways to reduce such time consumption, the model of the computational process in the form of a Petri net was developed. Analysis of the model revealed that the process of synthesis of the empirical models of optimal complexity has internal parallelism.

3. The strategy of distribution of computational operations between the master and slaves, in which all slaves are loaded evenly, and the load of the master is always more than the load of slaves, was developed This strategy of separation made it possible to determine explicitly the required number of arithmetic operations for implementation of the parallel algorithm. It was shown that the developed strategy of load distribution between processors working in parallel (master and slaves) ensures the acceleration of the computational process by five or more times.

References

1. Gorbiychuk M. I., Schupak I. V., Oskolip T. Metod sinteza empiricheskih modeley s uchetom pogreshnostey izmereniy // Metody i pribory kontrolya kachestva. 2011. Issue 2 (27). P. 67–76.
2. Gorbiychuk M. I., Shufnarovych M. A. The Method of Constructing Mathematical Models of Complex Processes on the Basis of Genetic Algorithms // Iskusstvenniy intellekt. 2010. Issue 4. P. 50–57.
3. Metod syntezu empirychnykh modelei na zasadakh henetychnykh alhorytmiv / Gorbiychuk M. I., Kohutiak M. I., Vasylenko O. B., Shchupak I. V. // Rozvidka ta rozrobka naftovykh i hazovykh rodovyshch. 2009. Issue 4. P. 72–79.
4. Stepashko V. S., Bulgakova A. S. Obobschennyy iteratsionnyy algoritm metoda gruppovogo ucheta argumentov // Upravlyayuschie sistemy i mashiny. 2013. Issue 2. P. 5–17.
5. Gupta S., Bhardwaj S., Bhatia P. K. A reminiscent study of nature inspired computation // International Journal of Advances in Engineering & Technology. 2011. Vol. 1, Issue 2. P. 117–125.
6. Voevodin V., Antonov A., Popova N. Studying the Structure of Parallel Algorithms as a Key Element of High-Performance Computing Education // Lecture Notes in Computer Science. 2019. P. 199–210. doi: https://doi.org/10.1007/978-3-030-10549-5_16
7. Ortega J. M. Introduction to parallel and vector solution of linear systems. Springer, 1988. doi: https://doi.org/10.1007/978-1-4899-2112-3
8. Dvukhglavov D. E., Kulynych V. E. Development of software solution for building route of a orders group delivery in presence of time constraints // Bulletin of National Technical University «KhPI». Series: System Analysis, Control and Information Technologies. 2017. Issue 55. P. 64–71. doi: https://doi.org/10.20998/2079-0023.2017.55.11
9. Parallel'nye algoritmy resheniya zadach vychislitel'noy matematiki / Himich A. N., Molchanov I. N., Popov A. V., Chistyakova T. V., Yakovlev M. F. Kyiv: Naukova dumka, 2008. 248 p.
10. Khimich A. N., Popov A. V., Polyankoa V. V. Algorithms of parallel computations for linear algebra problems with irregularly structured matrices // Cybernetics and Systems Analysis. 2011. Vol. 47, Issue 6. P. 973–985. doi: https://doi.org/10.1007/s10559-011-9377-4

11. Rutkovskaya D., Pilin'skiy M., Rutkovskiy L. Neyronnye seti, geneticheskie algoritmy i nechetkie sistemy. Moscow: Goryachaya liniya-Telekom, 2004. 452 p.
12. Bogatyrev M. Yu. Invarianty i simmetrii v geneticheskih algoritmah. URL: http://www.raai.org/conference/cai-08/files/cai-08_paper_287.pdf
13. Gladkov L. A., Kureychik V. V., Kureychik V. M. Geneticheskie algoritmy. 2-e izd., ispr i dop. Moscow: FIZMATLIT, 2006. 320 p.
14. Gorbiychuk M. I., Medvedchuk V. M., Pashkovskyi B. V. The parallelism in the algorithm of the synthesis of models of optimal complexity based on the genetic algorithms // Eastern-European Journal of Enterprise Technologies. 2014. Vol. 4, Issue 2 (70). P. 42–48. doi: https://doi.org/10.15587/1729-4061.2014.26305
15. Gorbiychuk M. I., Shufnarovych M. A. Metod syntezu matematychnykh modelei kolyvnykh protsesiv z nekratnymy chastotamy // Naukovyi visnyk Ivano-Frankivskoho natsionalnoho tekhnichnoho universytetu nafty i hazu. 2010. Issue 1. P. 105–112.
16. Luszczek P. Parallel Programming in MATLAB // The International Journal of High Performance Computing Applications. 2009. Vol. 23, Issue 3. P. 277–283. doi: https://doi.org/10.1177/1094342009106194
17. Verzhbitskiy V. M. Osnovy chislennyh metodov: ucheb. Moscow: Vysshaya shkola, 2002. 840 p.
18. Paterson Dzh. Teoriya setey Petri i modelirovanie sistem. Moscow: Mir, 2000. 263 p.
19. Marahovskiy V. B., Rozenblyum L. Ya., Yakovlev A. V. Modelirovanie parallel'nyh protsessov. Seti Petri: kurs dlya sistemnyh arhitektorov, programmistov, sistemnyh analitikov, proektirovschikov slozhnyh sistem upravleniya. Sankt-Peterburg: Professional'naya literatura, 2014. 398 p.
20. Gorbiychuk M. I., Medvedchuk V. M., Lazoriv A. N. Analysis of Parallel Algorithm of Empirical Models Synthesis on Principles of Genetic Algorithms // Journal of Automation and Information Sciences. 2016. Vol. 48, Issue 2. P. 54–73. doi: https://doi.org/10.1615/jautomatinfscien.v48.i2.60
21. Korn G., Korn T. Spravochnik po matematike dlya nauchnyh rabotnikov i inzhenerov. Moscow: Nauka, 1978. 832 p.
22. Volkov E. A. Chislennye metody: ucheb. pos. 2-e izd., isp. Moscow: Nauka, 1987. 248 p.