COMPUTATIONAL PROBLEMS OF ELECTRICAL ENGINEERING

Vol. 2, No. 2, 2012

ACCELERATION OF THE PARAMETERS IDENTIFICATION FOR DYNAMIC MODELS CONSTRUCTION USING PARALLELIZATION

Liliana Byczkowska-Lipińska[1], Petro Stakhiv[2], Yuriy Kozak[2]

[1]Technical University of Łódz, Poland; [2]Lviv Polytechnic National University, Ukraine

lilip@ics.p.lodz.pl, petro.stakhiv@p.lodz.pl , ykozak@mail.ru

Abstract: Construction of mathematical models for nonlinear dynamical systems using optimization requires significant computation efforts to solve the optimization task. This makes it reasonable to use parallelization of calculations for optimization task solving, especially taking into account current tendency for increasing the number of CPU cores in a single chip. The effectiveness of particular parallel implementation of optimization process is the subject of investigation in this paper.

Key words: macromodels, optimization, parallelization.

1. Introduction

The complexity of dynamical systems to be designed and analysed is constantly increasing. Now the systems of such a type include components of different nature thus they should be described by mathematical models of different types. So, mathematical description of these systems becomes a complex task, which requires significant computation resources. One more problem is a complexity of available mathematical models of some components included into the system under design or analysis. These problems lead to the necessity of developing an universal approach intended for mathematical description of single components as well as the entire system with minimal required computational resources.

In such conditions the usage of macromodels allows significant reducing of required computation efforts because it makes it possible to ignore effects not important for particular analysis. Macromodels can be used to describe both single components and subsystems of significant size including elements of different nature. They can replace complex models in a simulation process as well as describe elements for which regular models are not available.

Considerable number of existing approaches to macromodel constructions [4, 5] can't be considered as the universal approaches as they apply many restrictions relatively to the modelled object and required input information, which is not always available

An alternative approach for the macromodel construction which does not have such drawbacks is the usage of optimization. This approach can be used to construct macromodels in any mathematical form described by a limited set of unknown coefficients. It also does not apply any constraints to the required input information except the obvious requirement to describe the object fully enough.

Also the optimization allows eliminating computational problems related to the ill-conditioned problems which often appear in macromodel identification.

The main problem which complicates the usage of optimization approach for the macromodels creation is a complexity of optimization task, which requires the application of significant computational resources. There are techniques to simplify this task [2]. One of them is to parallelize calculations related to optimization task solving [1]. The analysis of effectiveness of such parallelization is a subject of presented paper.

2. Macromodel construction with optimization

Let's consider the process of macromodel construction based on the "black box" approach which allows us to ignore a real internal structure of the object and, as a result to construct more simple models.

The object for which the macromodel will be constructed is shown schematically in the Fig. 1.

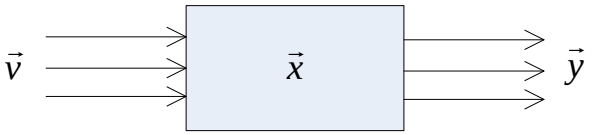

Fig. 1. An object for the macromodel construction.

In the Fig. 1 vector $\vec{v}$ describes input values, vector $\vec{y}$ are output values, vector $\vec{x}$ reflects values corresponding to the internal state of the object. The goal of the research is to find an operator which will allow calculating output values of the object $\vec{y}$ using known input values $\vec{v}$ and initial values of the internal state of the object $\vec{x}$.

We use a discrete state equations form of model mathematical representation:

$$\begin{cases} \vec{x}^{(k+1)} = F\vec{x}^{(k)} + G\vec{v}^{(k)} + \vec{\Phi}\left(\vec{x}^{(k)}, \vec{v}^{(k)}\right) \\ \vec{y}^{(k+1)} = C\vec{x}^{(k+1)} + D\vec{v}^{(k+1)} \end{cases} \quad (1)$$

where F, G, C, D are matrices of the model coefficients; Φ is some nonlinear vector-function of many arguments; k is a discrete index.

The discrete form of representation is best suited for computer calculations because it allows omitting approximation of input data arrays. The state form of equations is also determined by convenience of further usage of the model as a component of a dynamic system containing a greater number of elements.

Let us consider some object for which the model should be constructed using the form (1) or any other form which can be described by a limited set of unknown coefficients. Input information about the object can be presented in the form of an array of it's transient characteristics $\left\{v_i^{(k)}; y_i^{(k)}\right\}$, where k is a discrete index, i is an index of transient characteristic. Let us introduce some goal function representing a measure of inaccuracy with which our model describes the object. Its simplest mathematical representation can take a form of the root-mean square deviation:

$$Q\left(\vec{\lambda}\right) = \sum_i \sum_k \left(\tilde{\vec{y}}_i^{(k)} - \vec{y}_i^{(k)} \right)^2 \qquad (2)$$

where $\tilde{\vec{y}}_i^{(k)}$ is the object response calculated using it's model, $\vec{\lambda}$ is a set of unknown coefficients. For the model form (1) the vector $\vec{\lambda}$ includes elements of matrices F, G, C, D and coefficients of vector-function Φ. The set of the model coefficients $\vec{\lambda}^*$ can be considered to be optimal if the goal function (2) approaches its minimum. Thus, the model coefficients determination can be carried out by finding the global minimum point of the function (2).

The proposed approach is suitable for the coefficients identification when the model is presented in any form and it does not apply any special restriction concerning the input information used for the coefficients identification. This makes it possible to use proposed approach effectively as a universal method for coefficients identification during the creation of the dynamical models.

Taking into account the complexity of an optimization task (in case the goal function under research is significantly non-linear with many unknown parameters and considerable difference at the level of its dependency on different coefficients), an attention should be paid to the selection of an optimization algorithm. As practice shows, the stochastic optimization algorithms are the most suitable ones for such optimization tasks. They are also less sensitive to the great amount of local minimums produced by rounding errors and considerable calculation efforts.

3. Parallel implementation of optimization algorithm

Most optimization algorithms, including Rastrigin's director cone method [3] with adaptive algorithms of step size and cone aperture angle, used by authors, do several (sometimes dozens) goal function calculations to perform a single iteration of algorithm. The set of points (values of vector $\vec{\lambda}$) for which a goal function needs to be calculated is generally known at the very beginning of the iteration (this is true for both deterministic and stochastic algorithms). This permits to do the calculation of the goal function for different points using several CPUs simultaneously.

The practical implementation of this method can look like shown in Fig. 2:

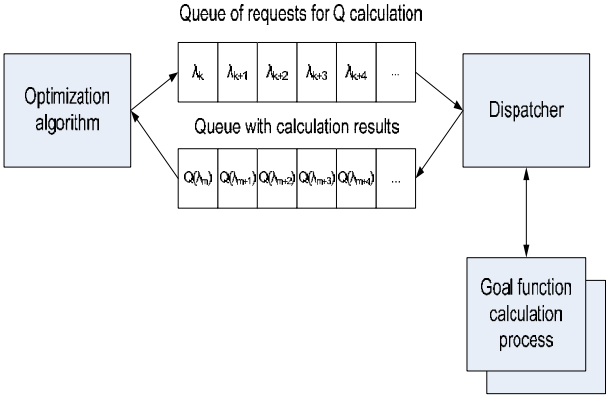

Fig. 2. Structure of software for the optimization algorithm with the parallel calculation of a goal function for different points.

The optimization algorithm itself in this implementation is executed in a separate thread. It forms a list of points for which the goal function should be calculated and puts them into a queue. After all points needed for current iteration are known, the process waits for goal function calculation results. When values of goal function for all added points are calculated, the algorithm process starts performing, analyses the results and switches to the next iteration.

Goal function calculations for needed points are done in a set of threads, which are coordinated by a dispatcher and can be executed on a same computer or other computers. The limitation here is the proportion of time needed for a single calculation of goal function and communication time between the dispatcher and the calculation process. Communication time increases in case when several computers are used, but for complex goal functions which are calculated based on tens of thousands of samples communication time can be negligible comparing with the single calculation of goal function even when communication is done via the internet.

Key points of this approach are:

- Optimization algorithm thread does not perform goal function calculations itself. It only prepares the set of points for which the goal function should be calculated;
- Optimization algorithm determines all points needed for the iteration and only then waits for the goal function calculation results.

Here it should be noted that this parallelization approach is suitable for many but not for all optimization algorithms. In particular it can be used for optimization algorithms satisfying the following requirements:

1. An algorithm should calculate the goal function several times during one iteration, because the effectiveness of parallelization grows with the increasing of the number of goal function calculations at one iteration

2. The set of coefficients $\vec{\lambda}$ for all points where goal function is calculated during one iteration, should not depend on the result of the goal function calculation in other points at same iteration.

An example of the algorithm, which satisfies these requirements, is Rastrigin's director cone method. Another example of algorithm, which cannot be used with considered parallelization, is the Nelder–Mead method [6]. Also there are algorithms for which mentioned criteria are satisfied partially. For such algorithms effective usage of the proposed parallelization technique requires changing over parallel and sequential calculations of a goal function during the optimization process. An example of such algorithm is Gradient descent method [4], which does not satisfy the second criterion because after the determination of a gradient direction the algorithm requires the goal function to be calculated one more time, and the position of a point where it should be calculated depends on results of the goal function calculations already conducted at the same iteration.

Total CPU time needed for algorithm with parallelization will be greater than CPU time without parallelization. This is caused by the need to switch between threads. This includes both time needed to perform thread switching and delays caused by thread synchronization. One more potential problem is the usage of software libraries not designed for or not effective in parallel algorithms. For example this problem was faced by authors with a memory manager in Delphi 5.0 as it works in a single thread.

4. Theoretical analysis of parallelization effectiveness

Because the optimization algorithm in this implementation works in a single thread, parallelization efficiency will depend on the number of discretes used for the goal function calculation.

Time required to perform one iteration without additional time needed for parallelization can be estimated next way:

$$T(m,M,N) = T_A(m) + T_Q \cdot \frac{m \cdot M}{N} \qquad (3)$$

$$T_A(m) = T'_A + T''_A \cdot m$$

where m is the number of points for which the goal function is calculated at one iteration; M is the number of discretes used for the goal function calculation; N is the number of CPUs; $T_A(m)$ is a time needed for one CPU for one iteration of the algorithm (making decisions, the determination of points for which the goal function needs to be calculated. This time depends on the number of points for which the goal function is calculated during one iteration); T_Q is a time needed for one CPU to account a single discrete in the goal function calculation.

From (3) we can get a maximum limit for a parallelization factor. Without additional time needed for parallelization itself the limit can be estimated by a next formula:

$$\eta(m,M,N) = \frac{T(m,M,1)}{T(m,M,N)} = \frac{T_A(m) + T_Q \cdot m \cdot M}{T_A(m) + \frac{T_Q \cdot m \cdot M}{N}} \qquad (4)$$

To account additional time needed for parallelization we need to add an additional element to the denominator:

$$\eta(m,M,N) = \frac{T_A(m) + T_Q \cdot m \cdot M}{T_A(m) + \frac{T_Q \cdot m \cdot M}{N} + T_P(N) \cdot m} = \frac{\frac{T_A(m)}{m} + T_Q \cdot M}{\frac{T_A(m)}{m} + \frac{T_Q \cdot M}{N} + T_P(N)} \qquad (5)$$

At practice we always have $T_Q << T_P(N)$ for $N > 1$. In this case we can do two more important estimations of the parallelization factor:

- For small values of M:

$$\eta(m,M,N) \approx \frac{\frac{T_A(m)}{m} + T_Q \cdot M}{\frac{T_A(m)}{m} + T_P(N)} < 1 \qquad (6)$$

- For big values of M (when $T_Q \cdot M >> T_P(N)$ and $T_Q \cdot M >> T_A(m)$):

$$\eta(m,M,N) \approx \frac{T_Q \cdot M}{\frac{T_Q \cdot M}{N}} = N \qquad (7)$$

So the parallelization will be not effective for small number of discretes in input information and will be effective for a big number of discretes.

5. Experimental verification of effectiveness of parallelization

To measure the real effectiveness of the considered parallel implementation of the optimization process authors compared average time needed to conduct 400 iterations of optimization algorithm (the goal function was calculated 30 times at one iteration) for different number of discretes in input information. The average was found for 3 experiments.

The results are presented in table 1, and in fig. 3 and 4. Obtained experimental results conform to theoretical estimates (5), (6) and (7), and shows, that parallelization factor can reach its theoretical limit equal to the number of CPUs in case of great number of discretes in transient characteristics.

It should be noted, that proposed parallelization approach generally does not solve the problem of optimization task complexity, but can only speed-up the process by using all available CPU power. Other techniques should be used together with parallelization to get best results.

Proposed parallelization approach can be easily combined with other techniques of optimization task simplification. Particularly it can happen in the case when a macromodel construction is split into the set of stages [2]. The limiting factor in this case will be total computational power of all available CPU cores.

As the total performance in case of great number of discretes in input information increases with the number of used CPUs and actual execution steps done in all calculation threads are exactly the same, we can expect that considered parallelization approach can be implemented in such a way when computation power of modern graphical processors, for example NVIDIA GeForce cards, will be used. This option looks to be promising, though it is out of scope of this paper.

Table 1

Results of experimental testing of considered parallelization approach

Points in input data	1 CPU		2 CPUs			3 CPUs			4 CPUs		
	Time (sec)	Points per second	Time (sec)	Points per second	% from one CPU	Time (sec)	Points per second	% from one CPU	Time (sec)	Points per second	% from one CPU
3	1.500	800.0	5.141	233.4	29%	25.555	47.0	6%	30.089	39.9	5%
10	2.037	1964.0	4.104	974.7	50%	25.611	156.2	8%	28.458	140.6	7%
30	3.166	3789.9	3.537	3392.7	90%	23.355	513.8	14%	29.589	405.6	11%
100	7.000	5714.6	4.543	8805.4	154%	15.704	2547.2	45%	21.547	1856.4	32%
300	17.682	6786.4	9.679	12398.4	183%	12.500	9599.7	141%	15.740	7624.0	112%
1000	54.968	7276.9	29.125	13733.9	189%	23.573	16968.3	233%	19.984	20015.7	275%
3000	172.186	6969.2	83.755	14327.6	206%	60.349	19884.3	285%	47.635	25191.6	361%

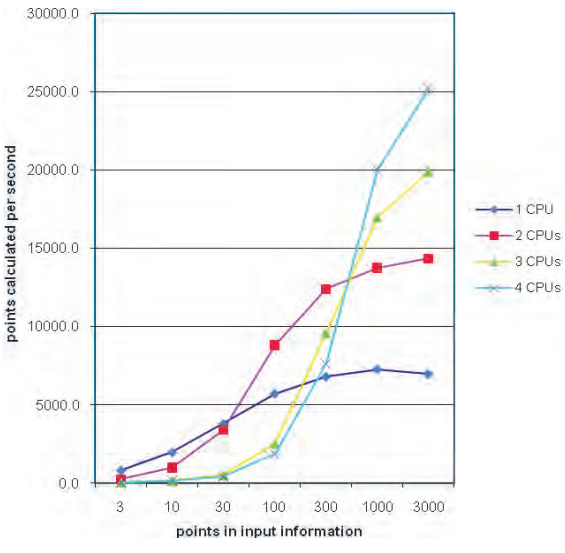

Fig. 3. Performance (number of discretes accounted in goal function calculation per second) as a function of the number of discretes in used transient characteristics.

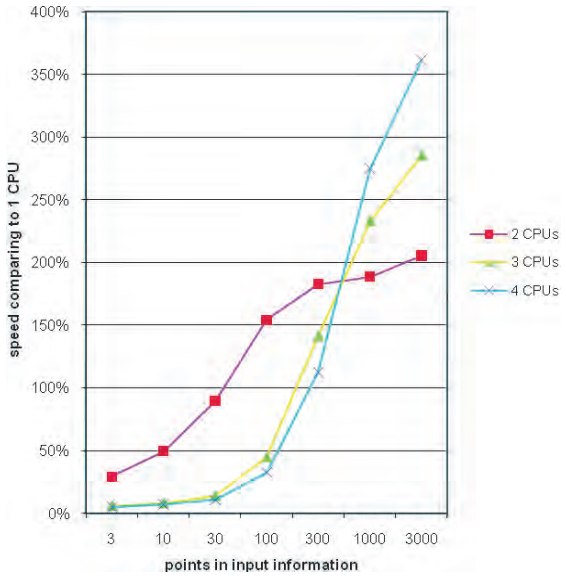

Fig. 4. Parallelization factor as a function of the number of discretes in used transient characteristics.

6. Conclusion

Theoretical estimations and experimental data shows, that the proposed parallel implementation of the optimization process used for the construction of the dynamical models allows to speed-up the macromodel construction in case when the model is built based on transient characteristics with a big number of discretes. Parallelization factor in this case almost reaches the theoretical limit equal to the number of CPUs. This allows to state, that proposed approach is highly effective in case if the model is built based on transient characteristics with a big number of discretes.

The parallelization of the calculation can be combined with other techniques of optimization task simplification, for example it can be done, as it was mentioned above, when the macromodel construction is split into a set of stages.

One more advantage of the considered parallelization approach is its potential ability to be implemented for usage of the high computation power of modern graphical processors.

References

1. L. Byczkowska-Lipinska, P. Stakhiv, Y. Kozak, Parallelization of Calculations in Construction of Mathematical Models using Optimization // In Proc. XIII International Conference “System Modelling and Control” SMC 2009. – Zakopane, Poland. – 2009.

2. P. Stakhiv, B. Melnyk, Y. Kozak, Simplification of Optimization Process during Mathematical Models Creation // Przegląd Elektrotechniczny – № 12. – 2008. – P. 281-283.

3. Y. Kozak, Modification of Rastrigin Method of Direction Cone // Elektronika i svyaz. Special edition on Problems and Physical and Biomedical Electronics. – Kyiv, Ukraine: Publishing House of Kyiv Polytechnic Institute. – 1997. – P. 424. (Ukrainian)

4. M. Avriel, Nonlinear Programming: Analysis and Methods // New York, USA: Dover Publications. – 2003.

5. L. Ljung, System Identification: Theory for the User. – New Jersey, USA: Prentice Hall – 1999. – 609 p.

6. J. Nelder, R. Mead, A Simplex Method for Function Minimization // The Computer Journal. – Oxford, UK: Oxford University Press. – № 7. – 1965. – P. 308–313.

ПРИШВИДШЕННЯ ПАРАМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ ДЛЯ ПОБУДОВИ ДИНАМІЧНИХ МОДЕЛЕЙ З ВИКОРИСТАННЯМ РОЗПАРАЛЕЛЕННЯ

Ліліана Бичковска–Ліпіньска, Петро Стахів, Юрій Козак

Побудова математичної моделі нелінійних динамічних систем з використанням оптимізації вимагає значних обчислювальних затрат для вирішення задачі оптимізації. Це робить доцільним використовувати розпаралелення обчислень для вирішення завдань оптимізації, особливо з урахуванням поточної тенденції збільшення числа процесорних ядер в одному чіпі. Ефективність зокрема паралельної реалізації процесу оптимізації є предметом дослідження в даній роботі.

Liliana Byczkowska – Ph.D., D.Sc., Professor, born in Poland, received her MSc degree from the Electrical Engineering Faculty of the Technical University of Łódź, Poland. From 1995 she worked at the Institute of Computer Science of the Technical University of Łódź, Poland, as Associate Professor. From 1996 to 2004 she worked as Vice-director for Reserarch. Since October 2004 she has been the Head of the Institute of Computer Science at the Technical University of Łódź, Poland. Several times she undertook internship both in Poland and abroad (Ukraine, France). She is the author of more than 150 publications, 4 monographs, 5 lecture series and numerous projects done for industry. Her scientific interests cover telecommunication, computer aided modeling and simulations of electromagnetic fields and vibroacoustic phenomena and their threats to the natural environment. She is the Treasurer of the Main Board of the Electromagnetic Applications Society, apart from the membership in scientific committees of different conferences.

Petro Stakhiv– Ph.D., D.Sc., Professor, born in 1948 in Lviv region, Ukraine. In 1970 he graduated from Lviv State University, Department of Physics, Ukraine, and received his M.Sc. degree in Radio Physics and Electronics. From 1970 to 1973 he was a Ph.D. student at Department of Theoretical Electro and Radio Engineering.

In 1975 he received his Ph.D. degree in theoretical electrical engineering. The theme of his Ph.D. work was «Synthesis of linear electric circuits (method of state variables)». In 1992 he received his D. Sc. degree in the same specialty after defending his doctor thesis on «Analysis of dynamic regimes in electric and electronic circuits with multiterminal elements». From 1973 to 1996 he worked as Assistant Professor, Associate Professor, Professor and Head of Department of Theoretical Electro and Radio Engineering at Lviv State University. In 1996 he began working at Lviv Polytechnic National University as Professor and Head of Department of Theoretical and General Electrical Engineering.

His scientific interests are mainly concerned with mathematical modeling and simulation of dynamic processes in electrical engineering systems, numerical methods, optimization techniques, system theory, and parallel programming.

Yuriy Kozak – Ph.D., born in 1975 in Lviv region, Ukraine, graduated from Lviv Ivan Franko State University, Department of Physics, in 1997 and received his M.Sc. degree in Radiophysics and Electronics.

From 1997 to 2000 he was a post-graduate student at the above mentioned university.

In 2002 he received Ph.D. degree in Theoretical Electrical Engineering. The theme of his Ph.D. work was «Construction of mathematical models of components of electrical engineering systems».

From 2004 to 2010 he worked as Associate Professor at the Department of Theoretical Electrical Engineering at Lviv Polytechnic National University, Ukraine. Since 2010 he has been a D.Sc. student at the same department.

His scientific interests are focused on the mathematical modeling and simulation of dynamic processes in electrsical engineering systems, numerical methods, optimization techniques, and parallel programming.

He is the author and co-author of more than 30 publications including scientific journal papers and abstracts for scientific conferences.