DOI 10.15276/imms.v11.no4.268
УДК 004.032.24:519.612.2

Informatics and Mathematical Methods in Simulation
Vol. 11 (2021), No. 4, pp. 268-277

EVALUATING THE EFFECTIVENESS OF THE IMPLEMENTATION OF COMPUTATIONAL ALGORITHMS USING THE OpenMP STANDARD FOR PARALLELIZING PROGRAMS

O.O. Zhulkovskyi, I.I. Zhulkovska, V.V. Shevchenko

Dniprovsky State Technical University, Dniprobudivska str., 2,
Kamianske, 51918, Ukraine olalzh@ukr.net

The relevance of the work lies in the need to increase the efficiency of computer modeling through the use of progressive hardware and software parallelization and synchronization of calculations on modern computers with multi-core architecture. The purpose of the work is to increase the speed of computational algorithms of Thomas algorithm by using advanced technologies for programming parallel computing. Serial and parallel algorithms for software implementation of Thomas algorithm have been developed; a comparative evaluation of the implementation efficiency (speed of execution) of these algorithms by means of an open standard for parallelizing OpenMP programs for a significant (up to 5×107) SLAE order was performed. The use of progressive programming technologies in the implementation of the indicated methods for solving SLAEs made it possible to increase the computation speed by 1,9–2,9 times. The obtained results correspond with the known literature data. In this work, the time values of the software implementation of sequential and computational algorithms of Thomas algorithm for SLAEs of significant order were obtained for sequentially and parallelized into two streams using OpenMP tools. The expediency of such parallelization for SLAE of the order of more than 2,5×105 is shown. The practical significance of the work lies in the use of the results obtained in the numerical study of stationary and non-stationary, linear and nonlinear processes in mathematical modeling problems, where the multiple solution of a significant order SLAE is the most resource-intensive stage.

Keywords: a computational algorithm, SLAE numerical solution, Thomas algorithm, speed, parallel computing, OpenMP, speeding up the computations.

Introduction

The rapid development of computer technology, including Improving the PC architecture through the use of ultrafast multi-core processors, increasing cache and system memory, etc. serves as a constant incentive for the synchronous development of software corresponding to these requirements. In addition, the modern tasks of computer modeling, requiring the processing of colossal amounts of data, are ahead of the performance capabilities of single-processor (single-core) computers. The aforementioned served as an impetus for increasing the efficiency of computer modeling and the associated computational experiment through the use of rapidly developing parallel computing technologies.

Thus, the researchers are facing the urgent task of improving the efficiency of computer modeling by increasing the speed of computational algorithms by using modern hardware and software parallelization and synchronization of calculations.

The object of research is the Thomas Algorithm or the Tridiagonal Matrix Algorithm.

The subject of the research is the means of increasing the efficiency (increasing speed) of various modifications of Thomas algorithm by their implementation on computers with a multicore architecture.

The purpose of the work is to increase the efficiency of computer modeling by increasing the speed of computational algorithms by using advanced technologies for programming parallel computing and their implementation on modern computers with multi-core architecture.

To achieve this goal, the following tasks are set before work:

– develop sequential and parallelized algorithms for software implementation of Thomas algorithm;

– perform a comparative assessment of the implementation efficiency (speed of execution) of these algorithms, including using OpenMP (Open Multi-Processing) open standard tools for a significant order of SLAE in IDE (Integrated Development Environment) MVS (Microsoft Visual Studio) C ++;

– develop recommendations regarding the appropriateness of using a parallel algorithm of Thomas algorithm in the numerical study of all kinds of processes in mathematical modeling problems.

Formal problem statement. As you know, most mathematical models are represented by systems of linear and nonlinear differential equations, the basis of the methods for solving which is solving SLAE (Systems of Linear Algebraic Equations).

Among the widespread computational algorithms used in the numerical solution of SLAEs, the most widely used is Thomas algorithm, which is a special case of the Gauss method with sequential exclusion of unknowns and used to solve systems of equations with a three-diagonal matrix [1]. Matrices of this type also arise in solving spline interpolation problems [2].

The main objective of the work is to further develop approaches to improve the efficiency of computer modeling using parallel computational algorithms of Thomas algorithm by implementing them on computers with a multi-core architecture.

In this paper, we set the task of comparatively evaluating the efficiency (speed of execution) of a serial Thomas algorithm parallel and parallelized into two streams using the open standard OpenMP algorithm for a significant (up to 5×10^7) SLAE order.

Literature review

One of the most common methods for classifying computer architectures is Flynn's taxonomy systematics, in which the main attention when analyzing the architecture of computer systems is paid to the methods of interaction between sequences (streams) of executed commands and processed data [3].

This classification uses two concepts to build parallel organizations — the instruction stream and the data stream [4].

Depending on the multiplicity of these flows, Flynn proposed the following four classes of organization of architectures [5].

1. SISD (Single Instruction, Single Data) is a traditional von Neumann architecture computer with one processor, executing one command after another in succession, working with one data stream. This type includes pipelined, super-scalar and VLIW (Very Long Instruction Word) processors.

2. SIMD (Single Instruction, Multiple Data) is one of the most common types of parallel computers. This class includes vector processors, ordinary modern processors, when they execute vector extension commands, matrix processors. Here, one processor loads one command, the data set for them, and performs the operation described in this command on the entire data set at the same time.

3. MISD (Multiple Instruction, Single Data) is, in fact, a hypothetical class, since real systems of representatives of this type do not yet exist. Some researchers attribute to it conveyor computers.

4. MIMD (Multiple Instruction, Multiple Data) is another common type of parallel computer, including multiprocessor systems, where processors process multiple data streams. This, as a rule, includes traditional multi-processor machines, multi-core and multi-threaded processors, as well as computer clusters.

All modern advanced processors, both general and special purpose, fall into the MIMD class [5]. They simultaneously execute several independent threads of instructions at once, providing hardware parallelism.

Concurrency is a complex problem in solving interdisciplinary direct and inverse super-tasks related to mathematical modeling, which is currently the main tool for obtaining new fundamental knowledge and optimizing industrial production [6]. Programming parallel systems is moving forward rapidly, introducing parallelism wherever performance matters. These systems provide all models with their own set of computing environment of the enterprise, including application scalability, increased resource utilization, faster execution time [7]. Therefore, the use of parallel programming is of considerable interest here.

Today, there are a number of high-performance specialized BLAS (Basic Linear Algebra Subprograms) libraries for optimizing and accessing mathematical calculations. BLAS libraries, such as Intel MKL (Math Kernel Library), AMD (Advanced Micro Devices, Inc.) CML (Core Math Library), or CUBLAS (CUda Basic Linear Algebra Subroutines) for NVIDIA GPUs, are configured for their own architecture. All vendors optimize BLAS code specifically for their base equipment to achieve maximum performance [8].

For scientific, engineering and financial calculations requiring maximum performance, the most popular is the Intel MKL library of optimized and parallelized mathematical procedures, compatible with advanced development environments and compilers.

Of greatest interest for the present work is the well-known [9] similar study of the performance of Thomas algorithm, carried out using the Intel MKL library.

The HPC (The High Performance Computing) community has developed a wide variety of parallel programming models to simplify the expression of the required levels of parallelism for using hardware capabilities. So, to provide parallel computing in C ++, which is currently one of the most popular and popular universal object-oriented languages, there are several software APIs (Application Programming Interface) [10]: OpenMP, Cilk Plus, C ++ 11, POSIX threads (PThreads), Intel TBB, OpenCL, Microsoft PPL (Parallel Patterns Library), etc. Each of them has its own unique set of features and benefits. Also, these tools have certain functionalities implemented in various interfaces. For example, most of them support both data parallelism and task parallelism patterns for the CPU.

The most functional and accessible OpenMP interface [11], considered in this study, offers a simple mechanism for implementing parallel computing in applications using multithreading, in which the «master» thread creates a set of «slave» flows and the task is distributed between them. The OpenMP specification is being developed by several large manufacturers of computer hardware and software, whose work is regulated by the non-profit organization OpenMP ARB (Architecture Review Board) [12]. The OpenMP standard is supported by Fortran, C/C ++ and is formulated as an API for writing portable multi-threaded applications on shared-memory multiprocessor systems (SPMD).

The OpenMP library is actively developing to date (now the standard of 2018 version 5.0 is relevant) [13]. At the same time, the MVS C ++ compiler only supports version 2.0, while GCC (GNU Compiler Collection) supports version 4.5 [14].

Portability is associated with the OpenMP programming model, which provides a platform-independent set of compiler directives, function calls and environment variables that clearly «tell» the compiler where and how to use parallelism in the application. Thus, the developer is not burdened by additional difficulties associated with the problems of creating, synchronizing, balancing the load and destroying threads.

OpenMP technology aims to ensure that the user has one version of the program for parallel and sequential execution. However, it is possible to create programs that work correctly exclusively in parallel mode, and in sequential mode give a different result. In addition, due to the accumulation of rounding errors, the result of performing calculations using a different number of threads may differ in some cases, which must be taken into account when analyzing the results of a computational experiment.

Main part

Materials and Methods. As you know, Thomas algorithm itself is an effective method for classical type architectures, unsuitable for productive implementation even on single-core superscalar systems that support parallelism at the instruction level. To parallelize a SLAE solution with a three-diagonal matrix, use its parallel substitute. Therefore, the study of the scalability efficiency of the classical Thomas algorithm, as an absolutely non-parallel algorithm, is not of practical interest. At the same time, of considerable interest is the analysis of the scalability of parallel versions of the method relative to its classical uniprocessor implementation [15].

In this case, the complexity of the sequential method, characterized by the time of solving SLAE of order n, is determined [9] by the value $10n\tau$, while for the parallel two-sided Thomas algorithm this value is $5n\tau+\delta$. It is easy to see that the theoretical acceleration of the calculation process due to parallelization cannot exceed two.

Also, according to Amdahl's law [16], used to estimate possible acceleration in parallel data processing by more than one module, the theoretical limit for acceleration due to parallelization into two streams (for this work) is equal to two. Even such an acceleration value will provide a significant increase in the efficiency of computer modeling as a whole, which, in fact, is the goal of this work.

As mentioned above, in this paper, to parallelize the software implementation of the two-sided Thomas algorithm, we used the most popular OpenMP library in mathematical calculations, which provides an accessible and multifunctional parallel computing interface.

A significant part of OpenMP functionality is implemented using compiler directives of the form:

#pragma omp <directive> [modifier[[,] modifier]…],

allowing parallel computing of different sections of code.

The number of threads in a group running in parallel can be controlled in several ways. One of them is the use of the environment variable OMP_NUM_THREADS. Another way is to call omp_set_num_threads (). Another way is to use the num_threads expression in combination with the #pragma omp parallel directive [11]. In this computational experiment, the second of the approaches was used.

During the execution of the program, the values of the computation time were recorded using the steady_clock class from the C ++ <chrono> library, which represents a monotonous clock that is not related to the system time and therefore most suitable for measuring the intervals under study.

Experiments and results. When performing this kind of research, it is assumed that the processors available in the structure of the computer are equal in performance, are equal in access to the shared memory, and the access time to the memory is the same. The above set of requirements is satisfied by the multi-core processors of modern PCs, in which each core is an almost independently functioning computing module. Thus, such studies must be carried out exclusively on systems with multi-core (from two cores) CPUs, such as, for example, in this case (Table 1).

For the purpose of conducting research, the functions of the software implementation of the classical sequential right-run method and the counter-parallel (parallel combination of left and right) Thomas algorithm parallel to two streams were developed.

In the study, the size of SLAEs was varied in the range of 1×10^5—5×10^7, and the values of the coefficients of the equation were randomly generated (taking into account the conditions for the diagonal prevalence of the matrix) into variables of standard, hardware-supported type double.

Table 1.

Computing Experiment Infrastructure

CPU		Intel Core i5-8400 (6 cores, 2.8 GHz), cache 9 MB
Memory (RAM)		Goodram DDR4 (4 GB, 2666 MHz, 21300 MB/s)×4
Operating system (OS)		Microsoft Windows 10
Development environment (IDE)		MVS C++ 15.9
Programming technology		OpenMP, v.2.0

The calculation results (Table 2) reflect the time of the software implementation of the sequential algorithms of the right-eliminations and two-sided Thomas algorithm, as well as the options for the two-sided Thomas algorithm parallel to two streams depending on the order of the SLAE.

Table 2.

Results of a computational experiment

SLAE order	Sequential Algorithms			Parallel Algorithm		
	right-eliminations ($t1$, c)	two-sided ($t2$, c)	$s1=t1/t2$	two-sided ($t3$, c)	$s2=t1/t3$	$s3=t2/t3$
100000	0,001969	0,001070	1,84019	0,001890	1,04180	0,56614
200000	0,003778	0,002325	1,62495	0,002502	1,50999	0,92926
300000	0,005328	0,003411	1,56201	0,003177	1,67705	1,07365
400000	0,007292	0,004712	1,54754	0,003646	2,00000	1,29238
500000	0,009424	0,005836	1,61480	0,004201	2,24328	1,38919
600000	0,010809	0,007042	1,53493	0,004684	2,30764	1,50342
700000	0,012564	0,008276	1,51812	0,005502	2,28353	1,50418
800000	0,014579	0,009442	1,54406	0,005996	2,43145	1,57472
900000	0,016272	0,011038	1,47418	0,006652	2,44618	1,65935
1000000	0,018275	0,011839	1,54363	0,007466	2,44776	1,58572
2500000	0,047037	0,030118	1,56176	0,017072	2,75521	1,76418
5000000	0,092932	0,058827	1,57975	0,032406	2,86774	1,81531
10000000	0,184777	0,119720	1,54341	0,064571	2,86161	1,85408
15000000	0,301776	0,198740	1,51845	0,104556	2,88626	1,90080
20000000	0,370866	0,238921	1,55225	0,127913	2,89936	1,86784
35000000	0,654413	0,418034	1,56545	0,237180	2,75914	1,76252
50000000	0,942063	0,610584	1,54289	0,367498	2,56345	1,66146

The most relevant results illustrating the novelty and significance of the work are presented in Fig. 1—3.

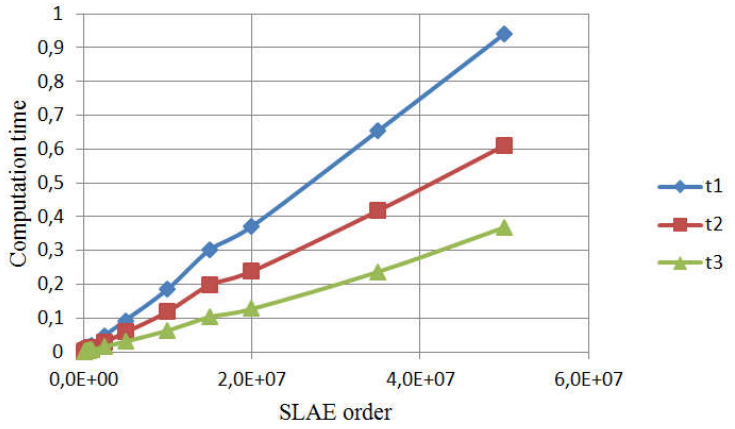

Fig. 1. The dependence of the solution time on the SLAE order in the range of 1×105—5×107 for the implementation of Thomas algorithm: t1 — right-eliminations (sequential calculation); t2 — two-sided (sequential calculation); t3 — two-sided (parallel calculation)

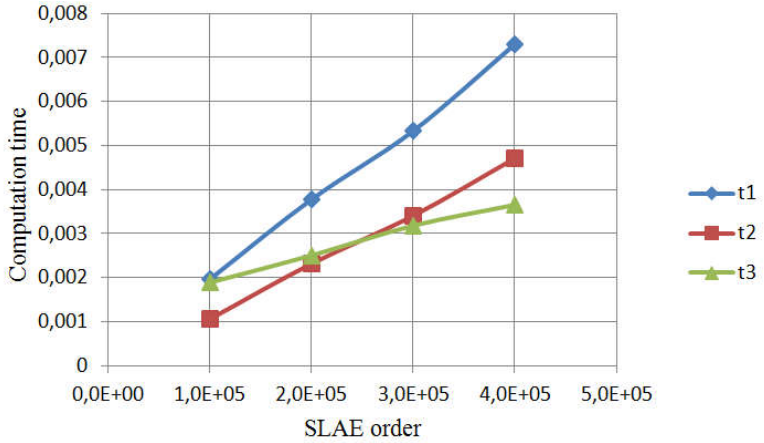

Fig. 2. The dependence of the solution time on the SLAE order in the range of 1×105—4×105 for the implementation of Thomas algorithm: t1 — right-eliminations (sequential calculation); t2 — two-sided (sequential calculation); t3 — two-sided (parallel calculation)

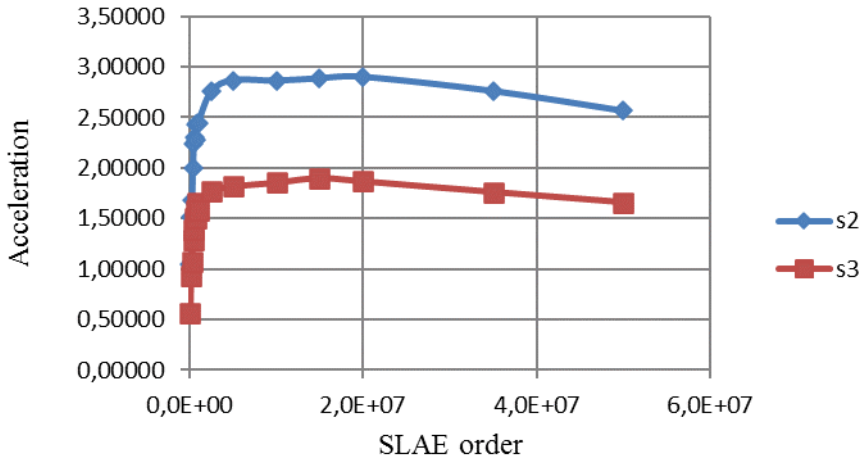

Fig. 3. The dependence of the parallel calculation acceleration on the SLAE order with respect to the sequential implementation of the right-eliminations (s2) and two-sided (s3) Thomas algorithm

Discussion

As the research results showed, the use of the two-sided Thomas algorithm (without using the OpenMP parallelization tools), instead of the traditional method of right-elimination (or left, it doesn't matter) Thomas algorithm, allows even on single-core architectures to increase the speed of the computational algorithm as a whole (Fig. 1, 2). Thus, the acceleration (s1) due to such a replacement in the studied range of changes in the order of SLAEs amounted to 1,47—1,84.

For any of the considered implementations of Thomas algorithm, the maximum computation time under the conditions of the used infrastructure of the computational experiment in the studied range of changes in the order of SLAEs did not exceed one second (Fig. 1).

Realization of parallel computations using the two-sided Thomas algorithm (using OpenMP parallelization tools) instead of the traditional one allows speeding up the computation process (s2) by 1,04—2,90 times in the same range of order changes of SLAEs (Fig. 3).

Not so unambiguous is the conclusion about the effectiveness of a parallelized algorithm (counter-running) and the use of special software tools for such parallelization compared to a serial analog in the considered range of matrix size variation. So, under the conditions of the used infrastructure of the computational experiment for SLAE of the order of less than $2{,}5\times10^5$, the execution time of parallel calculations when implementing the two-sided Thomas algorithm will be longer than for the traditional sequential calculation procedure, i.e. t3> t2 (Fig. 2) and s3 <1 (Fig. 3). For more significant values ($>2{,}5\times10^5$) of the SLAE order, the comparative calculation time becomes shorter and acquires predictable values, and the acceleration s3 exceeds unity (i.e., the process begins to accelerate) and reaches a value of 1,9.

The slowdown of calculations for SLAEs of the order of less than $2{,}5\times10^5$ is explained by the use of computer time to create computational flows (values δ are comparable or exceed the time of direct calculations) with a relative slowdown in the software implementation of the parallelized algorithm. Thus, the issue of the inappropriateness of using multi-core architectures and parallel technologies to a certain order of SLAEs is being updated.

As can be seen (Fig. 3, graph s3), the results obtained in the present study [17, 18] are fully consistent with Amdal's law, as well as with the results of similar studies by Russian scientists [9] obtained on a PC of similar architecture using Intel MKL math library.

In general, irrespective of the approach to the implementation of one or another of the studied algorithms of Thomas algorithm, with the increase in the size of the matrix, the computation time also increases almost in direct proportion. However, the acceleration of calculations due to the use of parallelized algorithms at significant orders of magnitude ($>2.5\times10^5$) SLAE always exceeds unity (Fig. 3), which means that they favor the use of such algorithms in practice when processing significant amounts of data.

Conclusion

Thus, the feasibility of using advanced hardware and software to increase the efficiency of a computational experiment by organizing parallel computing using modern multi-core architectures is proved.

The use of these programming technologies in the implementation of the common methods for solving SLAEs made it possible to increase the computation speed by 1,9—2,9 times.

However, the development of algorithms and programs oriented to multi-threaded computing requires higher qualifications of both an applied mathematician and a programmer, and the developed applications turn out to be rigidly tied to a specific computer computing

architecture. As studies have shown, overcoming these difficulties will be advisable only when developing computer models with a significant amount of arithmetic calculations.

The scientific novelty of the work lies in the further development of progressive approaches to increasing the efficiency of computer modeling using parallel computational algorithms of Thomas algorithm by their implementation on computers with multicore architecture. For the first time, the values of the software implementation time of sequential and parallelized into two streams using OpenMP computational algorithms for Thomas algorithm for a significant (up to 5×10^7) SLAE order are obtained. The expediency of such parallelization for the order of SLAEs of more than $2{,}5\times10^5$ is shown.

The practical significance of the work lies in the use of the results obtained in the numerical study of stationary and non-stationary, linear and nonlinear processes in mathematical modeling problems, where the multiple solution of SLAEs of a significant order is the most resource-intensive stage.

The prospects for further research are seen in the development of such studies in the context of a deeper parallelization of the considered computational algorithm through the use of a more laborious, but also more scalable modification of Thomas algorithm, as well as the use of other, no less advanced, software tools for implementing parallel computing processes.

References

1. Samarskij A.A. Teorija raznostnyh shem. M.: Nauka, 1989. 616 s.
2. Verzhbickij V.M. Chislennye metody (matematicheskij analiz i obyknovennye differencial'nye uravnenija). M.: Vysshaja shkola, 2001. 382 s.
3. Gergel' V.P., Strongin R.G. Osnovy parallel'nyh vychislenij dlja mnogoprocessornyh vychislitel'nyh sistem. N. Novgorod: NNGU im. N.I. Lobachevskogo, 2003. 184 s.
4. Ngoko Y., Trystram D. Revisiting Flynn's Classification: The Portfolio Approach. Euro-Par 2017: Parallel Processing Workshops, University of Santiago de Compostela, 28–29 August 2017. *Santiago de Compostela.* Springer, 2017. P. 227–239. DOI: https://doi.org/10.1007/978-3-319-75178-8_19.
5. Kudin A.V., Linjov A.V. Arhitektura i operacionnye sistemy parallel'nyh vychislitel'nyh system. N. Novgorod: NNGU im. N.I. Lobachevskogo, 2007. 73 s.
6. Il'in V. On the Parallel Strategies in Mathematical Modeling. *Parallel Computational Technologies: 11th International Conference PCT* 2017, Kazan, 3–7 April 2017. Springer-Verlag, 2017. P. 73–85. DOI: https://doi.org/10.1007/978-3-319-67035-5_6.
7. Kvasnica P., Kvasnica I. Distributed Mathematical Model Simulation on a Parallel Architecture. *Journal of Computing and Information Technology*. 2012. Vol. 20, № 2. P. 61–68. DOI:10.2498/cit.1001771.
8. Gepner P., Gamayunov V., Fraser D. Effective implementation of DGEMM on modern multicore CPU. *Procedia Computer Science*. 2012. Vol. 9. P. 126–135. DOI: 10.1016/j.procs.2012.04.014.
9. Barkalov K.A. Metody parallel'nyh vychisleni. N. Novgorod: NNGU im. N.I. Lobachevskogo, 2011. 124 s.
10. Salehian S., Liu J., Yan Y. Comparison of Threading Programming Models. *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW),* Lake Buena Vista, USA, 29 May–2 June 2017. IEEE, 2017. P. 766–774. DOI:10.1109/IPDPSW.2017.141.

11. Antonov A.S. Parallel'noe programmirovanie s ispol'zovaniem tehnologii OpenMP. M.: MGU, 2009. 77 s.
12. Grudzinskij D.A. Celesoobraznost' sozdanija standarta OpenMP dlja obektno-orientirovannyh jazykov programmirovanija na primere Java. *Vestnik Nizhegorodskogo universiteta im.* N.I. Lobachevskogo. 2011. № 3(2). S. 201–206.
13. Open MP Technical Report 7: *Version 5.0 Public Comment Draft*, July 2018. URL: https://www.openmp.org/wp-content/uploads/openmp-TR7.pdf.
14. Yliluoma J. Guide into OpenMP: Easy multithreading programming for C++. URL: https://bisqwit.iki.fi/story/howto/openmp/.
15. Frolov A.V., Voevodin V.V, Teplov A.M. Progonka, tochechnyj variant. URL: https://algowiki-project.org.
16. Popov G., Mastorakis N., Mladenov V. Calculation of the acceleration of parallel programs as a function of the number of threads. ICCOMP'10 Proceedings of the 14th WSEAS international conference on Computers: part of the 14th WSEAS CSCC multiconference, Corfu Island, Greece, 23–25 July 2010. *Stevens Point: World Scientific and Engineering Academy and Society,* 2010. Vol. II. P. 411–414.
17. Zhulkovskyi O. Shevchenko V., Zhulkovska I. Use of modern software increases the efficiency of computer simulation. *Збірник тез VI Всеукр. наук.-практ. конф. молодих науковців «Інформаційні технології — 2019»*. К.: Київ. ун-т ім. Б. Грінченка, 2019. С. 119, 120.
18. Shevchenko V.V. Zhul'kovskij O.A., Zhul'kovskaja I.I., Ocenka programmnyh sredstv povyshenija jeffektivnosti vychislitel'nogo jeksperimenta. *Naukova Ukraïna: Zbirnik statej V Vseukr. nauk. konf. studentiv»*. Dnipro: Akcent PP, 2019. S. 303–305.

ОЦІНКА ЕФЕКТИВНОСТІ РЕАЛІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ АЛГОРИТМІВ ЗАСОБАМИ СТАНДАРТУ OpenMP ДЛЯ РОЗПАРАЛЕЛЮВАННЯ ПРОГРАМ

О.О. Жульковський, І.І. Жульковська, В.В. Шевченко

Дніпровський державний технічний університет, вул. Дніпробудівська, 2, Кам’янське, 51918, Україна olalzh@ukr.net

Актуальність роботи полягає в необхідності підвищення ефективності комп'ютерного моделювання за рахунок використання прогресивних апаратних і програмних засобів розпаралелювання і синхронізації обчислень на сучасних комп’ютерів з багатоядерною архітектурою. Мета роботи – збільшення швидкодії обчислювальних алгоритмів методу прогонки шляхом використання прогресивних технологій програмування паралельних обчислень. В роботі використовувалися методи матричної алгебри, паралельних обчислень, аналізу ефективності алгоритмів і програм. Розроблено послідовні і розпаралелений алгоритми програмної реалізації методу прогонки; виконана порівняльна оцінка ефективності реалізації (швидкості виконання) даних алгоритмів засобами відкритого стандарту для розпаралелювання програм Open Multi-Processing для значного (до 5×107) порядку систем лінійних алгебраїчних рівнянь. Використання прогресивних технологій програмування при реалізації зазначених методів вирішення систем лінійних алгебраїчних рівнянь дозволило збільшити швидкість обчислень в 1,9-2,9 рази. Отримані результати кореспондуються з відомими літературними даними. Наукова новизна роботи полягає в подальшому розвитку прогресивних підходів до підвищення ефективності комп'ютерного моделювання, що використовує розпаралелені обчислювальні алгоритми методу прогонки шляхом їх реалізації на сучасних комп’ютерах. В роботі вперше отримані значення часу програмної реалізації послідовних і розпаралеленого на два потоки засобами Open Multi-Processing обчислювальних алгоритмів методу прогонки для систем лінійних алгебраїчних рівнянь значного порядку. Показана доцільність такого розпаралелювання для систем лінійних алгебраїчних рівнянь порядку більше 2,5×105. Практична значимість роботи полягає в використанні отриманих результатів при чисельному дослідженні стаціонарних і нестаціонарних, лінійних і нелінійних процесів в задачах математичного моделювання, де багаторазове рішення систем лінійних алгебраїчних рівнянь значного порядку є найбільш ресурсномістких етапом. Перспективи подальших досліджень проглядаються в контексті більш глибокого розпаралелювання алгоритму за рахунок використання більш масштабованої модифікації методу прогонки.

Ключові слова: обчислювальний алгоритм, чисельне рішення СЛАР, Open Multi-Processing метод прогонки, швидкодію, паралельні обчислення, OpenMP, прискорення обчислень.