Automated Design of a Parallel Program for Modeling Intraparticle Diffusion and Adsorption in Heterogeneous Nanoporous Media

Anatoliy Yu. Doroshenko [1], Mykhaylo R. Petryk [2], Dmytro M. Mykhalyk [2], Pavlo A. Ivanenko [1], and Olena A. Yatsenko [1]

[1] *Institute of Software Systems of the National Academy of Sciences of Ukraine, Glushkov prosp. 40, build. 5, Kyiv, 03187, Ukraine*
[2] *Ternopil Ivan Puluj National Technical University, Ruska str. 56, build. 1, Ternopil, 46001, Ukraine*

Abstract

The design of high-level programming abstractions in the form of algebra-algorithmic languages and models is one of the promising directions in the development and research of parallel computing systems. The purpose of such design is the development of architecture- and language-independent programming tools for multiprocessor platforms. The paper gives the results of the construction and parallelization of a program implementing a Crank-Nicolson scheme using algebra-algorithmic specifications represented in a natural-linguistic form. The Crank-Nicolson scheme is applied for obtaining a numerical solution of the model of a distributed mass transfer system and identifying the distribution of diffusion coefficients in a heterogeneous nanoporous medium. Heterogeneous media consisting of thin layers of particles of forked porous structure with different physical-chemical properties are widely used in science-intensive technologies and priority sectors of industry, medicine, ecology, etc. Such layers are distributed systems of pores consisting of two main spaces: micro- and nanopores of particles and macropores and cavities between particles. In modeling concentration and gradient fields for various diffusible components, an important scientific problem is the identification of kinetic parameters of a transfer, predetermining mass transfer velocity on macro- and micro levels, and also equilibrium conditions. The tools for automated design, synthesis, and auto-tuning of programs were applied that provided the translation of the Crank-Nicolson algorithm into source code in a target programming language and its tuning for execution environment to increase the program performance. The experiment results of auto-tuning the software implementation of the Crank-Nicolson scheme demonstrated high multiprocessor speedup on test data input.

Keywords

Automated software design, heterogeneous and nanoporous media, mass transfer, mathematic model, parallel computing, program auto-tuning

1. Introduction

One of the promising directions in the development and research of parallel computing systems is the construction of high-level programming abstractions in the form of algebra-algorithmic languages and models, the purpose of which is the development of architecture- and language-independent programming tools for multiprocessor platforms. In works [1–5], methods and tools for automated design, synthesis, and automatic self-adjustment (auto-tuning) of parallel programs based on Glushkov's system of algorithmic algebras (SAA) and the rewriting rules technique were developed. SAA is intended for formalized design of programs presented in the form of high-level schemes.

[1]13th International Scientific and Practical Conference from Programming UkrPROG'2022, October 11-12, 2022, Kyiv, Ukraine
EMAIL: doroshenkoanatoliy2@gmail.com (A. 1); mykhaylo_petryk@tntu.edu.ua (A. 2); d.mykhalyk@gmail.com (A. 3); paiv@ukr.net (A. 4); oayat@ukr.net (A. 5)
ORCID: 0000-0002-8435-1451 (A. 1); 0000-0001-6612-7213 (A. 2); 0000-0001-9032-695X (A. 3); 0000-0001-5437-9763 (A. 4); 0000-0002-4700-6704 (A. 5)

Auto-tuning provides optimization of programs — their adjustment to a specific computing environment by the automated search for the optimal program from a set of possible options, each of which is executed on a given parallel architecture with performance (execution time) measurement. Auto-tuning tools apply rewriting rules to transform programs.

In this paper, the tools developed are used for the design and auto-tuning of a parallel program for the implementation of the Crank-Nicolson scheme for execution on a multi-core processor. The Crank-Nicolson method is a special numerical technique for solving differential equations with partial derivatives based on a special scheme of the finite difference method, in particular, for heat conduction and diffusion equations. In [6], the Crank-Nicolson scheme is applied for obtaining a numerical solution of the model of a distributed mass transfer system and identification in the direct and conjugate formulation for identifying the distribution of diffusion coefficients in a heterogeneous nanoporous medium based on the theory of optimal control of the state of multicomponent systems. A gradient procedure for identifying parameters of internal transfer kinetics was implemented and the distribution of diffusion coefficient values for intraparticle and interparticle transfer was obtained.

Heterogeneous nanoporous media consisting of thin layers of particles of a branched porous structure with different physical and chemical properties are widely used in science-intensive technologies and priority sectors of industry, medicine, ecology, etc. Such layers are distributed multi-level pore systems consisting of two main spaces: micro- and nanopores of particles and macropores and cavities between particles. The intraparticle space has a higher degree of adsorptive capacity and, at the same time, a lower rate of diffusion intrusion compared to the interparticle space [7–9]. When modeling concentration and gradient fields for various diffused components, an important scientific problem is the identification of kinetic transport parameters that determine the rate of mass transfer at the macro- and micro-levels, as well as equilibrium conditions.

2. The tools for automated design, generation, and auto-tuning of programs

In this work, we use the system of algorithmic algebras [1] intended for the construction of algorithms and programs represented in the form of high-level schemes. SAA is the two-sorted algebra $GA = <\{Pr, Op\};\ \Omega_{GA}>$, where Pr and Op are sets of predicates and operators defined on an information set; Ω_{GA} is a signature of operations, consisting of logic (disjunction, conjunction, negation) and operator constructs, which will be considered further. In this work, a natural-linguistic form of operations notation is used. SAA is based on the language SAA/1 [1], which uses a representation of algorithms close to natural language and can be translated into a target programming language (C++, Java and other). Algorithms represented in SAA/1 are called SAA schemes. Identifiers of predicates are written in single quotes, and operators are enclosed in double ones. Predicates and operators in SAA/1 can be basic or compound. Basic elements are elementary atomic abstractions in SAA schemes.

Compound operators are built from basic operators based on the following operations:

- composition (sequential execution) of operators: "*operator* 1"; "*operator* 2";
- branching: IF '*condition*' THEN "*operator* 1" ELSE "*operator* 2" END IF;
- for loop: FOR (*counter* FROM *start* TO *fin*) "*operator*" END OF LOOP;
- asynchronous execution of p operators (threads): PARALLEL($j = 1,..., p$)("*operator* j");
- control point: CP '*condition*', which is associated with a synchronization condition that has the value "false" until the computation process has reached this point in an algorithm scheme, and the value "true" after the point is reached;
- synchronizer: WAIT '*condition*', which delays the computation until the value of the synchronization condition becomes "true".

The developed integrated toolkit for program design and synthesis (IDS) [1, 4] provides automated construction of algorithm schemes and generation of corresponding code in target programming languages (C, C++, Java). Algorithms are designed using a list of SAA operations and a tree. The user selects constructs from the list and adds them to the algorithm construction tree. At each step of the design process, the system allows the user to select only those operations, the insertion of which into the scheme does not violate its syntactic correctness. The algorithm tree is further used to

automatically generate the text of the SAA scheme and the code in a programming language. The representation of each SAA construct in the text in the programming language is specified as a template in the database of the integrated toolkit.

The application of formal methods in the IDS toolkit and rewriting rules of the TermWare system [4, 5] provides an opportunity to automate the manual work of programmers and perform more complex parallelization of algorithms. TermWare provides a language for describing rewriting rules that operate on special data structures — terms, as well as tools for processing and interpreting rules for transforming terms. The performance of the designed programs can be improved by using the developed automatic program tuning system TuningGenie [2, 3]. TuningGenie is intended for the automated generation of autotuner applications from source code in Java language. The idea of the autotuner is to empirically evaluate several versions of the input program and select the best one — with reduced execution time and higher accuracy of the results. The system works with the program text, using the expert knowledge of the developer, who adds certain metadata (parameter names and value ranges) to the source code in the form of special comments (pragmas). By using such expert knowledge, the number of options for the evaluation program is reduced, which increases the performance of the autotuner.

TuningGenie uses TermWare to extract expert knowledge from original software code and generate a new version of a program at each autotuning iteration. By manipulating terms represented as abstract syntax trees, TuningGenie can perform structural changes in program computations using the declarative style of the TermWare system. In the process of analyzing the source code and building a term tree, the autotuner builds a set of configurations based on expert data. These configurations are then translated into rewriting rules. Also, at this preliminary stage, some program parameter values are computed. The outputs of this step include a program term, a set of parameterized rewriting rules, and a set of rule configurations that specify specific parameter values. Each of these configurations specifies a unique version of the input application. Next, TuningGenie searches for the most efficient configuration for a given computing environment.

Expert knowledge about the subject domain and implementation is stored in a source code as special directives called pragmas. Pragmas are special form comments, so they are ignored by the Java compiler. Adding pragmas does not change the structure of computations, and a program equipped with pragmas can be translated by any compiler without additional libraries. Below is an example of one of TuningGenie's pragmas called *tunableParam*. This pragma sets the possible values for the *NumThreads* variable, which stores the number of parallel threads, in a range $[1\ldots16]$ with step 1:

```
// tunableParam name=NumThreads start=1 stop=16 step=1
int NumThreads = 1;
```

The *tunableParam* pragma is used for algorithms that use data parallelization: it allows finding the optimal decomposition of calculations by estimating the size of a block that is executed on one processor. It can also be used when it is necessary to estimate the optimal amount of limited resources, for example, the size of the cache.

In this work, the developed tools are used for the automated design and tuning of a parallel program for the implementation of the Crank-Nicolson scheme. The formulation of the problem for which the scheme is used is considered in the next section.

3. Formulation of the problem of two-level transport in a heterogeneous system of nanoporous particles and an algorithm for implementing the gradient method of identifying intraparticle diffusion coefficients

Mass transfer in a system of heterogeneous media consisting of small particles of a nanoporous structure causes two types of mass transfer: diffusion in macropores, due to the space between particles, and diffusion in the system of micro- and nanopores inside the particles of a heterogeneous medium. To determine the contribution of each type of diffusion to the overall mass transfer system, it

is necessary to know the values of the parameters that define the adsorption equilibria, etc. In this work, a heterogeneous nanoporous medium is considered, which consists of a large number of $(n+1)$ thin layers, nanoporous spherical particles located perpendicular to the direction of the incoming flow and interconnected by a system of conditions of *n*-interface interactions. This is decisive for heterogeneous thin nanoporous samples, especially in the case of gas diffusion before the state of adsorption equilibrium, taking into account the system of multi-interface interactions. Mass transfer occurs through the permeable surface of the bed in two directions: in the axial direction — in the space of macropores (direction z along the height of the bed, perpendicular to the surface of the layers) and radial — in the space of micro- and nanopores. The evolution of the system towards equilibrium is carried out by concentration gradients in macropores and micro- and nanopores of particles (from the surface to the center).

The mathematical model of such a transfer, taking into account the above-mentioned physical factors, is described in the form of a mixed boundary value problem [7]: to construct a bounded in domain

$$D_n = \left\{ t > 0,\ r \in (0, R),\ z \in \bigcup_{k=1}^{n+1} (l_{k-1}, l_k);\ l_0 = 0;\ l_{n+1} \equiv l < \infty \right\}$$

solution of the system of equations in partial derivatives written in matrix form:

$$\frac{\partial}{\partial t}\begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_{n+1} \end{bmatrix} = \frac{\partial}{\partial z}\left(\begin{bmatrix} D_{\text{inter}_1} & & \dots & 0 \\ & D_{\text{inter}_2} & \dots & \\ \dots & \dots & \dots & \dots \\ 0 & & \dots & D_{\text{inter}_{n+1}} \end{bmatrix} \frac{\partial}{\partial z}\begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_{n+1} \end{bmatrix} \right) -$$

$$- \left(\begin{bmatrix} \nu_1 D_{\text{intra}_1} & & \dots & 0 \\ & \nu_2 D_{\text{intra}_2} & \dots & \\ \dots & \dots & \dots & \dots \\ 0 & & \dots & \nu_{n+1} D_{\text{intra}_{n+1}} \end{bmatrix} \frac{\partial}{\partial r}\begin{bmatrix} q_1 \\ q_2 \\ \dots \\ q_{n+1} \end{bmatrix} \right)_{r=R}, \tag{1}$$

$$\frac{\partial}{\partial t}\begin{bmatrix} q_1 \\ q_2 \\ \dots \\ q_{n+1} \end{bmatrix} = \frac{1}{r^2}\frac{\partial}{\partial t}\left(\begin{bmatrix} D_{\text{intra}_1} & & \dots & 0 \\ & D_{\text{intra}_2} & \dots & \\ \dots & \dots & \dots & \dots \\ 0 & & \dots & D_{\text{intra}_{n+1}} \end{bmatrix} r^2 \frac{\partial}{\partial r}\begin{bmatrix} q_1 \\ q_2 \\ \dots \\ q_{n+1} \end{bmatrix} \right), \tag{2}$$

with initial conditions:

$$c_k(t,z)_{t=0} = 0\ ,\ q_k(t,z)_{t=0} = 0, \tag{3}$$

boundary conditions:

$$c_{n+1}(t, z=l) = c_{\infty_{n+1}},\ \frac{\partial c_1}{\partial z}(t, z=0) = 0, \tag{4}$$

$$\frac{\partial}{\partial r} q_k(t,r,z)_{r=0} = 0,\ q_k(t,r,z)_{r=R} = K_k c_k(t,z). \tag{5}$$

and a system of conditions of *n*-interface interactions along coordinate z:

$$\left[c_k(t,z) - c_{k+1}(t,z) \right]\Big|_{z=l_k} = 0,\ \left[\frac{\partial}{\partial z} c_k(t,z) - \nu_k \frac{\partial}{\partial z} c_{k+1}(t,z) \right]\Bigg|_{z=l_k} = 0, \tag{6}$$

where $K_k = \dfrac{q_{\infty_k}}{c_{\infty_k}}$, $\overline{q}_k(t,z) = \dfrac{1}{R^2}\int_0^R q_k(t,x,z)\, r dr$, $\nu_k = \dfrac{3\left(1 - \varepsilon_{\text{inter}_k}\right)}{R \cdot \varepsilon_{\text{inter}_k}}$, $k = \overline{1, n+1}$.

The system of differential equations (1) describes the transfer in the interparticle space, limited by the right-hand parts of the system, which take into account the effect of micro transfer on the outer surfaces of particles or crystallites $(r = R)$ for each *k*-th layer of the bed. The system of equations (2) describes transport in micro- and nanopores of the intraparticle space. The relation between concentrations c_k in the interparticle space and concentrations q_k in the intraparticle space is defined by the system of right-hand boundary conditions (5), which also defines the conditions of adsorption equilibrium on the surfaces of spherical particles, $\Delta l_k = l_{k+1} - l_k$ is the thickness of the *k*-th layer, R is the radius of the particle. The problem is solved using the Crank-Nicolson difference scheme [6].

The procedure for implementing the gradient method of identifying intraparticle mass transfer coefficients (D_{intra_m}, $m = \overline{1, n+1}$) is based on the use of the system state matrix $\mathrm{M}_m\left(t_k, z_i, D^{\theta}_{\text{intra}_m}\right)$, which corresponds to the total accumulated mass of the diffused component in the pores of particles in the interparticle and intraparticle space [10]. The matrix is defined by the formula

$$M_m(t,z) = \left[U_m\left(t, z, D_{\text{intra}_m}\right) + \frac{1}{R}\int_0^R q_m\left(t, r, z, D_{\text{intra}_m}\right) r dr \right]_{L^2(\gamma_m), m=\overline{1,n+1}},$$

where $M_{\exp} = \left[M_{\exp_{km}} \right]_{\substack{m=\overline{1,n+1} \\ k=\overline{1,N}}}$ is the matrix of experimental studies data for *i*-th surface and *k*-th time layers (Figure 1) [11].

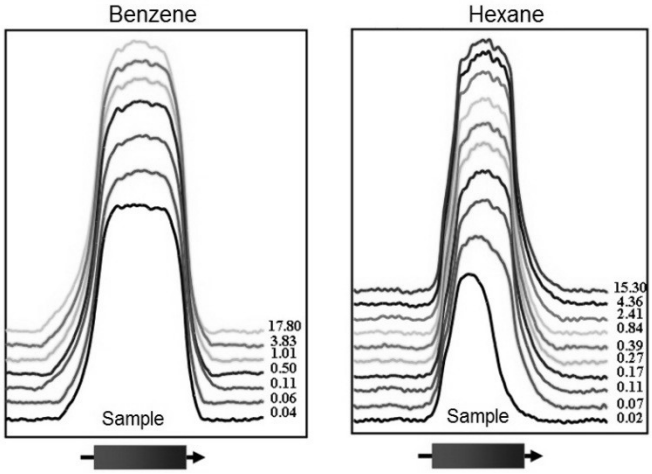

Figure 1: Experimental data of studies of competitive mass transfer in a heterogeneous nanoporous catalytic medium [11]

In the matrix $M_m\left(t_k, z_i, D^{\theta}_{\text{intra}_m}\right)$, time and space variables t and z define specific states of the competitive transfer system for heterogeneous (by direction z) catalytic medium of nanoporous particles, for which identification of kinetic parameters-coefficients of intraparticle diffusion is carried out $D_{\text{intra}_m}, m = \overline{1, n+1}$ for each of $n+1$ layers.

To identify this distribution (vector) D_{intra_m}, one of the gradient methods is used, the mathematical substantiation of which is applied to the problems of parametric identification of multicomponent distributed systems and is presented in [12, 13]. Based on the specifics of the problem, the method of minimum errors is the most suitable, according to which, to define the $(\theta+1)$-th the approximation of the diffusion coefficient in the intraparticle space D_{intra_m}, we apply the following gradient-identification procedure defined in matrix form:

$$D^{\theta+1}_{\text{intra}_m} = D^{\theta}_{\text{intra}_m} - \eta^{\theta} \cdot \nabla J\left(D^{\theta}_{\text{intra}_1}, \ldots, D^{\theta}_{\text{intra}_{n+1}}\right),\ m = \overline{1, n+1},$$

where η^{θ} is the value of the coefficient for each θ-th step of the iteration.

The general scheme of the algorithm for identifying intraparticle diffusion coefficients D_{intra_m}, $m = \overline{1, n+1}$, is shown in Figure 2.

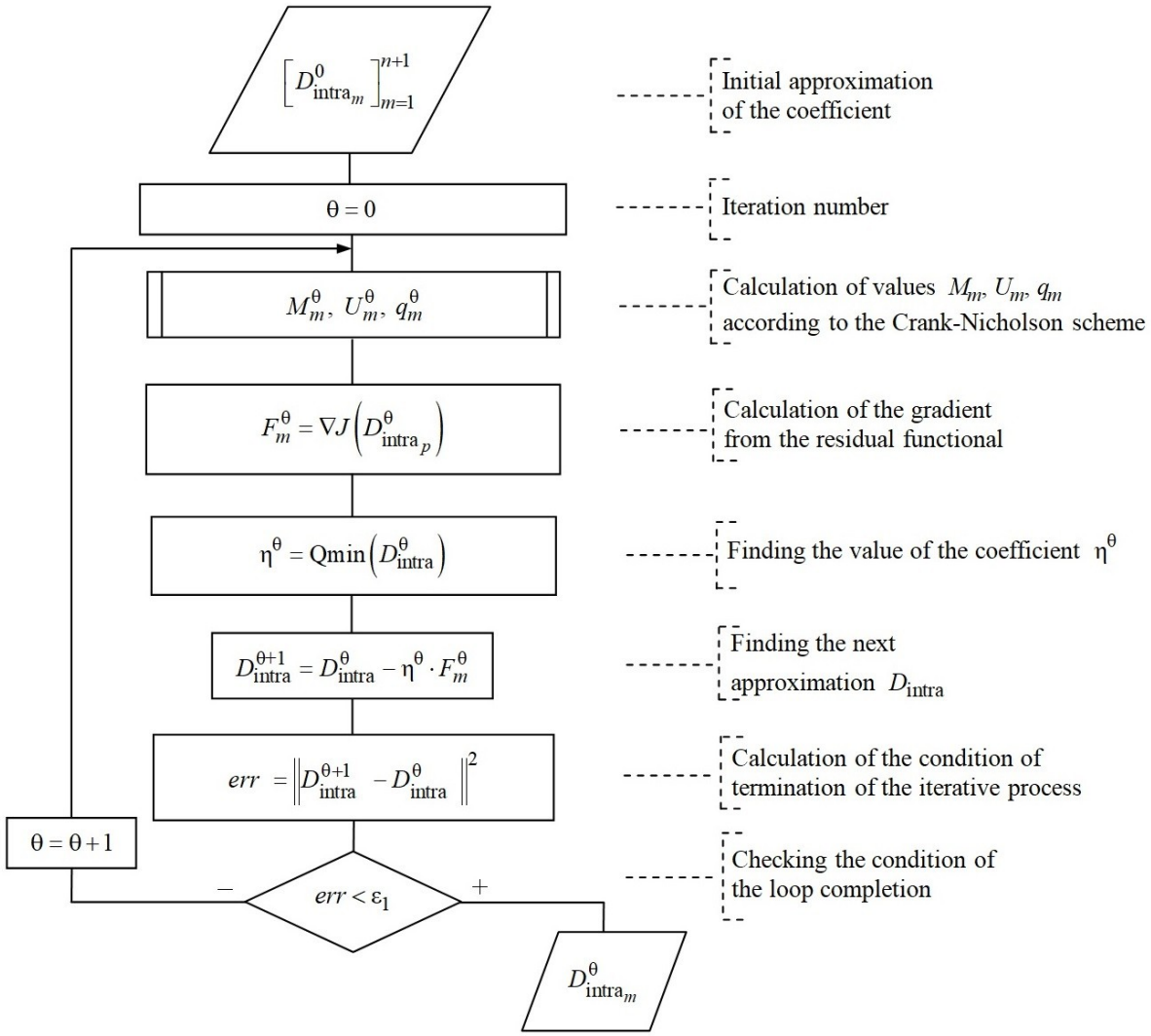

Figure 2: Block diagram of the algorithm for identifying intraparticle diffusion coefficients

The results of the experiments on numerical modeling and identification of kinetic parameters based on the given algorithm are considered in [6].

4. Results of experiments

This section gives the results of tuning the parallel program implementing the Crank-Nicolson scheme to a target execution environment and numerical modeling and identification of kinetic parameters of a heterogeneous nanoporous mass transfer system.

Further, the process of parallelization of one of the subroutines of the implementation of the Crank-Nicolson scheme is described. The sequential SAA scheme of this routine, designed using the IDS toolkit, is shown below. The scheme is a loop by variable $k \in [1,...,N]$, in which functions iterate_c(k) and iterate_q(k) calculate the k-th layer for concentration values c_k and q_k in the interparticle and intraparticle space, respectively.

SCHEME CRANK-NICOLSON SEQUENTIAL ====

“iterations”
==== FOR (*k* FROM 1 TO *N*)
“iterate_c(*k*)”;
“iterate_q(*k*)”
END OF LOOP

END OF SCHEME

The parallelization of the scheme consists in the division of the segment $[1...N]$ into *NumThreads* sections processed simultaneously. The SAA scheme of the parallel algorithm is as follows:

SCHEME CRANK-NICOLSON PARALLEL ====

“iterations”
==== PARALLEL(*j* = 1,..., *NumThreads*)
(
“IterateThread(*j*)”
);
WAIT ‘Processing in all (*NumThreads*) threads is finished’;

“IterateThread(*j*)”
==== “*chunk* := *N* / *NumThreads*”;
“*start* := (*j* – 1) * *chunk* + 1”;
“*end* := (*j* – 1) * *chunk* + *chunk*”;
IF (*j* = *NumThreads*) THEN
“*end* := *N*”
END IF;
FOR (*k* FROM *start* TO *end*)
“iterate_c(*k*)”;
“iterate_q(*k*)”
END OF LOOP;
CP ‘Processing in the thread (*j*) is finished’;

END OF SCHEME

Based on the designed scheme, the IDS toolkit performed automated code generation in the Java language. Next, the TuningGenie system was used to find the optimal configuration for the program when running on a multi-core processor. With this system, various combinations of performance-related JVM compiler options (-XX:-UseBiasedLocking/-XX:BiasedLockingStartupDelay) were tested at various thread count values. Changing compiler options did not show a significant impact on computational performance, so the following results refer mainly to changes to the *NumThreads* parameter. The characteristics of the test environment were the following: quad-core processor Intel Core i7-6820HQ with frequency 2,7 GHz; cache L1: 32k/32k x4, cache L2/L3: 256k x4, 8 MB; RAM 16 GB, 2133 MHz LPDDR3; OpenJDK build 11.0.2+9; macOS v11.6.

Intuition suggests that the optimal number of threads should be 8, since the processor is a quad-core with support for hyper-threading technology. This configuration is marked with a triangle in the diagram (Figure 3) and demonstrated an acceptable multiprocessor speedup of 5.16 on the test input data.

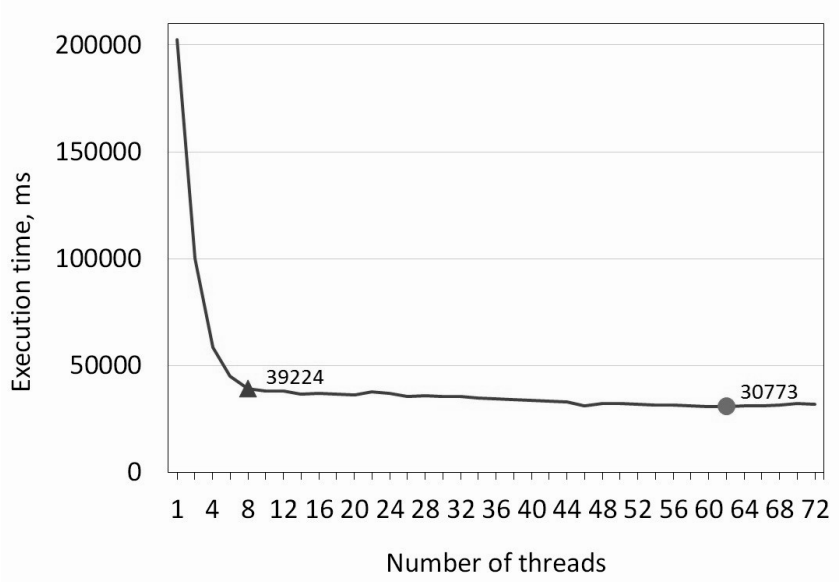

Figure 3: The dependency of the execution time on the number of threads for the parallel program implementing the Crank-Nicolson scheme

However, increasing the number of threads further resulted in additional performance improvement of approximately 140%. The fastest application configuration with 62 threads (indicated by the dot in the diagram) achieved a fairly good multiprocessor speedup of 6.5. This is explained by the increased efficiency of using processor caches. Fine-grained decomposition is more likely to be placed in the L1–L3 cache and saves time for access to "slow" RAM. This effect is balanced by the additional time cost of thread competition. Nevertheless, the overall effect was positive.

The results of the parametric identification of kinetic parameters carried out in accordance with the considered methodology and using the specified experimental data are presented in Figure 4 and Figure 5. Like the results of the experimental studies (Figure 1), they were obtained for different time snapshots for the cases of the process of independent diffusion of benzene and hexane. Figure 4 (a) and Figure 5 (a) show the graphical distributions of the values of the identified diffusion coefficients along coordinate z during the diffusion of benzene for the moments of time $\tau = 0.02$ hrs and $\tau = 0.39$ hrs.

From the given graphic results of the process of identification of diffusion coefficients D_{intra_m}, $m = \overline{1, n+1}$, for the intraparticle space, it is possible to distinguish general characteristic laws, which consist in a somewhat pseudo-exponential decrease of the values of diffusion coefficients in the range of $3 \cdot 10^{-12} \div 4 \cdot 10^{-14}$ m/sec2 (taking into account the computation errors). A similar picture is observed for the diffusion process of hexane, for which the procedure of identification of diffusion coefficients in the intraparticle space D_{intra_m} for time snapshots $\tau = 0.04$ hrs and $\tau = 0.50$ hrs was carried out. The results of the identification are presented in Figure 5 (a).

The obtained results of the identified distributions of diffusion coefficients in the intraparticle space along coordinate z (the main direction of heterogeneity of the system) allow sufficiently accurate modeling of the concentration fields and integral mass distributions in the heterogeneous catalytic nanoporous layer. Figure 4 (b) and Figure 5 (b) present the concentration profiles demonstrating a comparative analysis of model curves (2), constructed as a result of the numerical solution of the problem (1)–(6) using the Crank-Nicolson scheme, and also the comparison of the results of identified coefficients within particle diffusion with approximations of experimental distribution curves of the absorbed mass in the nanoporous layer used for the identification procedure.

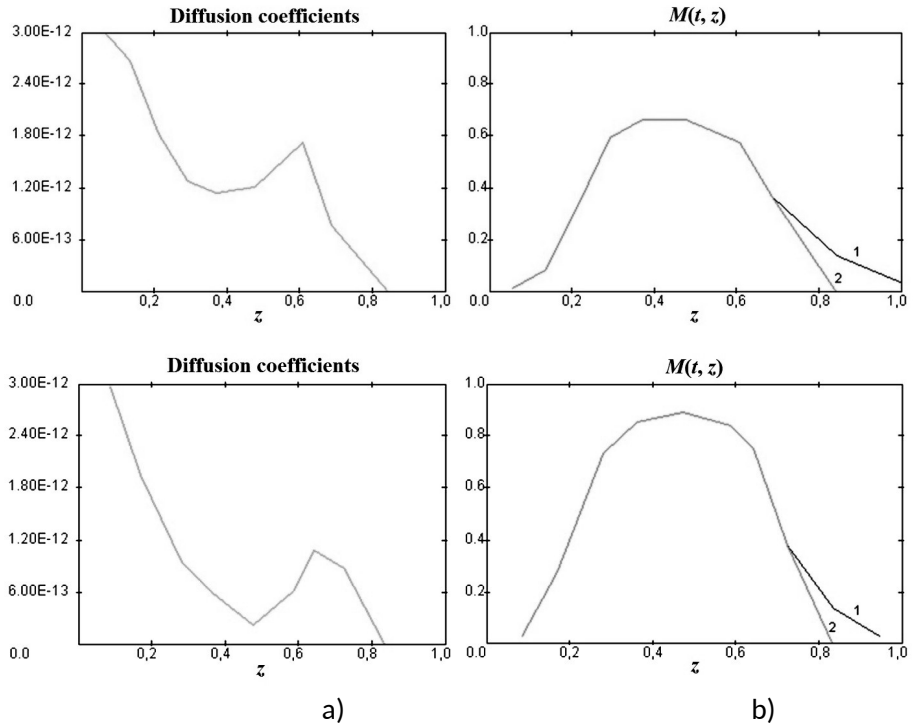

Figure 4: The results of identification of diffusion coefficients for time points $\tau = 0.02$ and $\tau = 0.39$ for benzene diffusion: a) the distribution of diffusion coefficients in the intraparticle space; b) the comparison of model (2) and experimental (1) curves

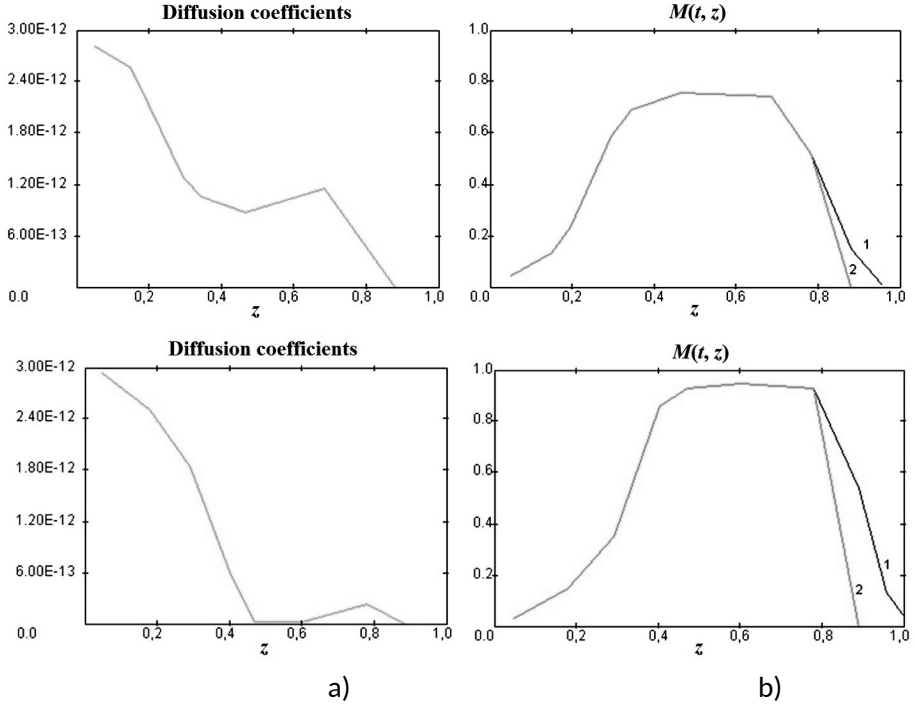

Figure 5: The results of identification of diffusion coefficients for time points $\tau = 0.04$ and $\tau = 0.50$ for hexane diffusion: a) the distribution of diffusion coefficients D_{intra_m} in the intraparticle space; b) the comparison of model (2) and experimental (1) curves

5. Conclusion

Automated design and parallelization of the Crank-Nicolson scheme implementation program using algebraic-algorithmic specifications presented in the natural-linguistic form were performed. Previously developed tools for automated design, synthesis, and auto-tuning of programs were applied, which provided the transformation of algebraic-algorithmic schemes into source code in a programming language and its automated tuning to the execution environment to increase the program performance. The results of the experiment on the automated tuning of the parallel program demonstrated a high multiprocessor speedup on test input data.

References

[1] P. I. Andon, A. Yu. Doroshenko, K. A. Zhereb, O. A. Yatsenko, Algebra-Algorithmic Models and Methods of Parallel Programming, Akademperiodyka, Kyiv, 2018.

[2] A. Doroshenko, P. Ivanenko, O. Novak, O. Yatsenko, A mixed method of parallel software auto-tuning using statistical modeling and machine learning, in: V. Ermolayev, M. Suárez-Figueroa, V. Yakovyna, H. C. Mayr, M. Nikitchenko, A. Spivakovsky (Eds.) Information and Communication Technologies in Education, Research, and Industrial Applications, ICTERI 2018, volume 1007 of Communications in Computer and Information Science, Springer-Verlag, Cham, 2019, pp. 102–123. doi:10.1007/978-3-030-13929-2_6.

[3] P. Ivanenko, A. Doroshenko, K. Zhereb, TuningGenie: auto-tuning framework based on rewriting rules, in: V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky, G. Zholtkevych (Eds.) Information and Communication Technologies in Education, Research, and Industrial Applications, ICTERI 2014, volume 469 of Communications in Computer and Information Science, Springer-Verlag, Cham, 2014, pp. 139–158. doi:10.1007/978-3-319-13206-8_7.

[4] A. Doroshenko, K. Zhereb, O. Yatsenko, Developing and optimizing parallel programs with algebra-algorithmic and term rewriting tools, in: V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky, G. Zholtkevych (Eds.) Information and Communication Technologies in Education, Research, and Industrial Applications, ICTERI 2013, volume 412 of Communications in Computer and Information Science, Springer-Verlag, Cham, 2013, pp. 70–92. doi:10.1007/978-3-319-03998-5_5.

[5] A. Doroshenko, R. Shevchenko, A rewriting framework for rule-based programming dynamic applications, Fundamenta Informaticae 72 (2006) 95–108.

[6] M. R. Petryk, D. M. Mykhalyk, I. V. Hoianiuk, High-performance methods of identification of kinetic parameter for monodiffusion adsorption mass transfer [in Ukrainian], Bulletin of the National University of Water and Environmental Engineering 4 (2020) 91–104.

[7] J. Kärger, D. M. Ruthven, Diffusion and adsorption in porous solids, in: F. Schüth, K. S. W. Sing, J. Weitkamp (Eds.). Handbook of Porous Solids, Wiley-VCH, Weinheim, 2002, pp. 2089–2173.

[8] N. Y. Chen, T. F. Degnan, M. C. Smith, Molecular Transport and Reaction in Zeolites: Design and Application of Shape Selective Catalysis, Wiley, New York, 1994.

[9] D. Ruthven, Principles of Adsorption and Adsorption Processes, Wiley, New York, 1984.

[10] J. Kärger, F. Grinberg, P. Heitjans, Diffusion Fundamentals, Leipziger Universitätsverlag, Leipzig, 2005.

[11] M. R. Petryk, J. Fraissard, Mathematical modeling of nonlinear competitive two-component diffusion in media of nanoporous particles, Journal of Automation and Information Sciences, 41 (2009) 37–55. doi:10.1615/JAutomatInfScien.v41.i3.60.

[12] I. V. Sergienko, V. S. Deineka, System Analysis of Multicomponent Distributed Systems [in Russian], Naukova dumka, Kyiv, 2009.

[13] M. R. Petryk, J. Fraissard, D. M. Mykhalyk, Modeling and analysis of concentration fields of nonlinear competitive two-component diffusion in medium of nanoporous particles, Journal of Automation and Information Sciences, 41 (2009) 13–23. doi:10.1615/JAutomatInfScien.v41.i8.20.