INFORMATICA, 1996, Vol. 7, No. 3, 295–310

RUNNING FINITE-DIFFERENCE SCHEMES FOR 3D DIFFUSION PROBLEMS ON PARALLEL COMPUTERS WITH DISTRIBUTED MEMORY

Raimondas ČIEGIS

Institute of Mathematics and Informatics
Akademijos 4, 2600 Vilnius, Lithuania
Email: raimondas.ciegis@ fm.vtu.lt

Juozas ŠIMKEVIČIUS

Vytautas Magnus University
Vileikos 8, 3000 Kaunas, Lithuania

Jerzy WAŚNIEWSKI

The Danish Computer Center for Research and Education
UNI-C, DTH, Bldg. 305, DK-2800 Lyngby, Denmark

Abstract. In this papper we consider the problem of solving 3D diffusion problems on distributed memory computers. We present a parallel algorithm that is suitable for the number of processors less or equal 8. The pipelining method is used to enlarge the number of processors till 64. The computational grid decomposition method is proposed for heterogenous clusters of workstations which preserves the load balancing of computers. The numerical results for two clusters of workstations are given.

Key words: finite difference schemes, parallel algorithms, LOD methods, distributed memory computers.

1. Introduction. Parallel and parallel/vector computers are widely used for modelling a large variety of physical phenomena. Such processes can be described by means of partial differential equations, and numerical solutions of such equations are considered. Resent advances in parallel hardware and software are giving us a possibility to achieve the following goals (Freeman and Phillips, 1991):

- to decrease the execution time of programs so that results can be obtained

in reasonable time (possibly even in real time),

- to obtain higher accuracy or more accurate modelling of the underlying physical problem by increasing the problem size.

Distributed memory computers have enabled the mathematical modelling of applications requiring computational resources previously unavailable. The discrete problems, obtained by finite difference method, for example, can become very large as the number of grid points is increased and the time step is small.

In order exploit fully the computing power of parallel and/or vector computers, existing numerical algorithms must be re-examined. With parallel computers we again need to worry about details of data transfer time between memory and processors and we must take into account which numerical operations are most efficient. For many problems the fastest sequential algorithm does not parallelize well and so a distinct algorithm is used. At the same time for parallel computers with distributed memory the theory and compilators of automatic parallelization are in the first stage of development.

As was mentioned above we deal with big variety of mathematical models and various computer systems. Hence the following goal is important in scientific computing:

- to test the performance efficiency of parallel algorithms designed for solving the most important types of PDE.

It is well-known that complicated mathematical models can be effectivelly solved by using the splitting method, when different physical processes are solved numerically separately (Marchuk, 1990). Then results obtained for various types of PDEs will give a useful measure of computation speed likely to be achieved for a large application. For example, air pollution models can be splitted into the following four main parts: advection, deposition, diffusion and chemistry (Zlatev and Waśniewski, 1992). The discrete approximations of air polution models are used as important benchmarks for testing numerical algorithms on various types of parallel computers (Zlatev *et al.*, 1991). We note that diffusion process was not included into these experiments.

The numerical methods for solving 3D diffusion problems on parallel computers with distributed memory are considered in this paper. In fact we are especially interested in algorithms which are efficient on clusters of workstations connected over local area network (LAN). A cluster of workstatios is a useful environment for many simulations. Typically communication overheads

between workstations are still quite high and must be taken into account. Hence the problem of minimizing the amount of data communicated between processors is very important for such class of parallel computers. We have used PVM for a practical realization of our algorithms (Geist *et al.*, 1993).

2. Description of the model problem. Let Q be a cube in R^3:

$$Q = \{x : x = (x_1, x_2, x_3), \quad 0 < x_1, x_2, x_3 < 1\}, \tag{1}$$

γ be its boundary, $(0, T]$ be a bounded half open interval in R and let $Q_T = Q \times (0, T]$. We consider the 3D, constant coefficient diffusion model problem

$$\begin{aligned} \frac{\partial u}{\partial t} &= \sum_{i=1}^{3} \frac{\partial^2 u}{\partial x_i^2} + f(x,t), \quad (x,t) \in Q_T, \\ u(x,t) &= \mu(x,t), \quad (x,t) \in \Gamma = \gamma \times (0,T], \\ u(x,0) &= u_0(x), \quad x \in \overline{Q}. \end{aligned} \tag{2}$$

Let ω_τ, ω_h be difference grids

$$\begin{aligned} \omega_\tau = \{ & t_{j+\alpha/3} = (j + \alpha/3)\tau, \\ & \alpha = 0,1,2,3,\ j = 1,2,\ldots,K,\ K\tau = T\}, \end{aligned} \tag{3}$$

$$\begin{aligned} \omega_h = \{ & (x_{1i_1}, x_{2i_2}, x_{3i_3}), \\ & x_{\alpha i_\alpha} = i_\alpha h,\ \alpha = 1,2,3,\ i_\alpha = 1,2,\ldots,N-1,\ Nh = 1\}. \end{aligned} \tag{4}$$

We consider the following Locally One Dimensional (LOD) scheme

$$\begin{aligned} \frac{y_\alpha - y_{\alpha-1}}{\tau} = \Lambda_\alpha \frac{y_\alpha + y_{\alpha-1}}{2} + \varphi_\alpha, & \qquad (x,t) \in \omega_h \times \omega_\tau, \\ y_\alpha = \nu_\alpha, & \qquad (x,t) \in \gamma_h \times \omega_\tau,\ \alpha = 1,2,3, \\ y(x,0) = u_0(x), & \qquad y(t_{j+1}) = y_3, \end{aligned} \tag{5}$$

where Λ_α is the discrete approximation, resulting from the use of 2nd-order central differences in the x_α direction. There we have used the notation for grid functions

$$y_\alpha = y_{j,\alpha} = y(t_j + \alpha/p), \quad \alpha = 0,1,2,3.$$

The analysis of the convergence of LOD schemes is given in Čiegis and Kiškis (1994a), Čiegis and Kiškis (1994b).

Some theoretical problems are still not solved completely, hence numerical experiments with 3D LOD schemes are also important in order to test some theoretical hypothesis (Čiegis and Kiškis, 1994b).

3. Parallel algorithm. We will consider the realization of LOD scheme (5) on a distributed-memory multiprocessor or on a cluster of workstations. We will make the following assumptions (see also Lin and Chen, 1994):

1. There are p processors that are connected through a network. Each processor contains a local memory that is sufficient to hold all data needed. There is no global memory.
2. A cluster of processors is heterogeneous, the execution time of all computations (addition, multiplication, division) can be different for each processor.
3. Different processors can perform computations concurrently; different pair of processors can communicate each other concurrently.

In order to get a solution of LOD scheme (5) at each time level t_j, one must solve $3(N-1)\times(N-1)$ tridiagonal systems of size $N-1$. The splitting direction also changes at each fractional time step, hence sequential factorization algoritms are not efficient in the case of large number of processors.

There exist many parallel algorithms to solve tridiagonal systems; see, e.g., Lin and Chen (1994), Demmel *et al.* (1993), or to solve multimidimensional problems by using alternating-direction type methods; see, e.g., Johnson (1990), Ortega (1988). However, all known parallel direct tridiagonal solvers carry a high floating-point overhead (typically 2.5 times more arithmetic operations). Hence we restrict our analysis to the well-known modification of the Thomas algoritm, which requires the same amount of floating-point operations as the basic Thomas methods.

Let consider the system of linear equations

$$TY = F, \tag{6}$$

where T is a tridiagonal matrix

$$T = \begin{pmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & a_{M-1} & b_{M-1} & c_{M-1} \\ 0 & 0 & 0 & 0 & a_M & b_M \end{pmatrix}. \tag{7}$$

In the *2-way Gaussian elimination* algorithm (this method is called *twisted factorization* algorithm in Demmel *et al.*, 1993) the sequential elimination process is carried out in parallel from both sides. In the first elementary loop we calculate factorization parameters α_i, β_i :

$$\alpha_i = -\frac{c_i}{b_i + a_i\alpha_{i-1}}, \quad \beta_i = \frac{f_i - a_i\beta_{i-1}}{b_i + a_i\alpha_{i-1}}, \tag{8}$$
$$\alpha_0 = 0, \quad \beta_0 = 0, \quad i = 1, 2, \ldots, L;$$

$$\alpha_i = -\frac{a_i}{b_i + c_i\alpha_{i+1}}, \quad \beta_i = \frac{f_i - c_i\beta_{i+1}}{b_i + c_i\alpha_{i+1}}, \tag{9}$$
$$\alpha_{M+1} = 0, \quad \beta_{M+1} = 0, \quad i = M, M-1, \ldots, L+1.$$

In our case coefficients α_i must be calculated only once. Then processors exchange coefficients (α_k, β_k) and $(\alpha_{k+1}, \beta_{k+1})$ by using one message and compute the boundary components of the solution

$$y_L = \frac{\beta_L + \alpha_L\beta_{L+1}}{1 - \alpha_L\alpha_{L+1}}, \quad y_{L+1} = \frac{\beta_{L+1} + \alpha_{K+1}\beta_L}{1 - \alpha_L\alpha_{L+1}}. \tag{10}$$

After this both processors in parallel evaluate the back-substitution loop

$$y_i = \alpha_i y_{i+1} + \beta_i, \quad i = L-1, L-2, \ldots, 1, \tag{11}$$
$$y_i = \alpha_i y_{i-1} + \beta_i, \quad i = L+2, L+3, \ldots, M.$$

We see that 2-way Gaussian elimination takes as many arithmetic operations as the standard factorization, and can be carried out with twofold parallelism.

4. Data layout on distributed memory computer. It is well-known that for distributed-memory computers an important issue is *data layout*, or how the computational grid is partitioned across the processors. This determines both the

amount of parallelism and the cost of communication. The latter characteristic is very important for clusters of workstations, because communication between processors is slow in comparence with computation speeds.

The 2-way Gaussian elimination algoritms defines a distribution of $N \times N \times N$ computational grid over $p \leqslant 8$ processors grid. Let consider a docomposition of the three-dimensional cube into 8 subdomains (see Fig. 1).

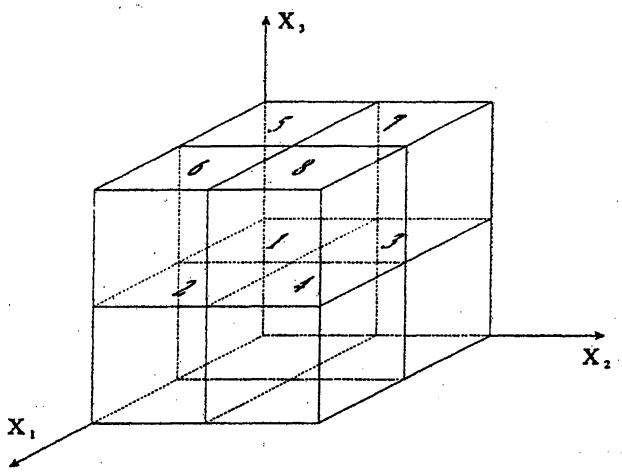

Fig. 1. Decomposition of 3D cube into 8 subcubes.

The algorithm of assigning 8 subcubes to p processors is implemented in three steps. We will describe in detail the first step of cube decomposing in the x_3 direction.

If $p = 1$, then all subcubes $Q_1, Q_2, \ldots, Q_8$ are assigned to the first processor. If $p > 1$, then we divide all processors into two sets. Let denote

$$p_L = \lfloor p/2 \rfloor, \quad p_R = p - p_L.$$

The subcubes Q_1, Q_2, Q_3, Q_4 are assigned to processors $1, 2, \ldots, p_L$ and subcubes Q_5, Q_6, Q_7, Q_8 are assigned to processors $p_{L+1}, p_{L+2}, \ldots, p$. Next we repeat this procedure in x_2 direction for both groups of processors independently. In the third stage we simillarly divide computational grid in x_1 direction.

The examples of data layouts for $p = 3$ and $p = 6$ processors are given in Fig. 2.

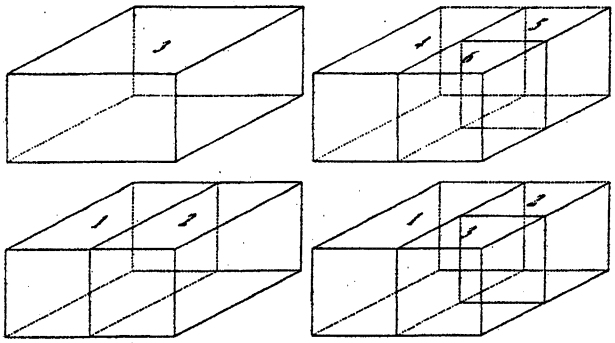

Fig. 2. Data layouts for $p = 3$ (a) and $p = 6$ processors (b).

We finish this section with estimation of data amount which must be communicated between processors. Let consider the case when we have 8 processors and processors can not communicate concurrently. The number of elements transmitted by each processor to its neighbors is equal to the number of boundary grid points. In the case of $p = 8$ processors the total number of such grid points is equal to $3(N-1) \times (N-1)$. It is easy to see that this number is minimal possible for different partitions. For example, if we consider the partition of the cube, which is given in Fig. 3, then the number of internal boundary grid points is equal to $7(N-1) \times (N-1)$.

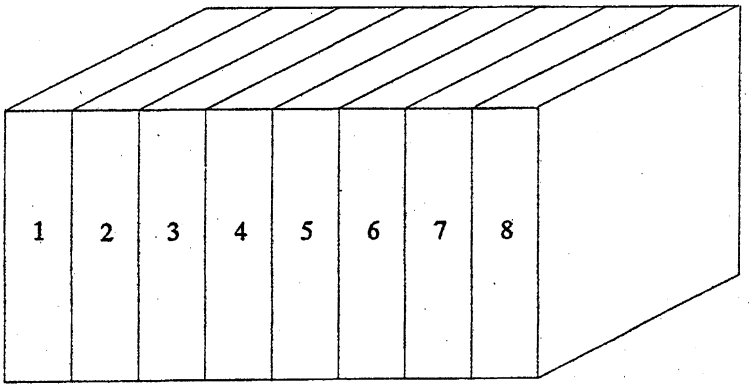

Fig. 3. A partition of the cube for 8 processors.

Simillar results are valid when processors can communicate concurrently. For the proposed basic data layout each processor must transmit $0.75(N-1)\times(N-1)$ elements. In the case of grid partition, which is given in Fig. 3, the number of such elements is equal to $2(N-1)\times(N-1)$.

5. Load balancing. In this section we consider the case of heterogeneous workstation cluster. Let's assume that performance rates of processors satisfy the following inequalities

$$w_1 \leqslant w_2 \leqslant \ldots \leqslant w_p.$$

In previous sections it was shown that implementation of LOD scheme (5) requires $O(N^3)$ floating-point operations and processors must communicate $O(N^2)$ elements. Hence Amdahl's Law suggests that for large problems the algorithm can be effectively parallelized and we can take into account only computational time when we solve a *load balancing* problem.

Let's denote by M_j the number of grid points assigned to jth processor. In order to get the equitable distribution of the total computational work we must solve the following optimization problem

$$\min_{M_i} \max_{1\leqslant j\leqslant p} t_j = t^*, \quad t_j = \frac{M_j}{W_j} \tag{12}$$

subject to the constraints

$$M_1 + M_2 + \cdots + M_p = (N-1)^3, \quad M_j > 0.$$

The solution of (12) is well known for real-valued M_j and it is defined as

$$\frac{M_1}{w_1} = \frac{M_2}{w_2} = \cdots = \frac{M_p}{w_p}. \tag{13}$$

Now we will finish the definition of the partitioning algorithm from Section 4. It is sufficient to consider one step of the algorithm, for example the first step. Suppose that according to the layout defined in Section 4 all processors are divided into two sets

$$Q_1 = \{q_1, q_2, \ldots, q_l\}, \quad Q_2 = \{q_{l+1}, q_{l+2}, \ldots, q_p\},$$

and both sets are not empty. We denote W_1, W_2 the total computational rates of processors, belonging to sets Q_1, Q_2, respectively:

$$W_1 = w_1 + w_2 + \cdots + w_l, \quad W_2 = w_{l+1} + w_{l+2} + \cdots + w_p.$$

The block size of grid points in x_3 direction, assigned to processors from the first group, is obtained by finding the integer valued solution of the optimization problem (12). It is given by

$$K_1 = \left\lfloor (N-1)\frac{W_1}{W_1 + W_2} \right\rfloor, \tag{14}$$

if

$$\frac{K_1 + 1}{W_1} \geqslant \frac{N - 1 - K_1}{W_2},$$

otherwise

$$K_1 = \left\lfloor (N-1)\frac{W_1}{W_1 + W_2} \right\rfloor + 1, \tag{15}$$

where $\lfloor z \rfloor$ denotes the greatest integer less than or equal to z. Such procedure is applied in all three directions. The example of computational grid partition for $p = 8$ processors in the case of heterogeneous cluster is given in Fig. 4.

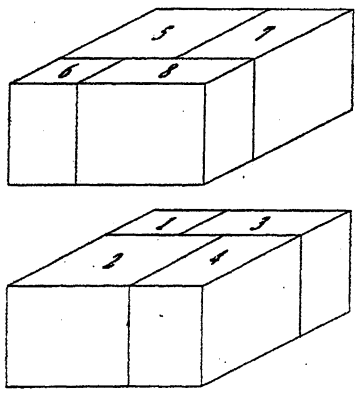

Fig. 4. Partition of 3D computational grid for 8 processors.

6. Pipelining method. In this section we consider a generalization of our algorithm. Here we assume that the number of processors $p \leqslant 64$. We assign

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Fig. 5. Assignment of the subdomains to 16 processors of a 4×4 grid.

the subcubes of the computational grid to 4 processors in each direction as shown in Fig. 5 for the first layer of processors. For the realization of LOD scheme (5) we apply the method of pipelining the computation (Hofhaus and van de Velde, 1996).

The forward sweep is started from both ends by two processors with only one tridiagonal system in each of the processors. As soon as these processors are finished with their part of Gaussian elimination of their first tridiagonal system they send the information to their neighbours and start the forward sweep on the second tridiagonal system, while the other two processors start working on their part of the first tridiagonal system. Thus after first step all processors will be active. The backward sweep is similar.

7. Numerical results. Some results of experiments performed with different clusters of workstations will be presented in this section. The test problem described in Section 2 was solved. We have used the following initial condition and right hand side function

$$u(x_1, x_2, x_3, 0) = \sin \pi x_1 \sin \pi x_2 \sin \pi x_3,$$
$$f(x_1, x_2, x_3) = 3\pi^2 \sin \pi x_1 \sin \pi x_2 \sin \pi x_3.$$

The LOD algorithm (5) was run on four grids: $(62 \times 62 \times 62)$, $(82 \times 82 \times 82)$, $(102 \times 102 \times 102)$, $(122 \times 122 \times 122)$ and 16 time steps were performed on each grid.

First we present results obtained for the homogeneous cluster of two MOTOROLA workstations. Our goal was to investigate the efficiency of the algorithm in unfavourable conditions when:

- the computational speed of wokstations is rather high,
- the communication rates are rather low (the computers were included into the cluster with commercial OMNITELNET and the distance between them was about 100 km).

We note that the best efficiency and speedup results can be achieved for homogeneous clusters of workstations with low computational speed and high communication speed.

In Table 1 we show the execution time T_E and parallel efficiency E_2, which is defined as

$$E_2 = \frac{T(1)}{2T(2)},$$

where $T(k)$ is the time required to execute the program on k processors.

Table 1. Execution times and efficiency indices for various space grids

N	62	82	102	122
T_E	0:33	1:12	2:15	3:54
E_2	0.82	0.89	0.94	0.96

Our second cluster was heterogeneous and included 2 SUN SPARC stations (with SOLARIS), HP 9000/720, 2 PC PENTIUM (with SOLARIS), PC AT/486 (with SOLARIS). All these workstations were clustered in one local network. The relative computational speeds of workstations are presented in Table 2.

Table 2. The relative computational speed of workstations

Workstation	SUNSPARC	PC AT/486	HP 9000	PC PENTIUM	MOTOROLLA
Speed	100	110	180	220	500

Firstly we treated this cluster as homogeneous cluster of workstations of the slowest type, using the homogenous partitioning algorithm. The obtained

execution times and efficiency indices E_6, are presented in Table 3 where the parallel efficiency is defined as

$$E_6 = \frac{T_{SUN}(1)}{6\,T(6)},$$

Table 3. Execution times and efficiency indices for homogeneous cluster of 6 workstations

N	42	62	82	102
T_E	0:29	1:05	2:15	3:54
E_6	0.50	0.76	0.86	0.91

We then solved LOD scheme (5) by taking into account the relative computational rates of workstations. The results are given in Table 4, where the parallel efficiency is defined as

$$E_6 = \frac{100\,T_{SUN}(1)}{W_6 \cdot T(6)} = \frac{T_{SUN}(1)}{10.35\,T(6)},$$

and W_6 is the total relative computational speed of the cluster.

Table 4. Execution times and efficiency indices for heterogeneous cluster of 6 workstations

N	42	62	82	102
T_E	0:21	0:44	1:29	2:23
E_6	0.40	0.65	0.79	0.87

We see that parallel efficiency degrades a little due to higher computational speeds of workstations. Even with the lower efficiency, however, execution times are shorter with heterogeneous cluster than for homogeneous cluster.

8. Conclusions. We have demonstrarated the effectiveness of parallel algorithms for solving 3D diffusion problem on distributed memory computers. The

problem is approximated by LOD scheme and solved by the 2 way Gaussian elimination method.

The domain decomposition method is proposed for heterogeneous clusters of workstations which preserves the load balancing of computers. The basic method can be implemented on 8 processors and the pipelining metod is used to enlarge the number of processors till 64. The performance of the algorithm is demonstrated for two clusters of workstations.

REFERENCES

Čiegis, R., and K. Kiškis (1994a). On the stability of additive finite difference schemes with respect to boundary conditions. *Differenc. Uravnenya,* **30**(4), 480–487 (in Russian).

Čiegis, R., and K. Kiškis (1994b). On the stability of LOD difference methods with respect to boundary conditions. *Informatica,* **5**, 297–323.

Demmel, J.W., M.T. Heath and H.A. van der Vorst (1993). Parallel numerical linear algebra. *Acta Numerica,* 111–197.

Freeman, T.L., and C. Phillips (1991). *Parallel Numerical Algorithms.* Prentice Hall, New York, London, Toronto, Sydney, Tokyo, Singapure.

Geist, A., A. Bequelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderman (1993). *PVM 3.0 Users Guide and Reference Manual.* ORNL/TM–12187, USA.

Ho, C., and S. Johnson (1990). Optimizing tridiagonal solvers for alternating direction methods on Boolen cube multiprocessors. *SIAM J. Sci. Statist. Comput.,* **11**(4), 563–592.

Hofhaus, J., and E.F. van de Velde (1996). Alternating direction line-relaxation methods on multicomputers. *SIAM J. Sci. Comput.,* **17**(2), 454–478.

Johnson, S.L., Y. Saad and M. Schultz (1987). Alternating direction methods on multiprocessors. *SIAM J. Sci. Statist. Comput.,* **8**(5), 686–700.

Lin, W.Y., and C.L. Chen (1994). A parallel algorithm for solving tridiagonal linear systems on distributed-memory multiprocessors. *International Journal of High Speed Computing,* **6**(3), 375–386.

Marchuk, G.I (1990). Splitting and alternating direction methods. In P.G. Ciarlet, J.L. Lions (Eds.), *Handbook of Numerical Analysis 1.* North–Holland, Amsterdam. pp. 197–462.

Ortega, J.M. (1988). *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, New York and London.

Sommeijer, B.P., and J. Kok (1994). Implementation and performance of a three–dimensional numerical transport model. *Report NM–R9402,* CWI, Amsterdam.

Zlatev, Z., J. Christensen, J. Moth and J. Waśniewski (1991). Vectorizing codes for studying long-range transport of air pollutants. *Math. Comput. Modelling,* **15**, 37–48.

Zlatev, Z., and J. Waśniewski (1992). Large scale computations in air pollution modelling. *Report UNIC–92–07,* Denmark.

Received September 1996

R. Čiegis has graduated from the Vilnius University Faculty of Mathematics in 1982, received the Degree of Doctor of Physical and Mathematical Sciences from the Institute of Mathematics of Byelorussian Academy of Sciences in 1985 and the Degree of Habil. Doctor of Mathematics from the Institute of Mathematics and Informatics, Vilnius in 1993. He is a head of Mathematical Modelling Department, Institute of Mathematics and Informatics. R. Čiegis is also a Professor at the Kaunas Vytautas Magnus University and a Professor and a head of Mathematical Modelling Department of Vilnius Technical University. His research interests include numerical methods for nonlinear PDE, parallel numerical methods and numerical modelling in physics, biophysics, ecology.

J. Šimkevičius is a researcher at the Department of Mathematics and Statistics of Vytautas Magnus University, Kaunas, Lithuania. Scientific interest include parallel computing, wavelet transform, digital signal processing.

J. Waśniewski graduated at the University of Warsaw, Department of Mathematics in 1995. He received there his M.S and PhD. He worked at several universities and research companies in several countries (Poland, Canada, Denmark, UK and USA). At present he is the Associate Professor and Senior Researcher at the Danish Computing Centre for Research and Education (UNI-C) in Lyngby, Denmark. His research interest is: numerical analysis software, basic linear algebra subprograms, sparse matrices and parallel scientific computing. He wrote many scientific publications in the international journals and four books.

TRIMAČIO ŠILUMOS LAIDUMO UŽDAVINIO SPRENDIMAS LYGIAGREČIUOJU KOMPIUTERIU SU PASKIRSTYTĄJA ATMINTIMI

Raimondas ČIEGIS, Juozas ŠIMKEVIČIUS, Jerzy WAŚNIEWSKI

Nagrinėjami skaitiniai uždavinio sprendimo algoritmai, kurie yra efektyvūs lygiagretiesiems kompiuteriams su paskirstytąja atmintimi. Pateikti algoritmai, kurie tinkami, kai procesorių skaičius yra ne didesnis už 8. Panaudojant konvejerio principą sukonstruota algoritmo modifikacija, leidžianti panaudoti iki 64 procesorių. Heterogeniškam kompiuterių klasteriui sprendžiamas tolygaus procesorių apkrovimo uždavinys, pateiktas diskrečiojo tinklo mazgų paskirstymo tarp procesorių algoritmas. Pateikti skaitinių eksperimentų, atliktų naudojant PVM, rezultatai.