Algorithms and software for modules of automatic neurocomputer control of smart electromechanical system such as hexapods

V A Romanchuk
Department of Informatics and Computer Technology, Ryazan State University named for S.Yesenin, Ryazan, Russia
E-mail: v.romanchuk@rsu.edu.ru

Abstract. The article discusses issues in developing algorithms and software for specialized computing devices based on neuroprocessors, to be used in automatic control of electric-mechanical system modules (in this case study, a hexapod) in a mode that is close to real-time. The practical implementation employed an NM6406 neuroprocessor based on an MC 51.03 tool module and an MB 77.07 microcomputer, developed by the Module Research Centre.

1. Introduction
Controlling dynamic objects involves two modes: operation and standby, due to the rigid necessity of high-speed processing of great amounts of data. In fact, no computing control system today can ensure such processing of data at a set tempo, when such input is fed from multiple sensors; therefore, computing operations have to be parallelized into additionally connected specialized computational devices (SCDs). Such task sharing supports the unsteady condition, without slowing down the basic computing process. Two primary tasks require real-time problem-solving: the computation of control action and integration of a differential equation system. Transforming the differential system into a system of algebraic equations (vector-matrix) is convenient, as far as it enables its implementation in digital computing, namely in SCDs: their processor modules will simultaneously perform accumulative multiplication actions with multiple operands.

To develop and implement the corresponding mathematical content, algorithms and software, we propose to activate a selected set of hardware tools: neuroprocessor devices based on special-class processors (Generation 6), widely used in data processing. The choice of neuroprocessors considered such parameters as the speed of parallel processing, availability of accumulative addition, and high energy efficiency [1, 2]. In our case, we chose the NeuroMatrix 640x processor family, manufactured by the Module Research and Development Centre (Moscow, Russia) [3].

The input data parameters for development of algorithms and software are as follows:
- definition of digital signals from sensors;
- description of algorithms for generation of management signals;
- a mathematical model implemented in a SIMULINK module model of the MATLAB software package.

The source data of algorithms developed include the following:
- definition of sensor signals received after conversion of analog signals from the hexapod into digital form (via an analog-digital converter).
The output data of algorithms developed include the following:
- control signals, converted into analog (via a digital-analog converter);
- auxiliary and maintenance signals.

2. **Mathematical formalization of SCD operational tasks**

The flowchart of an SCD is shown in figure 1. The device’s functional purpose is to calculate optimal parameter values for a shared controller and to calculate quotients \((k_1, k_2, k_3)\) of PID controllers of the hexapod’s movement engine, to ensure seizure-free and smooth motion of the hexapod’s platform.

The chart uses the following symbols:

- \(\Phi, \overline{\Phi}, S, \overline{S}, G, \overline{G}, \Phi\) — operational matrixes received from the parameter matrixes of the hexapod and from those of the correction filters;
- \(P^*, P\) — parameter matrixes of correction associations (6x100);
- \(\alpha^* (t)\) — program vector function (18x1);
- \(Y(t)\) — vector function of observation (signals received from the measuring system) (30x1);
- \(X(t), X^*(t)\) — vector function of the system’s current state and of its programmed state at Step \(i\) of its control (100x1);
- \(X(t + h), X^*(t + h)\) — the same at Step \((i + 1)\) (100x1);
- \(U(t)\) and \(U(t + h)\) — vector of control influences at Steps \(i\) and \((i + 1)\) of control (30x1);
- \(h\) — step of integration (~0.001);
- \(\theta\) — deviation value of the hexapod’s focal axis from the object’s line of sight (3x1);
- \(C^*, R^*, R\) — permanent matrixes sized (3x3, 3x100, 3x100).

We can say that the basis of a neuroprocessor unit is an emulation of a formal neuron, which is mathematically a total of operations of addition and multiplication, i.e., "accumulative multiplication".

From the hexapod control algorithm, it becomes clear that the main procedure in the SCD is the multiplications of the matrixes by the vector:

\[
\Phi \times X = \begin{bmatrix}
\Phi_{11} & \Phi_{12} & \cdots & \Phi_{1n} \\
\Phi_{21} & \Phi_{22} & \cdots & \Phi_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
\Phi_{m1} & \Phi_{m2} & \cdots & \Phi_{mn}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix}
= \begin{bmatrix}
\Phi_{11}x_1 + \Phi_{12}x_2 + \cdots + \Phi_{1n}x_n \\
\Phi_{21}x_1 + \Phi_{22}x_2 + \cdots + \Phi_{2n}x_n \\
\vdots \\
\Phi_{m1}x_1 + \Phi_{m2}x_2 + \cdots + \Phi_{mn}x_n
\end{bmatrix}
\]
One of the vector result components is the sum of the item products of matrix lines by the corresponding items of the vector-column. This sequence of operations should be repeated as many times as there are lines in a matrix. To calculate the product of matrixes by the vector, it is necessary to perform multiplication of operations \( T_{\text{vcm}} = m \times n \), where \( m \) is the number of line in the matrixes; \( n \) is the number of columns, and \( N_{\text{vcm}} \) is the number of addition operations, defined by the formula: \( N_{\text{vcm}} = m \times (n - 1) \) addition operations, or approximately 3,000 pairs of "multiplication – addition" operations.

3. Development of algorithm support of the SCD

As the conducted experimental research employed an NM6406 neuroprocessor based on an \( MC \ 51.03 \) toolkit and an \( MB \ 77.07 \) microcomputer, we have to say that parallelizing is made effective due to hardware support of the vector-matrix multiplication operation in the NeuroMatrix NM6406 neuroprocessor. All arithmetic computing directly related to computation of control influences is performed on a vector coprocessor. As the vector node allows handling data with variable width, it is convenient to allocate 32 bits to the integral part and 32 bits to the real part.

For development of algorithms and of program code, subprograms of the functional items (of the mathematical model of a hexapod-controlling SCD) were implemented [1]. So, the chief operation of the SCD consists in multiplication of matrixes which can be split into a couple of elementary operations «multiplication – addition» (accumulative multiplication). For this operation, a vector coprocessor is used, i.e., a shared matrix node, to perform operations of accumulative multiplication with arithmetic and logical operations, masking, and enabling vectors and matrixes.

The accumulative multiplication is performed in the operational matrix of the neuroprocessor [4]:

\[
Z_i = Y_i + \sum (X_iW_{ij}); i = 1, m; j = 1, n,
\]

where \( X_i \) is a data item sent to the vector coprocessor input;
\( Y_i \) is the subsum accumulated at the previous stage of weighted addition, or the remainder from the previous operation;
\( W_{ij} \) – the weight quotient, stored in a dedicated cell of the operating matrix of the processor;
\( m \) – the number of columns in the operative matrix of the processor;
\( n \) – the number of lines in the operative matrix of the processor.

Data input at \( X \) should be multiplied by the number of the matrix cells, they are further added column-wise, which means that in case of overflow there occurs a loss of sign bits. Data from the \( Y \) input are added item-wise with the multiplication product, resulting from operations with the data of the \( X \) input.

To perform weighted addition, we must first export \( W_{ij} \) weighted quotients into the matrix.

Partitioning of matrixes into lines is defined in the \( sb2 \) operational registry. It provides preliminary partitions of 64-bit input data words that are sent to the \( Y \) input. Operational matrix partitioning into column is set in \( nh2 \) operational registry. The same registry defines partitioning of 64-bit data at the \( Y \) input and is also preliminarily entered as a word in the registry. The same registry defines the computing bit rate that will move to the AFIFO buffer registry in the end. Thus, input data (operands) and output values are packed into 64-bit words during partitioning. All operations in the matrix are performed as parallel within one stroke.

The accumulator of two or more operands is implemented via a vector unit of the processor and operations of weighted addition (Figure 1) with void multiplication, i.e., a "single" shadow and operational matrix (where each 64x32 bit is filled with single bits). The effectiveness of this operation depends on the bit rate of input data, as the computing speed is increased via matrix partitioning into rows and columns, because work is performed simultaneously on several operands. One important property of a vector processor is its ability to work with operands of various length.
Activation functions are implemented with the help of the processor’s vector unit and operations of weighted addition (Figure 1). Subprograms of two activation function types are implemented: the threshold function and the overflow function. In activation units computation is performed on packed data words. Activation units allow using activation functions on all items of a packed word simultaneously. The main role in controlling the activation functions is that of the $f_{1cr}$ and $f_{2cr}$ registries. The activation units are located between the masking device and the operational matrix or the vector calculator. The activation function may be performed on input $X$ or $Y$ data, or on both inputs.

All elemental basic operators are implemented in macros form in the neuroassembler language.

4. Development of software for SCD operation

For work convenience with various source data a task model was implemented, to automatically control electric and mechanical systems in the developed Visual Programming subsystem of the NP Studio software platform (figure 2).

![Figure 2. Task model of automatic control of electric and mechanical system modules in the developed Visual Programming subsystem of the NP Studio software platform.](image)

The chief item in the subsystem is the notion of a functional unit, connectable to other functional units to perform certain functions.

The workspace of the software application is divided into functional parts:

1. The area of functional units, instance of which may be transferred to the modeling workspace with the drag-and-drop function. The same area contains control items for visualization and deletion of connections between items. When added to the workspace, each item gets a unique ID, which is a concatenation of the item category, the item number within the category and the index number of the item, e.g., "O2.9" defines an item from category Output Signals, Number, with "9" as its unique ID.

2. The workspace (which, in its turn, can be represented as visual items and connections between them, or as a matrix of connections between items (Figure 3).
Figure 3. Task model of automatic control of electric and mechanical system modules in the developed Visual Programming subsystem as matrix connections.

5. Conclusion

The article proposes algorithms and software for a specialized computational device based on neuroprocessors, to be used in automatic control of electric-mechanical systems (in this case study, a hexapod) in a mode that is close to real-time, employing a novel computing device, an NM6406 neuroprocessor, developed by the Module Research Centre. The research proves effectiveness of the neuroprocessor in performing tasks of automatic control of electromechanical system modules, through parallel processing and consequent speed gain, use of the accumulative addition operation, and low power consumption.

References

[1] Ruchkin V, Romanchuk V and Sulitsa R 2013 Clustering, restorability and designing of embedded computer system based on neuroprocessors Proceedings of the 2nd Mediterranean conference on embedded computing (Montenegro: Budva) 58-62

[2] Kolinummo P, PASI Pulkkinen P, Hämäläinen T and Saarinen J 2000 Neural Processing Lett. 12 171-82

[3] Izeboudjen N, Larbes C and Farah A 2014 Artif Intell Rev 41 491–534

[4] Pande S, Morgan F, Cawley S, Bruinjes T, Smit G, McGinley D, Carrillo S, Harkin J and McDaid L 2013 Neural Process Lett. 38 131–153