Journal of Computer Science 6 (3): 261-268, 2010
ISSN 1549-3636

Design and Implementation of Parallel Subunit for Synthesis Mathematical Models

Issa M. Shehabat
King Saud University, KSA-Riyadh-Huraimla 11962, P.O. Box 300, Saudi Arabia

Abstract: Problem statement: Mathematical modeling of different natural and technical objects and processes is one of the most important directions that needs high performance computing with huge memory. To reduce the computational time and expenses we need to carry out the calculations on specialized subunits. **Approach:** We described a self-organizing approximation method and introduced a new methodology of structural synthesis of specialized parallel processing subunits for realizing a group method of data handling algorithms. **Results:** The design procedure of the parallel subunit in addition to the selection of the computing units for this device has been introduced. **Conclusion/Recommendations:** The Group Method of Data Handling proved to be most effective to solve small and medium-sized problems with continuous output. It was tested on wide range of artificial and real-world problems.

Key words: Self-organizing algorithms, GMDH, parallel subunit

INTRODUCTION

One of the most common problems in engineering design and control is the problem of mathematical modeling. Consider the object under investigation as "black box" with several input variables (inputs) and one output variable (output). The purpose of modeling is to find some means of predicting the value output for any values of input, based on a set of learning data.

One of the methods of the mathematical modeling used for this purpose is the Group Method of Data Handling (GMDH) (Ivakhnenko, 1971; Farlow, 1984; Ivakhnenko *et al*., 1994; Dolenko *et al*., 1996).

There were many papers published and several books devoted to group method of data handling and its applications. GMDH can be considered as further propagation of inductive self-organizing methods to the solution of more complex practical problems (Ivakhnenko and Ivakhnenko, 1995). Most of GMDH algorithms use the polynomial reference functions. This method involves sorting, that is successive testing of models selected out of a set of candidate models according to specified criterion. Nearly all known GMDH algorithms use polynomial support functions. General connection between input and output variables can be found in the form of functional Volterra series, whose discrete analogue is known as the Kolmogorov-Gabor polynomial (Madala and Ivakhnenko, 1994):

$$y = a_0 + \sum_{i=1}^{M} a_i x_i + \sum_{i=1}^{M}\sum_{j=1}^{M} a_{ij} x_i x_j + \sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{M} a_{ijk} x_i x_j x_k$$

Where:
$X(x_1,x_2,\ldots,x_M)$ = The vector of the input variables
$A(a_1, a_2,\ldots, a_M)$ = The vector of the summands coefficients

In the iterative multilayered GMDH algorithm the iteration rule remains unchanged for all sequence, as shown in Fig. 1, the first layer tests the models that can be derived from the information contained in any two columns of the sample. The second uses information from four columns, the third from any eight columns, and so forth. the exhaustive-search termination rule is that in each layer the optimal models are selected by the minimum of external criterion e.g.:

$$E_k^1 = \sum_{i=1}^{m} (y_{2i}^{k1} - y_{2i})^2 / m \quad (1)$$

Where:
E_k^1 = Selection criterion for $k^{th\ partial}$ description of the first layer
$y2_i$ = The value of the function $f(x_1,x_2)$ on $2i^{th}$ point initial the experimental data m-number of testing points

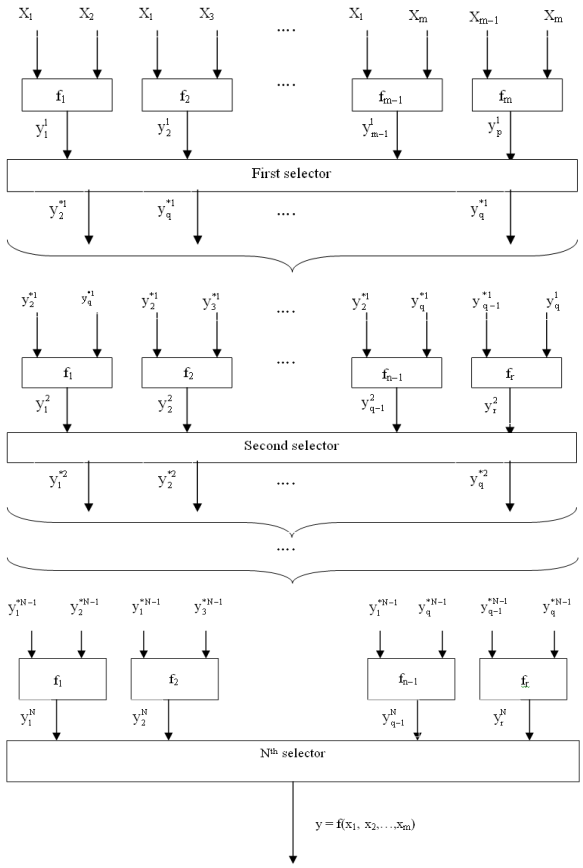

Fig. 1: Multilayered iteration algorithm

MATERIALS AND METHODS

Basics of the method: The idea of GMDH is the following: we are trying to build an analytical function (called "model") which would behave itself in such a way that the predicted value of the output would be as close as possible to its actual value. For many applications such an analytical model is much more convenient than the "distributed knowledge" representation that is typical for neural network approach.

The most common way to deal with such problem is to use linear regressing approach. In this approach, first of all we must introduce a set of basis functions. The answer will then be sought as a linear combination of the basis functions. For example, powers of input variables along with their double and triple cross-products may be chosen as bases functions.

To obtain the best solution, we should try all possible combinations of terms and choose those which give best predictions. The decision about quality of each model must be made using some numeric criterion. (Accurate choice of the criterion is separate problem.) However, it is clear that full testing for a problem with many inputs and a wide set of a basis functions is practically impossible, as it would take too much time and it would require too much computer memory, to reduce computational expenses, one should reduce the number of basis functions (and the number of input variables), which are used to build the tested models. To do that, one must change from one-stage procedure of model selection to a multi-stage procedure.

Let us take two input variables and let us combine a set of basis functions. For example, if we denote input variables as x_1 and x_2, let the set of basis functions be $\{1,x_1,x_2, x_1.x_2\}$.(1 corresponds to constant bias and must be always included in the set). Now we check 2^4-1=15 possible models and choose one that is the best. (Any one of the tested models is often called partial description or PD). After that, we take another pair of input variables and repeat operation, resulting in one more PD with its own value of criterion. Doing the same for each possible pair of n input variables, we obtain n*(n-1)/2PDs, each with its own value of the used criterion.

Then we compare these values and choose several PDs which give better approximation for the output variable. Usually we select a pre-defined number F of best PDs that must be preserved at the next step of algorithm.

The values predicted by the preserved PDs (Called Survivors), serve at the next iteration as input variable along with initial input variables of the whole system. All the described actions are repeated again with the broadened set of input variables and then the next iteration goes, and so on.

This method involves sorting, that is successive testing of model selected out of a set of candidate models according to a specified criterion. Nearly all known GMDH algorithms use polynomial support functions.

GMDH algorithms realization: A parallel computing can be implemented for realizing all algorithms that have multilayered structures and many different multiprocessor systems were designed such as multi-section and two-section pipeline architectures (Dmitrienko *et al*., 1998). To get the greatest gain in productivity of the pipeline systems, in this work, it is recommended to carry out calculations in specialized processing units (subunit) by entering in ALU structure additional hardware, multiplication units, division units, addition units, subtraction units and cache memory. The function of each subunit is determined: Each of them

forms on each i^{th} (i>1) layer a system of Gauss equations for all learning subsample points that are represented as:

$$y^i_{p1,...pd} = a^i_{0,p1,...pd} + a^i_{1,p1,...pd}\, y^{i-1}_{p1} + a^i_{2,p1,...pd}\, y^{i-1}_{p2} + ... + a^i_{d,p1,...pd}\, y^{i-1}_{pd} + \sum_{k=1}^{d}\sum_{\substack{j=1\\ j\le k}}^{d} a^i_{kj,p1,...,pd} y^{i-1}_{pk} y^{i-1}_{pj} \quad (2)$$

Where:

i = Points to the selection layer

y^{i-1}_{pq}, q = 1, q = The best partial descriptions $(i-1)^{st}$ layer

$a^i_{0,p1,...pd},...,a^i_{dd,p1,...pd}$ = Definable coefficients

Conditional Gauss equations on the learning subsample points $y_i, x_{1i}, x_{2i}, i = \overline{i,m}$ for Eq. 2 could be written as:

$$\begin{aligned} y^i_{p1,...,pd}(x_{11},x_{21}) &= a^i_{0,p1,...,pd} + a^i_{1,p1,000,pd} y^{i-1}_{p1}(x_{11},x_{21}) + ... + \sum_{k=1}^{d}\sum_{\substack{j=1\\ j\le k}}^{d} a^i_{kj,p1,...,pd} y^{i-1}_{pk}(x_{11},x_{21}) y^{i-1}_{pj}(x_{11},x_{21}) \\ &..... \\ y^1_{p1,...,pd}(x_{11},x_{21}) &= a^i_{0,p1,...,pd} + a^i_{1,p1,000,pd} y^{i-1}_{p1}(x_{1m},x_{2m}) + ... + \sum_{k=1}^{d}\sum_{\substack{j=1\\ j\le k}}^{d} a^i_{kj,p1,...,pd} y^{i-1}_{pk}(x_{1m},x_{2m}) y^{i-1}_{pj}(x_{1m},x_{2m}) \end{aligned} \quad (3)$$

where, $m \ge d + c^2_d + 1$, C^2_d-combination of d by 2.

From the equation system (3) with $m > d + c^2_d + 1$ we got a system of normal gauss equations:

$$\begin{aligned} \bar{y} &= a_0 + a_1 \overline{y_{p1}} + ... + a_d \overline{y_{pd}} + \sum_{k=1}^{d}\sum_{\substack{j=1\\ j\le k}}^{d} a_{ij} \overline{y_{pk} y_{pj}} \\ \overline{yy_{p1}} &= a_0 \overline{y_{p1}} + a_1 \overline{y_{p1} y_{p2}} + ... + a_d \overline{y_{pd} y_{p1}} + \sum_{k=1}^{d}\sum_{\substack{j=1\\ j\le k}}^{d} a_{ij} y_{pk} y_{pj} y_{p1} \\ &..... \\ \overline{yy_{pd} y_{pd}} &= a_0 \overline{y_{pd} y_{pd}} + a_1 \overline{y_{p1} y_{pd} y_{pd}} + ... + a_d \overline{y_{pd} y_{pd} y_{pd}} + \sum_{k=1}^{d}\sum_{\substack{j=1\\ j\le k}}^{d} a_{ij} \overline{y_{pk} y_{pj} y_{pd} y_{pd}}, \end{aligned} \quad (4)$$

Where:

$a_0 \equiv a^i_{0,p1,...pd}, a_1 \equiv a^i_{1.p1,...ppd}, ..., a_{dd} \equiv a^i_{dd,p1,..pd}$

$$\bar{y} = \frac{1}{m}\sum_{q=1}^{m} y(x_{1q}, x_{2q})$$

$$\overline{y_{p1}} = \frac{1}{m}\sum_{q=1}^{m} y^{i-1}_{p1}(x_{1q}, x_{2q})$$

$$ypd = \frac{1}{m}\sum_{q=1}^{m} y^{i-1}_{pd}(x_{1q}, x_{2q})$$

$$\overline{y_{p1} y_{pj}} = \frac{1}{m}\sum_{q=1}^{m} y^{i-1}_{pk}(x_{1q}, x_{2q}) y^{i-1}_{pj}(x_{1q}, x_{2q}), k, j = \overline{1,m}, j \le k \quad (5)$$

$$\overline{yy_{p1}} = \frac{1}{m}\sum_{q=1}^{m} y(x_{1q}, x_{2q}) y^{i-1}_{p1}(x_{1q}, x_{2q}) \quad (6)$$

$$\overline{y_{pk} y_{pj} y_{p1}} = \frac{1}{m}\sum_{q=1}^{m} y^{i-1}_{pk}(x_{1q}, x_{2q}) y^{i-1}_{pj}(x_{1q}, x_{2q}) y^{i-1}_{p1}(x_{1q}, x_{2q}) \quad (7)$$

$$\overline{y_{pk} y_{pj} y_{pd} y_{pd}} = \frac{1}{m}\sum_{q=1}^{m} y^{i-1}_{pk}(x_{1q}, x_{2q}) y^{i-1}_{pj} (x_{1q}, x_{2q}) y^{i-1}_{pd}(x_{1q}, x_{2q}) y^{i-1}_{pd}(x_{1q}, x_{2q}) \quad (8)$$

Solving a system of normal linear gauss Eq. 4 for each of the partial descriptions (2), we find $a^i_{0,p1,...,pd}, a^i_{1,p1,....pd}, ...a^i_{dd,pa,...,pd}$-coefficients for these descriptions. Then the produced models estimate the set of checking subsample points using a selection criterion (1):

$$E^i_{p1,...pd} = \sum_{q=1}^{m1} (y_q - (a^i_{0,p1,...,pd} + a^i_{1,p1,...,pd} y^{i-1}_{p1}(x_{1q}, x_{2q}) + ... + \sum_{k=1}^{d}\sum_{\substack{j=1\\ j\le k}}^{d} a^i_{kj,p1,...,pd} y^{i-1}_{pk}(x_{1q}, x_{2q}) y^{i-1}_{pj}(x_{1q}, x_{2q})))^2 \quad (9)$$

where, m1-number of checking subsample points.

We can write the system of normal Gauss Eq. 4 for the subunit in the following form:

$$\begin{aligned} y^{k1} &= a^k_1 y^{p(i-1)}_1 + a^k_2 y^{q(i-1)}_1 \\ y^{k2} &= a^k_1 y^{p(i-1)}_2 + a^k_2 y^{q(i-1)}_2 \end{aligned} \quad (10)$$

Where:

$$y^{k1} = \sum_{j=1}^{N1} y_j \bar{y}^{-p(i-1)}_j \quad (11)$$

$$y^{k2} = \sum_{j=1}^{N1} y_j \bar{y}^{-q(i-1)}_j \quad (12)$$

$$y^{p(i-1)}_1 = \sum_{j=1}^{N1} \bar{y}^{-p(i-1)}_j \bar{y}^{-p(i-1)}_j \quad (13)$$

$$y_1^{q(i-1)} = y_2^{p(i-1)} = \sum_{j=1}^{N1} \bar{y}_j^{p(i-1)} \bar{y}_j^{q(i-1)} \quad (14)$$

$$y_2^{q(i-1)} = \sum_{j=1}^{N1} \bar{y}_j^{q(i-1)} \bar{y}_j^{q(i-1)} \quad (15)$$

Where:

N_1 = Number of learning points

i = Number of the layer (i>1)

y_j = The value of the function in the j^{th} point of the initial data learning subsample

$\bar{y}_j^{p(i-1)}, \bar{y}_j^{q(i-1)}$ = The values of best partial descriptions $y^{p(i-1)}$, $y^{q(i-1)}$

In the j^{th} point in the initial data on $(i-1)^{st}$ layer.

The next important function of the subunit is solving a system of Eq. 10:

$$a_1^k = \frac{y^{k1}}{y_1^{p(i-1)}} - \frac{a_2^k y_1^{q(i-1)}}{y_1^{p(i-1)}} \quad (16)$$

$$a_2^k = \frac{y^{k2} - \frac{y^{k1} y_2^{p(i-1)}}{y_1^{p(i-1)}}}{y_2^{q(i-1)} - \frac{(y_1^{q(i-1)})^2}{y_1^{p(i-1)}}} \quad (17)$$

After determining the coefficients (16), (17) we get the model of i^{th} selection layer:

$$y^{ki} = a_1^k \bar{y}_p^{p(i-1)} + a_2^k \bar{y}_q^{q(i-1)} \quad (18)$$

which is evaluated on the checking subsample point, by using the following criterion:

$$\delta = \frac{1}{N_2} \sum_{j=1}^{N_2} \delta j = \frac{1}{N_2} \sum_{j=1}^{N_2} \left(y_{jmul} - a_1^k \bar{y}_{jmul}^{p(i-1)} - a_2^k \bar{y}_{jmul}^{q(i-1)} \right)^2$$

Where:

N_2 = The number of checking subsample points

δ_j = Square error in the j^{th} point of the checking subsample

y_{jmul} = The value of the function of the j^{th} point of checking subsample initial data

$\bar{y}_{pjmul}^{(i-1)}, \bar{y}_{qjmul}^{(i-1)}$ = The values of the partial descriptions $\bar{y}_p^{(i-1)}$, $\bar{y}_q^{(i-1)}$ on the j^{th} point of checking subsample initial data

The design of parallel subunit: We must pass from the formal representations (10-19) of the working algorithm of the subunit to its parallel tear form, which presented by (Voevodin, 1986). Assuming that the parallel system has five processing elements that perform binary multiplication, two of them performing also the division operation in addition to five processor elements that perform the addition and subtraction operations.

Then we have:

Data: $y_1, \ldots, y_{N1}, y_{1mul}, \ldots, y_{N2mul}, \bar{y}_{1p}^{(i-1)}, \ldots, \bar{y}_{pN1}^{(i-1)}, \bar{y}_{p1mul}^{(i-1)}, \ldots, \bar{y}_{pN2mul}^{(i-1)}, \bar{y}_{q1}^{(i-1)}, \ldots, \bar{y}_{qN1}^{(i-1)}, \bar{y}_{q1mul}^{(i-1)}, \ldots, \bar{y}_{qN2mul}^{(i-1)}$

Parallel-tier form:

Tier 1: $y_1 \bar{y}_{p1}^{(i-1)}$, $y_1 \bar{y}_{q1}^{(i-1)}$, $\bar{y}_{p1}^{(i-1)} \bar{y}_{p1}^{(i-1)}$, $\bar{y}_{p1}^{(i-1)} \bar{y}_{q1}^{(i-1)}$, $\bar{y}_{q1}^{(i-1)} \bar{y}_{q1}^{(i-1)}$

Tier 2: $y_2 \bar{y}_{p2}^{(i-1)}$, $y_2 \bar{y}_{q2}^{(i-1)}$, $\bar{y}_{p2}^{(i-1)} \bar{y}_{p2}^{(i-1)}$, $\bar{y}_{q2}^{(i-1)} \bar{y}_{p2}^{(i-1)}$, $\bar{y}_{q2}^{(i-1)} \bar{y}_{q2}^{(i-1)}$

Tier 3: $y_3 \bar{y}_{p3}^{(i-1)}$, $y_3 \bar{y}_{q3}^{(i-1)}$, $\bar{y}_3^{p(i-1)} \bar{y}_{p3}^{p(i-1)}$, $\bar{y}_{p3}^{(i-1)} \bar{y}_{q3}^{(i-1)}$, $\bar{y}_{q3}^{(i-1)} \bar{y}_{q3}^{(i-1)}$, $y_1 \bar{y}_{p1}^{(i-1)} + y_2 \bar{y}_{p2}^{(i-1)}, \ldots,$ $\bar{y}_{q1}^{(i-1)} \bar{y}_{q1}^{(i-1)} + \bar{y}_{q2}^{(i-1)} \bar{y}_{q2}^{(i-1)}$

Tier N1: $y_{N1} \bar{y}_{pN1}^{(i-1)}, y_{N1} \bar{y}_{qN1}^{(i-1)}, \ldots, \bar{y}_{qN1}^{(i-1)} \bar{y}_{qN1}^{(i-1)}, \sum_{j=1}^{N1-2} y_j \bar{y}_{pj}^{(i-1)} +$ $y_{N1-1} \bar{y}_{p(N1-1)}^{(i-1)}, \ldots, \sum_{j=1}^{N1-2} \bar{y}_{qj}^{(i-1)} \bar{y}_{qj}^{(i-1)} + \bar{y}_{q(N1-1)}^{(i-1)} \bar{y}_{q(N1-1)}^{(i-1)}$

Tier N1+1: $y_{p1}^{(i-1)} = \sum_{j=1}^{N1} \bar{y}_{pj}^{(i-1)} \bar{y}_{pj}^{(i-1)}$, $y_{1q}^{(i-1)} = \sum_{j=1}^{N1} \bar{y}_{pj}^{(i-1)} \bar{y}_{qj}^{(i-1)}$

Tier N1+2: $c_1 = \frac{y^{k1}}{y_{1p}^{(i-1)}}$, $c_2 = \frac{y_{1q}^{(i-1)}}{y_{1p}^{(i-1)}}$

Tier N1+3: $c_1 y_{2p}^{(i-1)}$, $c_1 y_{1q}^{(i-1)}$

Tier N1+4: $c_3 = k2 - c_1 y_2^{p(i-1)}$, $c4 = y_{2q}^{(i-1)} - c_2 y_{1q}^{(i-1)}$

Tier N1+5: $a_2^k = c_3 / c_4$

Tier N1+6: $a_2^k c_2$

Tier N1+7: $a_1^k = c_1 - a_2^k \frac{y_{1q}^{(i-1)}}{y_{1p}^{(i-1)}} = c1 - a_2^k c_2$

Tier N1+8: $a_1^k \bar{y}_{p1mul}^{(i-1)}$, $a_1^k \bar{y}_{p2mul}^{(i-1)}, \ldots, a_1^k \bar{y}_{p5mul}^{(i-1)}$

TierN1+9: $a_2^k \bar{y}_{q1mul}^{(i-1)}$, $a_2^k \bar{y}_{q2mul}^{(i-1)}, \ldots, a_2^k \bar{y}_{q5mul}^{(i-1)}$, $y_{1mul} - a_1^k \bar{y}_{p1mul}^{(i-1)}, \ldots, y_{5mul} - a_1^k \bar{y}_{p5mul}^{(i-1)}$

Tier N1+10: $a_1^k \bar{y}_{p6mul}^{(i-1)}, \ldots, a_1^k \bar{y}_{p10mul}^{(i-1)}$, $y_{1mul} - a_1^k \bar{y}_{p1mul}^{(i-1)} - a_2^k \bar{y}_{q1mul}^{(i-1)}, \ldots, y_{5mul} - a_1^k \bar{y}_{p5mul}^{(i-1)} - a_2^k \bar{y}_{q5mul}^{(i-1)}$

Tier N1+11: $\delta_1 = \left(y_{2mul} - a_1^k \overline{y}_{p1mul}^{(i-1)} - a_2^k \overline{y}_{q1mul}^{(i-1)}\right)^2, \ldots,$

$\delta_5 = \left(y_{5mul} - a_1^k \overline{y}_{p5mul}^{(i-1)} - a_2^k \overline{y}_{q5mul}^{(i-1)}\right),$

$y_{6mul} - a_1^k \overline{y}_{p6mul}^{(i-1)}, \ldots, y_{10mul} - a_1^k \overline{y}_{p10mul}^{(i-1)}$

Tier N1+12: $a_2^k \overline{y}_{q6mul}^{(i-1)}, \ldots, a_2^k \overline{y}_{q10mul}^{(i-1)}.$

Tier N1+13: $y_{6mul} - a_1^k \overline{y}_{p6mul}^{(i-1)} - a_2^k \overline{y}_{q6mul}^{(i-1)}, \ldots, \quad y_{10mul} - a_1^k \overline{y}_{p10mul}^{(i-1)} - a_2^k \overline{y}_{q10mul}^{(i-1)}, \; a_1^k \overline{y}_{p11mul}^{(i-1)}, \ldots, \; a_2^k \overline{y}_{p15mul}^{(i-1)}$

Tier N1+14: $\delta_6, \delta_7, \ldots, \delta_{10}, \quad y_{11mul} - a_1^k \overline{y}_{p11mul}^{(i-1)}, \ldots, \quad y_{15mul} - a_2^k \overline{y}_{p15mul}^{(i-1)}$

Tier N1+15: $\delta_1 + \delta_6, \delta_2 + \delta_7, \ldots, \delta_5 + \delta_{10}, \quad a_2^k \overline{y}_{q11mul}^{(i-1)}, \ldots, a_2^k \overline{y}_{q15mul}^{(i-1)}$

Tier N1+3m+7: $y_{(N_2-4)mul} - a_1^k \overline{y}_{p(N_2-4)mul}^{(i-1)} - a_2^k \overline{y}_{q(N_2-4)mul}^{(i-1)}, \ldots, y_{N_2mul} - a_1^k \overline{y}_{pN_2mul}^{(i-1)} - a_2^k \overline{y}_{qN\ mul}^{(i-1)}$

Tier N1+3m+8: $\delta_{N_2-4}, \delta_{N_2-3}, \ldots, \delta_{N_2}$

Tier N1+3m+9: $\delta_1 + \delta_6 + \ldots + \delta_{N_2-4} + \delta_2 + \delta_7 + \delta_{N_2-3}, \ldots, \delta_5 + \delta_{10} + \ldots + \delta_{N_2}$

Tier N1+3m+10: $\delta_1 + \delta_6 + \ldots + \delta_{N_2-4} + \delta_2 + \delta_7 + \delta_{N_2-3}, \; \delta_3 + \delta_8 + \ldots + \delta_{N_2-4} + \delta_4 + \delta_9 + \ldots + \delta_{N_2-1}$

Tier N1+3m+11: $\sum_{j=1}^{N_2} \delta_j$

Tier N1+3m+12: $\delta = \dfrac{\sum_{j=1}^{N_2} \delta_j}{N_2}$

RESULTS AND DISCUSSION

The first (N_1+1) tiers form the equation system (10). On the first and second tiers, only the first two summands of the (11-15) equations are computed. On the N_1^{st} tiers not only corresponding summands are computed, but also summands from the previous tiers added. This addition ends on the $(N_1+1)^{st}$ tier, corresponding to (11-15) equations and produces $y^{k1}, y^{k2}, y_{1p}^{(i-1)}, y_{1q}^{(i-1)} = y_{2p}^{(i-1)}, y_{2q}^{(i-1)}$. the tiers from third to N_{1-st} have the maximum height of the algorithm parallel form and equal 10 and needs for realizing for the algorithm five processor elements, that performs the multiplication operation and five two input addition units. On the first N_1 tiers $5N_1$ multiplication operations are performed, for performing $5N_1$ multiplication operations the initial data only is required, so with the best performance requirements to the subunit the first (N_1+1) tiers could be replaced by two. On the first tier all the multiplication operations with the help of $5N_1$ multiplication units and on the second performing simultaneous addition of N_1 summands.

On the tiers from $(N_1+2)^{nd}$ to the $(N_1+7)^{th}$ corresponding to (16), (17) equations, determined the coefficients a_1^k, a_2^k of (4.9), equations that are synthesis on the i^{th} tier of the learning subsample points. On these tiers not more than two processor elements are used, however on the $(N_1+2)^{nd}$ tier performed two division operations.

On the tiers from $(N_1+8)^{th}$ to the last(end) determined the mean-square error for the model (18) on the checking subsample points. Forming these tiers we assumed that the number of checking subsample points N_2 is a multiple of 5 (e.g., N_2 = 5 m, m is an integer). This is a general assumption, taking into account that if $N_2 \neq 5$ m we can use this parallel form of the algorithm, but some of the processor elements that are used with N_2 = 5 m, will not be used.

With N_1 = 5(m-1) the height of the algorithm of the parallel form can be decreased by 1 because of performing a part of addition operations of $(N_1+3m+9)^{th}$ and $(N_1+3m+10)^{th}$ tiers on the N_1+3m+8 tier. We mention that when it is necessary we can perform all multiplication operations that are related to the values of the model (18) on the checking subsample points on the tiers $(N_1+8)^{th}$-$(N_1+3m+6)^{th}$ could be performed on one tier, but $2N_2$ processor elements are required. With the availability of additional N_2 3-input addition units that are necessary for computing the values $\sqrt{\delta_1}, \sqrt{\delta_2}, \ldots, \sqrt{\delta_{N2}}$ and N_2-input addition unit to produce the sum $\sum_{j=1}^{N2} \delta_j$ the last 3m+5 tiers can be replaced by 5 tiers. The minimal height of the algorithm parallel form with the account of two-tier exchange of the first (N_1+1) tiers will be 13. But, the load of such system will be very small, because of the use of not more than two processor elements on the seven tiers of the algorithm.

The number of addition and multiplication/division operations on each tier for an algorithm of N_1+3m+12 height is shown in Table 1.

With $N_1 = N_2 = 15$ we have 36 tiers of the parallel algorithm. On these tiers 240 operations are performed, 127 multiplication and division operations and 113 addition and subtraction operations.

From the parallel computing of multiplication and addition operations follows the necessary of timing both operations using two types of computing devices.

However stands another question about implementing one or two universal mul/div devices or multiplier with built-in one or two divisors. By entering one divider, the number of tiers increased by 1, so the two division operations on the (N_1+2)-nd tier will be computed sequentially. But entering two dividers, one of them will be used only one time on the (N_1+2)-nd tier and the second four tiers. This selection must be based on the requirements to the problem-oriented computing devices, taking into account both the performance and the cost of the system. In this work we use two universal mul/div devices, which need approximately the same time to perform both operations.

In this work the largest time needed for multiplication and addition operations and the addition operations also needs the least time, then the selection of optimal multiplication devices directly Affect the subunit characteristics. Learning the available multipliers shows that:

Learning the available multiplication and division units (Kung, 1991; Veshinchouk and Cherkasky, 1990) shows that the best by the means of performance are matrix ones (Kung, 1991; Veshinchouk and Cherkasky, 1990). So some tiers perform only addition operations (Table 1a, b, and c) the best adders are the parallel ones (Gex, 1971; Saveliev, 1987).

Knowing the times τ_{mul}, τ_{div}, τ_{add} ($\tau_{mul} \approx \tau_{div} = \tau$, $\tau_{add}<0.1\tau$) needed to perform the arithmetic operations(multiplication, division and addition) in parallel subunit, we can evaluate its performance and the load coefficients of processing units according to their functionality algorithm.

Form parallel-tier algorithm of the subunit and Table 1a, b, and c we can see that, the computational units of the subunit have to perform not less than five addition operations at the time τ. In this case the total number of algorithm tiers n can be calculated by the following expression:

$$n = N_1 + \frac{3}{5}N_2 + 12 = n_{mul} + n_{add}$$

Where:

N_1,N_2 = Number of learning and checking subsample points consequently

n_{mul} = Number of tiers with multiplication operations

n_{add} = Number of tiers which perform only addition operations

On the first N_1+7 tiers performs $5N_1$+6 multiplication and division operations and $5(N_1-1)+3$ addition operations on the tiers from N_1+8 to n-$3N_2$+1 multiplication and division operations and $2N_2$+10 addition operations. Ignoring other operations we can calculate k_{per} -performance coefficient of the parallel subunit in comparison with serial computers.:

$$K_{Per} = \frac{(5N_1 + 3N_2 + 7)\tau + (5N_1 + 2N_2 + 8)\tau_{add}}{(5N_1 + \frac{3}{5}N_{2+5})\tau + 7\tau_{add}}$$

And also k_{mul} the load coefficient of multiplier (multiplier/divider):

$$k_{mul} = \frac{(5N_1 + 3N_2 + 7)\tau}{5(N_1 + \frac{3}{5}N_2 + 5)\tau + 7\tau_{add}}$$

Since $\tau_{add} < 0.1\tau$, then all the addition operations on any tier could be performed by one adder. And its load coefficient k_{add} can be calculated by the following equation:

$$k_{mul} = \frac{(5N_1 + 2N_2 + 8)\tau_{add}}{(N_1 + \frac{3}{5}N_2 + 5)\tau + 7\tau_{add}}$$

Table 1a: the number of Addition, Multiplication and division operations on the tiers 1 to N_1+7

Tier/N^Q	1	2	3	4	…	N_1	N_1+1	N_1+2	N_1+3	N_1+4	N_1+5	N_1+6	N_1+7
Number of mul/div operations	5	5	5	5	…	5	0	2(div)	2	0	1(div)	1	0
Number of addition operations	0	0	5	5	…	5	5	0	0	2	0	0	1

Table 1b: the number of Addition, Multiplication and division operations on the tiers N_1+ 8 to N_1+ 15

Tier	N_1+8	N_1+9	N_1+10	N_1+11	N_1+12	N_1+13	N_1+14	N_1+15	…
Number of mul/div operations	5	5	5	5	5	5	5	5	…
Number of addition operations	0	5	5	5	0	5	5	5	…

Table 1c: the number of addition, multiplication and division operations on the tiers N_1+ 3m +6 to N_1+ 3m + 12

Tier	N_1+3m+6	N_1+3m+7	N_1+3m+8	N_1+3m+9	N_1+3m+10	N_1+3m+11	N_1+3m+12
Number of mul/div operations	5	0	5	0	0	0	1(div)
Number of addition operations	5	5	5	2	2	1	0

Table 2: The numerical characteristics of the subunit

N_1	15.000	30.000	100.000	1000.000	15.000	30.000	100.000	1000.000
N_2	15.000	30.000	100.000	1000.000	15.000	30.000	100.000	1000.000
N_{add}	113.000	218.000	708.000	7008.000	113.000	218.000	708.000	7008.000
N_{mul}	27.000	247.000	807.000	8007.000	127.000	247.000	807.000	8007.000
N	36.000	60.000	172.000	1612.000	36.000	60.000	172.000	1612.000
n_{mul}	29.000	53.000	165.000	1605.000	29.000	53.000	165.000	1605.000
n_{add}	7.000	7.000	7.000	7.000	7.000	7.000	7.000	7.000
τ_{add}/τ	0.100	0.100	0.100	0.100	0.050	0.050	0.050	0.050
k_{mul}	0.872	0.930	0.977	0.998	0.874	0.931	0.978	0.998
k_{add}	0.380	0.406	0.427	0.436	0.193	0.204	0.214	0.218
Tser	1383.000	2688.000	8778.000	87077.000	2653.000	5158.000	16848.000	167148.000
$T_{subunit}$	297.000	537.000	1657.000	16057.000	587.000	1067.000	3307.000	32107.000
k_{per}	4.657	5.006	5.298	5.423	4.520	4.834	5.095	5.206

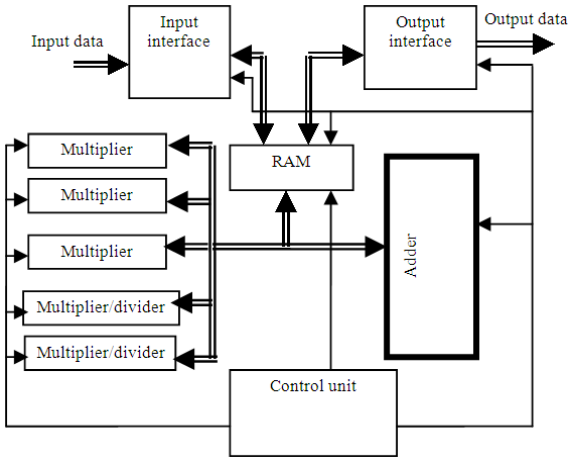

Fig. 2: The parallel subunit

The numerical characteristics of the parallel subunit, that performs on each algorithm tier not less than five multiplication operations or two division operations and five addition operations with different values of N_1,N_2 and τ_{add}/τ are shown in Table 2.

The first seven rows in Table 2 show N_{add}, N_{mul} numbers of addition and multiplication or division operations, n total number of algorithm tiers, n_{mul} number of algorithm tiers with addition operations corresponding to N_1, N2-numbers of learning and testing subsequence points.

The five rows represent k_{mul}, k_{add}-the load coefficients of multipliers and adders, also T_{ser}, $T_{SUBUNIT}$ relatively algorithm execution time in serial and parallel computing devices and k_{per}-performance coefficient of the designed subunit in comparison with serial devices.

From the coefficients values K_{mul}, K_{add}, K_{per}, shown in Table 2 with different τ_{add}/τ relations, various N_1, N_2, numbers, it is clear that multipliers play the most important role in maximizing the subunit performance. The adder is loaded less than half time with $\tau_{add}/\tau = 0.1$ and in 20% greater with $\tau_{add}/\tau = 0.05$, so increasing the number of parallel multipliers to ten or to twenty, the needed 10 or 20 addition operations on the time τ, could be performed by one adder, the final structure of the parallel subunit is shown in Fig. 2.

CONCLUSION

Structural synthesis technique of specialized computing devices by carrying out parallel calculations is developed at hardware-software realization of self-organizing algorithms at a level of separate mathematical models. This technique can be used for synthesis specialized computing devices for any known functional-oriented computing systems for data processing by a group method of data handling.

REFERENCES

Dmitrienko, V.D., N.I. Korsonov, S.Y. Leonov and I.M. Shehabat, 1998. Self-Organizing Algorithms and K-Type Dynamic Models. Nauka, ISBN: 5-02-015253-6, pp: 245.

Dolenko, S.A., Y. V. Orlov and I. G. Persiantsev, 1996. Practical implementation and use of Group Method of Data Handling (GMDH): Prospects and problems. Proceeding of the 2nd International Conference on Adaptive Computing in Engineering Design and Control, Mar. 1996, University of Plymouth, UK., pp: 153-159.

Farlow, S.J., 1984. Self-Organizing Method in Modeling: GMDH-Type Algorithms. In: Statistics, Textbooks and Monographs, Barron, R. *et al.* (Eds.). Marcel Dekker Inc., ISBN: 0-8247-7161-3 pp: 1-66.

Gex, A., 1971. Multiplier-divider cellular array. Elect. Lett., 7: 442-444.

Ivakhnenko, A.G., 1971. Polynomial theory of complex systems. IEEE Trans. Syst. Man Cybern., SMC-1: 364-378.

Ivakhnenko, A.G., G.A. Ivakhnenko and J.A. Muller, 1994. Self-organization of neuronets with active neurons. Patt. Recogn. Image Anal., 4: 177-188.

Ivakhnenko, A.G. and G. A. Ivakhnenko, 1995. The review of problems solved by algorithms of the Group Method of Data Handling (GMDH). Patt. Recog. Image Anal., 5: 527-535.

Kung, S.Y., 1991. VLSI Array Processors. Prentice-Hall, Inc., ISBN: 13: 978-0139427497, pp: 600.

Madala, H. and A.G. Ivakhnenko, 1994. Inductive Learning Algorithms for Complex Systems Modeling. CRC Press Inc., Boca Raton, pp: 384.

Saveliev, A.Y., 1987. Applied Theory of Digital Computers. High School, Moscow, pp: 272.

Veshinchouk, I.M. and N.V. Cherkasky, 1990. Algorithmetic Operational Devices and Super Computers. Tekhnika, pp: 197.

Voevodin, V.V., 1986. Mathematical Models and Method Parallel Processes, Nauka, pp: 296.