Computing and Informatics, Vol. 28, 2009, 139–150

PARALLEL SIMULATION OF A FLUID FLOW BY MEANS OF THE SPH METHOD: OPENMP VS. MPI COMPARISON

Paweł Wróblewski, Krzysztof Boryczko

Department of Computer Science
Faculty of Electrical Engineering, Automatics, Computer Science and Electronics
AGH University of Science and Technology, Kraków
e-mail: {pawel.wroblewski, boryczko}@agh.edu.pl

Revised manuscript received 29 June 2008

Abstract. The SPH method for simulating incompressible fluids is presented in this article. The background and principles of the SPH method are explained and its application to incompressible fluids simulations is discussed. The parallel implementation of the SPH simulation with OpenMP and MPI environments are demonstrated. Both models of parallel implementation are analyzed and discussed. The comparison of both models is performed and discussed, as well as their results.

Keywords: Computational fluid dynamics, SPH, parallel computing, OpenMP, MPI

1 INTRODUCTION

A number of existing computer methods can be used for simulating phenomena from the real world. Many of these phenomena, very important in the contemporary science and engineering, are related to fluid mechanics. These phenomena refer to different spatio-temporal scales, starting from micro scale, through meso scale to macro scale. Very interesting processes, e.g. turbulences, wall-fluid interactions, free surface behavior, take place in the domain between these scales. However, neither correct nor efficient methods have been available for this area yet. The SPH method is a very popular particle method for simulating processes in the macro scale [9]; it also seems to be possible to simulate phenomena from the domain between macro

and meso scales. In this paper the background and principles of the SPH method are shown along with its variants, depending on the target application. Adaptations of this method to incompressible fluids simulations are also presented.

The SPH method has been implemented for parallel execution by means of the OpenMP and MPI environments. The parallel implementation allows simulation of very large systems consisting of several millions of particles. The paper presents the details of the implementation and provides an analysis of the efficiency of the program. It shows the possible approaches for parallel implementation of simulation exploiting particle methods, which has arised from molecular dynamics. The implemented algorithm has been employed for simulating the phenomenon of fluid flow in an elongated vessel.

2 PARTICLE METHODS

The particle method is a computational model in which the simulated fluid is modeled by a number of particles. The particles interact mutually and follow the motion according to Newton's dynamics. They are located inside a computational box, as presented in Figure 1. The essential part of the simulation source code is the main loop which is executed assumed number of iterations (see Figure 2). In each iteration new positions of SPH particles are calculated, the forces between particles are evaluated and other necessary calculations are processed.

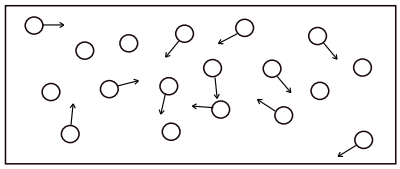

Fig. 1. Particles moving inside the computational box

If forces used in a simulation are of the short-range type, it is possible to exploit the cubic cells structure of the computational box. It rapidly accelerates calculations by omitting pairs of particles, for which the force is equal to zero [1]. However, the size of a cubic cell must be greater than or equal to the maximum distance in the simulation, for the which force is different than zero (see Figure 3). The computational box with the cubic cells structure is presented in Figure 4.

3 SMOOTHED PARTICLE HYDRODYNAMICS

The SPH method is one of many existing particle methods. It was created in order to simulate astrophysical phenomena [6, 4]. The main idea of the method is a field approximation in the set of points from space. The hydrodynamical forces satisfying the Navier-Stoke's equation are calculated in these points – the SPH particles – and

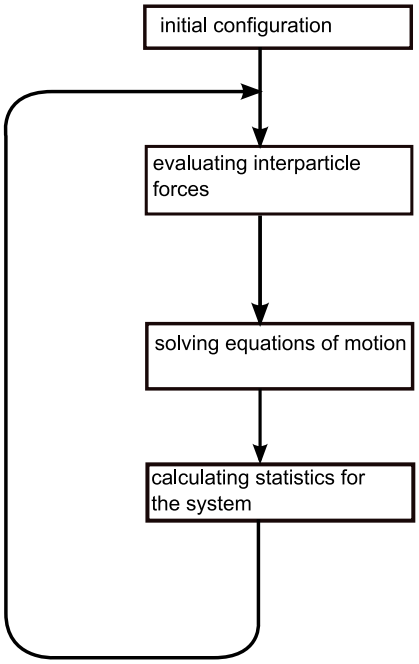

Fig. 2. The algorithm of a particle method

with such background the equations of motion are solved. The approximation procedure uses the kernel function, which vanishes at infinity and its integral is equal to unity. From the theoretical point of view one could choose the Gaussian bell-shaped function; however, in practice it is common to use a spline function with compact support. The authors used the kernel function proposed in [7]. The approximation procedure applies not only to the hydrodynamical forces, but also other quanti-

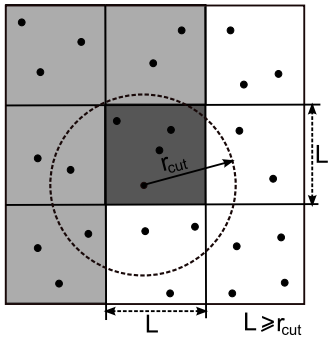

Fig. 3. A comparison of cubic cells' size L and cut-off radius r_{cut}

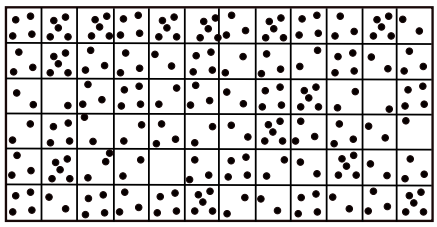

Fig. 4. The computational box consisting of cubic cells

ties referred to by the modeled phenomenon. As an example, the approximation equation for a density is presented below [7]:

$$\rho_i = \sum_j m_j W_{ij}, \tag{1}$$

where m_j – the mass of particle j, $W_{ij} = W(r_{ij}, h_{ij})$ – the kernel function evaluated for particles i and j. The sum in the above equation runs on all particles from the system. In practice, however, if the support of kernel function is compact, it is enough to count only those SPH particles, for which the kernel function is non-zero. When using the kernel function which vanishes for interparticle distances larger than the assumed cut-off radius, it is possible to use the structure consisting of cubic cells.

Every SPH particle follows the acceleration given by the formula:

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{ij}, \tag{2}$$

where P_i – the pressure at point i, ρ_i – the density at point i and Π_{ij} – viscosity part of the force. One can find the full derivation of equation (2) in [5]. Beside the force acting between fluid particles, it is also necessary to incorporate forces acting between fluid particles and the walls into the model. In this paper the authors use a wall consisting of particles, and the corresponding force is very similar to the one given in [9]. It is given by the formula

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \sum_j \left(\frac{c_0}{10} \right)^2 \frac{\Gamma(r_{ij}/r_{cut})}{r_{ij}} \frac{m_j}{m_i + m_j}, \tag{3}$$

where

$$\Gamma(q) = \begin{cases} \frac{2}{3}, & \text{if } x < \frac{1}{3}, \\ 2(2q - 3q^2), & \text{if } \frac{1}{3} < q < \frac{1}{2}, \\ 2(1-q)^2, & \text{if } \frac{1}{2} < q < 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

4 INCOMPRESSIBLE SPH

The standard SPH formula for density (1) is useless when modeling the fluids with free surface. If this equation is applied, the density in the vicinity of the surface changes continuously from the value assumed for all particles to the value of zero on the distance of $2h$, which is obviously a discord with the experiments. In the case of such fluids another formula derived from the SPH approximation is used:

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \sum_j m_j(\mathbf{v}_i - \mathbf{v}_j) \cdot \nabla_i W_{ij}, \tag{5}$$

which evaluates only the rates of change of the density. Applying this equation requires the initialization of the density values at the beginning of the simulation.

The incompressible character of the fluid is modeled by an appropriate equation of state, which is used for evaluating the pressure values in Equation (2). The authors use the equation of state given by [9]:

$$P = \frac{\rho_0 c_0^2}{7}\left(\left(\frac{\rho}{\rho_0}\right)^7 - 1\right). \tag{6}$$

When modeling incompressible fluids it is very problematic to choose a proper timestep. If real values of speed of sound c are applied, the timestep is too small for any practical application. Therefore it is convenient to use a value of c several orders of magnitude smaller than the real one. This approach accelerates the calculation significantly, and does not influence the results [2].

5 PARALLELIZATION

Simulations of complex systems with particle methods require the use of computer systems with big computational power. The large memory complexity of the problems coerces the systems to have respectively large memory supply. Therefore, it is common to use multiprocessor architecture with shared or distributed memory. The former are relatively cheap, however the message passing paradigm suitable for it needs great programming experience and time effort. The latter are relatively expensive but the programming environments available for it allow generally fast parallel implementation of the problem.

5.1 Parallel Implementation with OpenMP

In case of computers with a multiprocessor architecture with shared memory it is appropriate to implement a parallel algorithm whenever possible, by means of the OpenMP environment. The OpenMP environment allows for parallel implementation of the problem by means of OpenMP directives placed into the source code. These directives serve as instructions for a compiler which blocks and loops in the

source code are to be executed in parallel by available processors. The directives also define which variables and how are to be shared, how to parallelize the loop, etc.

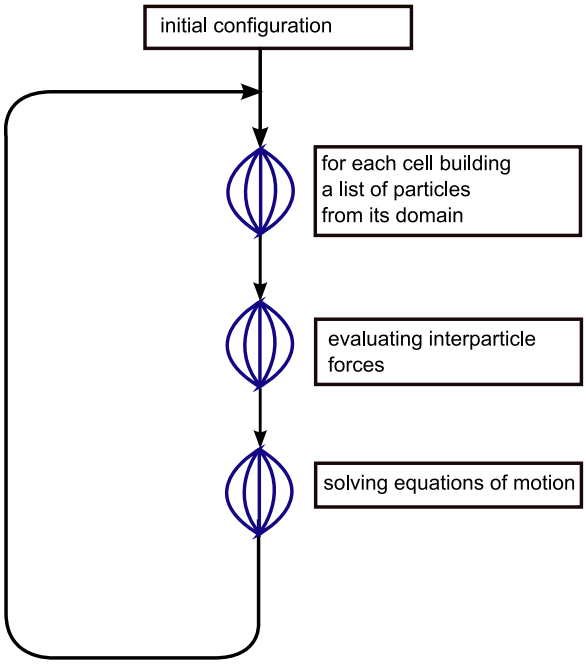

Fig. 5. The particles method algorithm parallelized with OpenMP environment

The loop was divided into several blocks, which were implemented to execute in parallel by means of OpenMP directives. The blocks of the main loop implemented to run in parallel are:

- evaluating the new positions and velocities of the particles, assigning particles to cells
- gathering information about particles from the adjacent cells
- evaluating of the forces (accelerations) for interacting pairs of particles.

In order to assess the efficiency of the parallel OpenMP implementation of the simulation the authors have prepared four versions of the simulation of breaking dam phenomenon, described in [13]. These four versions of the simulation differ by the number of particles in the system (accordingly 117 045 and 1 025 325) and the average number of the neighbors for each particle (accordingly 113 and 524). Each version of the simulation was run several times, each time on a different number of processors, and thus it was possible to evaluate the relative efficiency of the parallel implementation of the simulation. The results are presented in Figure 6.

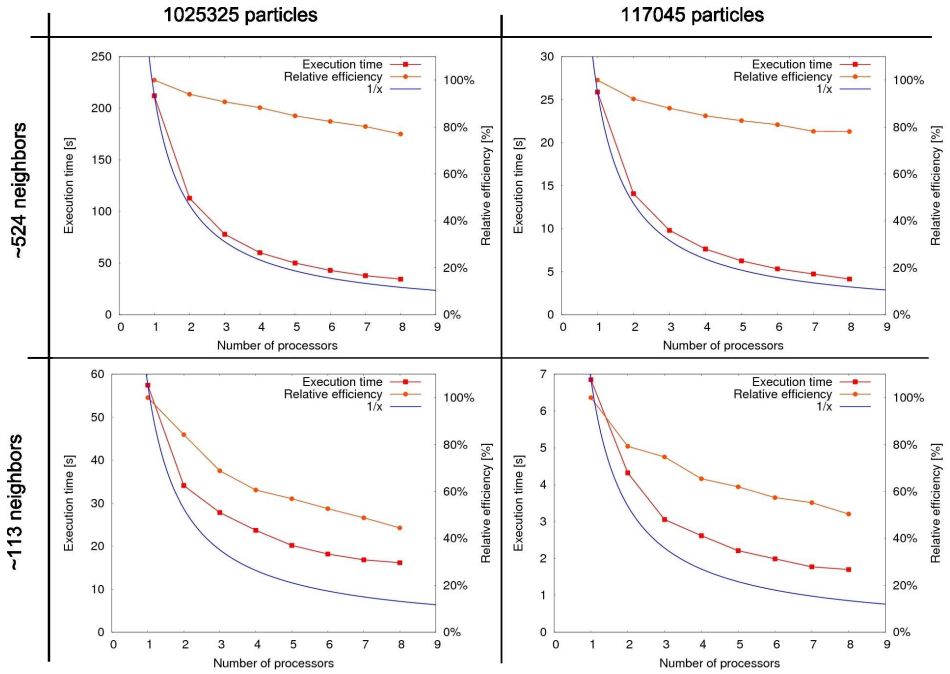

Fig. 6. The results of OpenMP parallelization

Four plots can be found in the figure, each of them presenting the real time of the main loop execution, the ideal case labeled as $1/x$ and the relative efficiency of the simulation. From these pictures one can notice, that the relative efficiency of the simulation depends on the average number of neighbors, and does not depend on the size of the whole system. When the average number of neighbors is large, then forces' calculations are much more time consuming. Since they are executed in the parallelized block of the program, the ratio of time consumed by the parallelized part of the program to the time consumed by the non-parallelized part of the program is higher than in the case when the number of neighbors for each particle is small. Therefore, this explanation shows that results depicted in Figure 6 are in good agreement with expectations. The relative efficiency of the simulation is not influenced by the size of the system, which, actually, influences the time consumed by a single iteration of simulation.

5.2 Parallel Implementation with MPI

The computer architecture with distributed memory system constrains the possible approaches to the message passing paradigm and domain decomposition model. The most popular environment for implementing this paradigm is Message Passing Interface, which was used in this research. In the domain decomposition approach the

computational box is divided into exclusive subdomains, each for every computational node. Every node runs the simulation on its own with the data assigned to it. While particles move in the computational box and interact mutually, it is necessary to send the particles' information between nodes. This is done by sending messages by calling the MPI functions, which is presented in Figure 7. The decomposition of the computational box is depicted in Figure 8, where cells to be sent are marked by color.

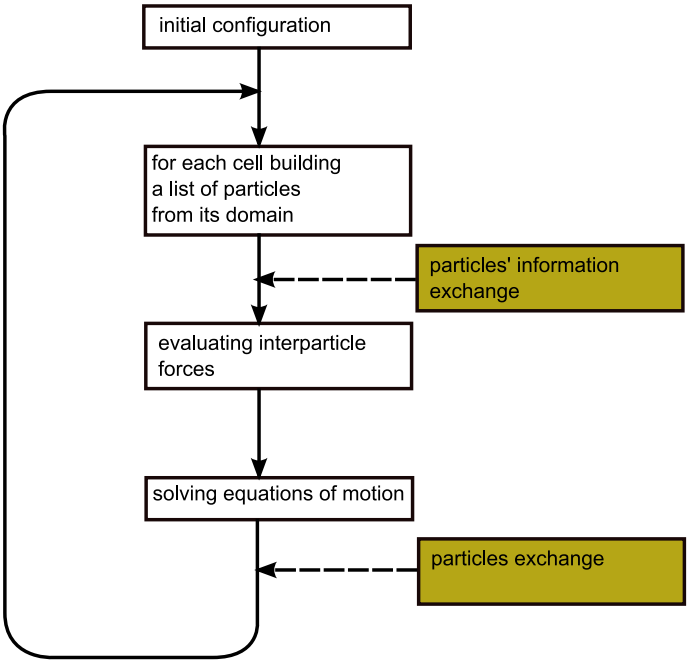

Fig. 7. The particles method algorithm parallelized with MPI environment

In order to find the efficiency of the parallel MPI implementation of the simulation the authors prepared three versions of the simulation of flow in an elongated vessel. These three versions of the simulation differ in the number of particles per node (accordingly: 16 384, 55 296 and 131 072). Each version of the simulation was run several times, each time on different number of processors, and thus it was possible to evaluate the relative efficiency of the parallel implementation of the simulation. The results are presented in Figures 9 and 10.

6 RESULTS

In the article the authors present the domain decomposition for the MPI implementation, as well as the algorithm decomposition for OpenMP implementation. The

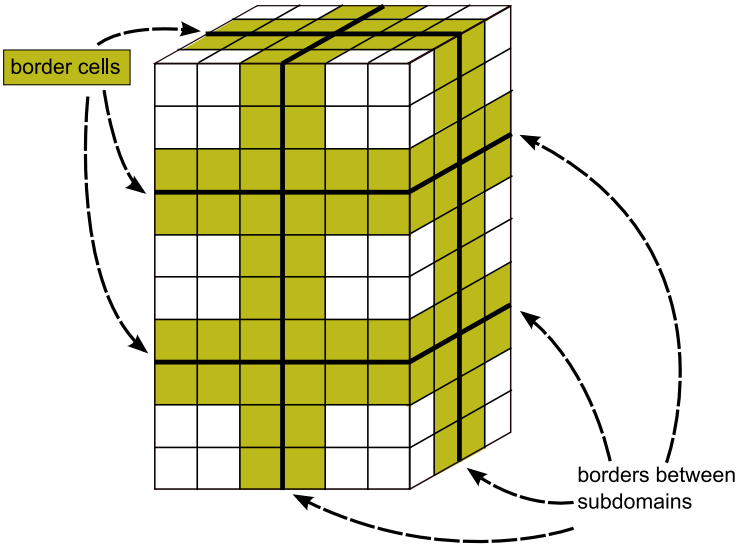

Fig. 8. The decomposition of the computational box into subdomains

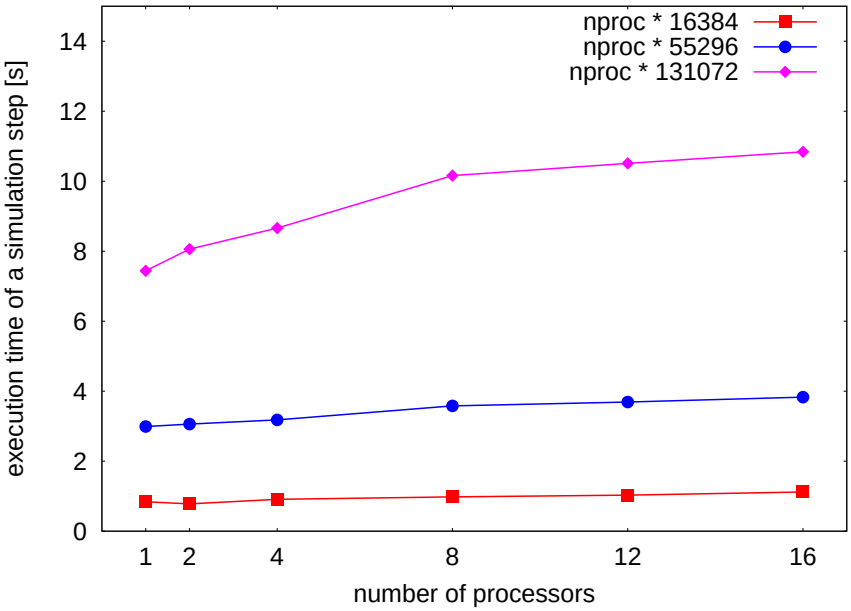

Fig. 9. The execution time of run MPI simulations

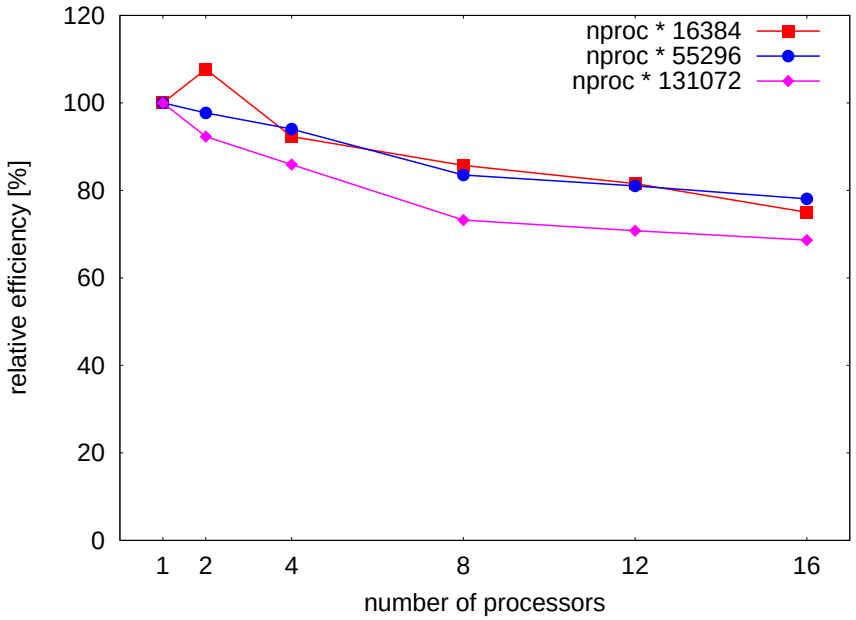

Fig. 10. The relative efficiency of run MPI simulations

comparison between two programming paradigms was performed on the SGI Altix 3 700 machine. This machine is a cache coherent, shared memory multiprocessor system, in which memory is physically distributed, but logically shared and kept coherent automatically by hardware. This system allows to run programs created with both parallel programming paradigms: message passing and shared memory. The comparison was performed for the varying number of processors and for the same simulations of the flow in an elongated vessel. The result of the comparison is presented in Figure 11 showing that the relative efficiency of the MPI implementation is significantly greater than the relative efficiency of the OpenMP implementation. It seems, therefore, that the approach, in which a programmer has more detailed influence on the parallelization is still more efficient than the approach, in which the parallelization is performed almost automatically. These results are in good agreement with the ones presented in the literature [1, 3], where similar tests where performed.

7 CONCLUSIONS AND FUTURE WORK

In future the authors plan to focus on the optimization of both implementations. We believe that it is still possible to improve the efficiency of implementations of both approaches. For the algorithm decomposition approach this can be done by

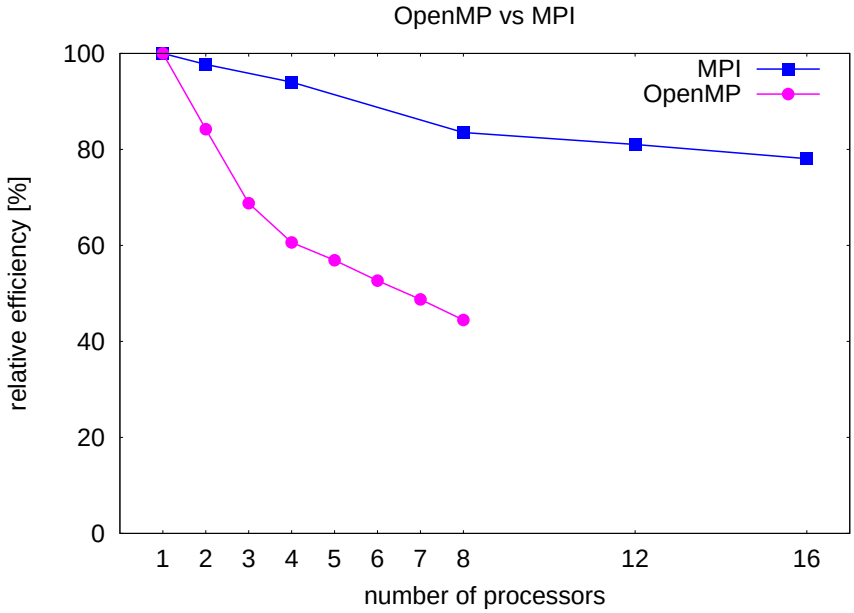

Fig. 11. The comparison of relative efficiency between OpenMP and MPI approach

a proper data organisation in the simulation program. For the approach with domain decomposition this might be done by improving the decomposition adaptation to a modeled phenomenon.

Acknowledgments

This research is financed by the Polish Ministry of Science and Higher Education, Project No. 3T11F01030.

REFERENCES

[1] Boryczko, K.—Dzwinel, W.—Yuen, D.: Parallel Implementation of the Fluid Particle Model for Simulating Complex Fluids in the Mesoscale. Concurrency and computation: practice and experience, Vol. 14, 2002, pp. 137–161.

[2] Colagrossi, A.—Landrini, M.: Numerical Simulation of Interfacial Flows by Smoothed Particle Hydrodynamics. J. Comp. Phys., Vol. 191, 2003, pp. 448–475.

[3] Couturier, R.—Chipot, C.: Parallel Molecular Dynamics Using OpenMP on a Shared Memory Machine. Computer Physics Communications, Vol. 124, 2000, pp. 49–59.

[4] Gingold, R. A.—Monaghan, J. J.: Smoothed Particle Hydrodynamics – Theory and Application to Non-Spherical Stars. Mon. Not. R. Astr. Soc., Vol. 181, 1977, pp. 375–389.

[5] Liu, G. R.—Liu, M. B.: Smoothed Particle Hydrodynamics: A Meshfree Particle Method. World Scientific, 2003.

[6] Lucy, L. B.: A Numerical Approach to the Testing of the Fission Hypothesis. Astron. J., Vol. 82, 1977, pp. 1013–1024.

[7] Monaghan, J. J.: Smoothed Particle Hydrodynamics. Annu. Rev. Astron. Astrophys., Vol. 30, 1992, pp. 543–574.

[8] Monaghan, J. J.: Simulating Free Surface Flows with SPH. J. Comp. Phys., Vol. 110, 1994, pp. 399–406.

[9] Monaghan, J. J.: Smoothed Particle hydrodynamics. Rep. Prog. Phys., Vol. 68, 2005, pp. 1703–1759.

[10] Morris, J. P.: Simulating Surface Tension With Smoothed Particle Hydrodynamics. Int. J. Numer. Meth. Fluids, Vol. 33, 2000, pp. 333–353.

[11] Nugent, S.—Posch, H. A.: Liquid Drops and Surface Tension With Smoothed Particle Applied Mechanics. Phys. Rev. E, Vol. 33, 2000, pp. 333–353.

[12] Tartakovsky, A.—Meakin, P.: Modeling of Surface Tension and Contact Angles With Smoothed Particle Hydrodynamics. Phys. Rev. E, Vol. 72, 2005, p. 02630.

[13] Wróblewski, P.—Boryczko, K.—Kopeć, M.: SPH – A Comparison of Neighbor Search Methods Based on Constant Number of Neighbors and Constant Cut-Off Radius. TASK Quart., Vol. 11, 2007, pp. 275–285.

Paweł Wróblewski received his Master's degree from AGH University of Science and Technology in Computer Science in 2003. He is currently working on his Ph. D. under the supervision of Prof. Krzysztof Boryczko. His area of interest includes parallel programming and fluid simulations.

Krzysztof Boryczko obtained his Ph. D. degree in Computer Science in 1992 from Department of Computer Science, AGH University of Science and Technology, Poland where he is now an associate professor. His research interests focus on large-scale simulations with particle methods. He is also interested in scientific visualization, feature extraction and clustering methods for analysis of simulation data.