6th European Conference on Computational Mechanics (ECCM 6)
7th European Conference on Computational Fluid Dynamics (ECFD 7)
1115 June 2018, Glasgow, UK

ON CPU AND GPU PARALLELIZATION OF VM2D CODE FOR 2D FLOWS SIMULATION USING VORTEX METHOD

KSENIIA S. KUZMINA[1,2], ILIA K. MARCHEVSKY[1,2] AND EVGENIYA P. RYATINA[1,2]

[1] Bauman Moscow State Technical University
105005, Russia, Moscow, 2-nd Baumanskaya st., 5

[2] Ivannikov Institute for System Programming of the RAS
109004, Russia, Moscow, Alexander Solzhenitsyn st., 25

kuz-ksen-serg@yandex.ru, iliamarchevsky@mail.ru, evgeniya.ryatina@yandex.ru

Key words: Vortex Methods, VM2D Code, CPU, GPU, MPI, OpenMP, CUDA

Abstract. VM2D is an open-source software being developed by the authors for two-dimensional incompressible flows simulations around airfoils. VM2D is based on meshless Lagrangian vortex methods. The main operations of the algorithm are pointed out, and the estimations of their computational complexity are given. Two model problems with different parameters are considered in order to analyze the ratio between computational complexities of the operations. Parallel algorithms are implemented for all time-consuming operations to perform the simulations on CPU and GPU. Test problems show that VM2D is efficiently parallelized; the accelerations achieved on GPUs are comparable to acceleration on hundreds and even thousands of CPU cores.

1 INTRODUCTION

In many engineering applications Fluid–Structure Interaction problems appear, when it is necessary to calculate the loads acting on the construction being immersed into the flow. Such problems become especially complicated when we deal with essentially unsteady flow with intensive vortex shedding: in this case, it is impossible to use approximate models based on stationary aerodynamic characteristics, so, the only way is to perform direct numerical simulation of the flow. Moreover, when designing a structure, it is necessary to perform a large number of numerical experiments; and in spite of large number of commercial and open-source codes for flow simulation based on different numerical methods, there is relevant problem of approximate but efficient methods developing for direct numerical simulation in FSI problems.

Most of the existing methods for numerical simulation in FSI problems belong to the class of Eulerian or hybrid Eulerian/Lagrangian methods and require mesh generation in flow domain; in case of moving or deformable body the mesh should be modified at every time step (except immersing boundary methods [1, 2]). In case of small body displacements we can do with mesh deformation; but if the body has significant displacement or rotation, it is necessary to reconstruct the mesh, at least after several time steps, what reduces essentially the efficiency of the numerical method. From this point of view, the class of meshless Lagrangian methods which doesn't require fluid domain meshing and permits arbitrary displacements of the body is more preferable.

In the present research the efficiency of meshless Lagrangian vortex method and its software implementation is investigated. It should be noted that the range of applicability of vortex methods is restricted by rather low Reynolds numbers and incompressible flow model. However, many problems which arise in engineering practice, satisfy such conditions and can be efficiently solved numerically using vortex methods.

The main idea of vortex methods [3, 4, 5, 6, 7] is the considering of the vorticity as the primary calculated variable. The body is replaced with three thin sheets on its surface: an attached vortex sheet, an attached source sheet and a free vortex sheet. The last one models vorticity flux from the body surface; it is discretized into separate vortex elements which, in turn, become part of the vortex wake behind the body. The intensities of the attached sheets are expressed through the velocity of the body surface; free vortex sheet intensity can be found from the no-slip boundary condition satisfaction on the body surface. So, at every time step we should solve boundary equation, which follows from boundary condition, and find the intensity of the free vortex sheet; and then this vorticity sheds to the vortex wake.

It should be noted that vortex methods are especially efficient in comparison with mesh methods when we simulate the external flow around the structures, since in this case the boundary condition of perturbations decay at infinity is satisfied automatically and there is no need to limit the computational domain artificially and set some boundary conditions on its outer boundaries. Moreover, in most cases of the external flow simulation the domain with non-zero vorticity is localized around and behind the body, so computational resources can be "concentrated" in this domain due to vorticity as chosen as primary calculated variable.

Vortex methods are well-investigated and they are rather popular in engineering community. There are various modifications of vortex methods both for numerical simulation in 3D and 2D problems. Nevertheless, vortex methods are still not implemented in any known software packages: both commercial and freely distributed. Of course, many researchers have their own "in-house" codes, however, such software is usually used only by small groups of scientists working also on their development. This fact negatively effects the popularity of vortex methods.

Therefore, having some research experience in vortex methods, as well as the experience of their software implementation in our "in-house" software [8], the authors have started to develop new software package **VM2D — Open source code for two-dimensional flows simulation using vortex methods** [9]. In [9] the general structure of the VM2D code and main approaches implemented there are described. Current paper doesn't aim to re-describe VM2D software; the main purpose is to analyze the efficiency of the VM2D code and its parallel properties.

The VM2D software is based on the method of Viscous vortex domains (VVD), developed by prof. G.Ya. Dynnikova [7]. Well known models and numerical algorithms as well as the authors' advancements are implemented in the VM2D code [10, 11, 12, 13]. At the present moment, it is expected that the code is used for external flows simulation, however, problems solution which require internal flow simulation is also possible, maybe after some modification of the algorithm for higher efficiency achievement.

It should also be noted that in order to provide the simplicity of program modification, both by the authors and other users, and the possibility of new approaches and features addition, we have adhered to the principles of the object-oriented approach, and the VM2D code has modular structure. The main purpose of the VM2D development is to create a tool for the engineering problems numerical solution in a short time, so, it is important to provide the possibility of computations performing in parallel mode. In this paper, we study the efficiency of parallel algorithms in the VM2D.

2 MAIN COMPUTATIONAL BLOCKS IN THE VM2D ALGORITHM

There are 7 basic computational blocks in the vortex method algorithm implemented in the VM2D. Their detailed descriptions one can find in [8, 10], here we will give only their brief description and point out the main operations in these blocks and estimate their computational complexities, which depend primarily on the airfoil discretization.

1. *Vorticity generation on the airfoil surface.* In order to calculate the free vortex sheet intensity on the airfoil surface line, we solve the integral equation, which follows from the no-slip boundary condition on the airfoil surface line. There are two approaches to derive such integral equation: by satisfying the boundary condition for normal or tangential components of flow velocity [14]. The right-hand side of these integral equations depends on the velocity, generated on the airfoil surface by all the vortex elements, which model the vortex wake in the flow. All the vortices contributions computation makes this procedure time-consuming. For the numerical solution of the integral equation, the airfoils are usually approximated by polygons consist of rectilinear panels (Fig. 1); then linear algebraic equations system is generated according to the chosen numerical scheme. There are number of different numerical schemes for the integral equation approximation, they have different computational complexities (with regard to coefficients computations) and lead to linear systems of different sizes, but in all the known scheme system size is commeasurable with number of panels. In [12] a hierarchy of such schemes is described, all of those schemes are based on the ideas of Galerkin approach, continuous or discontinuous. In the current version of the VM2D code the numerical scheme with constant basis and projection functions in the framework of tangential components approach is used, as well as some less accurate schemes.

In order to estimate the computational complexities of the operations, we use hereinafter the following designations:

N is number of vortex elements in the flow domain;

n is total number of panels on the surface lines of all the airfoils.

Computational complexities of the operations of this block are the following:

- the matrix coefficients computation: $Q_1 = 83n^2$,

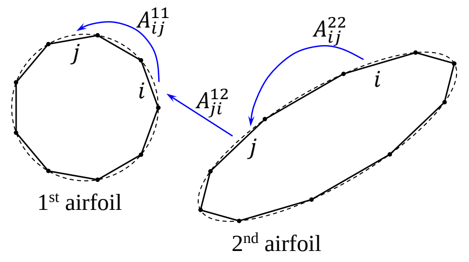

Figure 1: Two airfoils approximated with rectilinear panels

- the right-hand side vector computation: $Q_2 = 30Nn + 85n^2$,
- linear system solving: $Q_3 = n^3/3$.

In the most general case of flow simulation around m deformable airfoils, it is necessary to recompute and solve the linear system at every time step of the simulation. The structure of the matrix can be schematically represented as the following:

$$\begin{pmatrix} A^{11} & A^{12} & \dots & A^{1m} \\ A^{21} & A^{22} & \dots & A^{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A^{m1} & A^{m2} & \dots & A^{mm} \end{pmatrix},$$

where blocks A^{kk} express the self-influence of the k-th airfoil, and blocks A^{kp} express the contribution of the vorticity, being generated on the surface of the p-th airfoil, to limit value of the flow velocity on the k-th airfoil surface line.

The coefficients which form the corresponding blocks, depend on the relative positions of the panels only; therefore, in case of non-deformable (rigid) airfoils maintaining their positions relative to each other, the system matrix remains constant, that allows to calculate all the coefficients only once, then reverse the matrix (only at the first time step, computational complexity $Q_{rev} = n^3$), and solve the system by its multiplying the right-hand side ($Q_{mult} = n^2$ operations). In case of non-deformable airfoils being moved relative to each other, diagonal blocks A^{kk} remain constants, while the other (non-diagonal) blocks should be recalculated.

2. *Velocities computation.* The velocities of the vortex elements within the VVD approach [7] consist of two components: convective velocities and diffusive (cased by viscosity) velocities. When calculating the convective velocity, it is necessary to take into account the mutual influence of all vortex elements (i.e., to calculate the contributions of all the other vortices), according to the Bio — Savart law. The diffusive velocities are also calculated by taking into account the mutual influence of all vortex elements.

Computational complexities of these operations are the following:

- convective velocities computation: $Q_4 = 6N^2 + 8Nn$,
- diffusive velocities computation: $Q_5 = 9N^2 + 14Nn$.

Note, that complexity of diffusive velocities computation can be reduced by taking into account that vortices contributions to the diffusive velocity decrease exponentially with distance increase, so in practice it is necessary to calculate the influences only from the vortex elements, placed not very far one from others.

It also should be noted, that the surfaces of the airfoils also make a contribution to diffusive velocity of vortices in the flow. It should be taken into account only for vortices which are placed close to the airfoil surface, so the computational complexity of this procedure is proportional to n^2 and it can be neglected since $n \ll N$. In practice, however, this operation is important and in order to achieve high efficiency of parallel implementation, it also should be parallelized.

3. *Hydrodynamic loads computation.* In this block we compute hydrodynamic loads (forces and torque) acting the airfoils. For such purposes it is possible to use integral formulae, which are derived by prof. G.Ya. Dynnikova and adapted to several types of problems being solved by using vortex methods [15]. Note, that computational complexity of this block is much less in comparison with the other operation, so we will not take it into account.

4. *Vorticity evolution.* In this block vortices in the vortex wake are just being transferred along the calculated velocity field (recall, that it is superposition of convective and diffusive velocity fields). We use explicit Euler integration method, so it is necessary just to multiply vortices velocities by the time step value and add it to the current vortices positions. So the computational complexity of the block 4 is also much less than for the other operations, and normally it can be neglected.

5. *No-throw control.* This operation is necessary to exclude vortex elements that penetrate the airfoil. Its complexity depends on the implementation; in the first approximation $Q_6 \sim n^2$, and the proportionality coefficient is of the order of 10.

6. *Vortex wake restructuring.* Closely placed vortex elements can be merged, and vortex elements which move far away from the airfoil, can be excluded. The computational complexity of this algorithm, as calculations show, is of the order of $Q_7 \sim N^2$, the proportionality coefficient is relatively small.

Thus, the computational complexity of the last two operations, although it may be high, is much lower than the total complexity of the other operations, so their complexities can be taken into account approximately: we set $Q_6 = Q_1$ and $Q_7 = 0.2\,Q_4$.

3 MODEL PROBLEMS DESCRIPTION

In order to estimate the possible ratios of computational complexities of the above mentioned operations for different types of problems, we consider two model problems:

Problem 1. *Hydroelastic oscillations simulation for circular cylinder.* We consider flow around movable circular cylinder when vortex sheet on the surface line of cylinder is modeled with $n_{p0} = 200$ vortex elements. We assume that the vortex wake after the airfoil is simulated by $N_0 = 10\,000$ vortex elements, and number of time steps is $T_0 = 30\,000$. Such estimates are taken from practical simulation and correspond to the parameters of the real algorithm.

In order to improve the accuracy of simulation, which is necessary for flow simulation

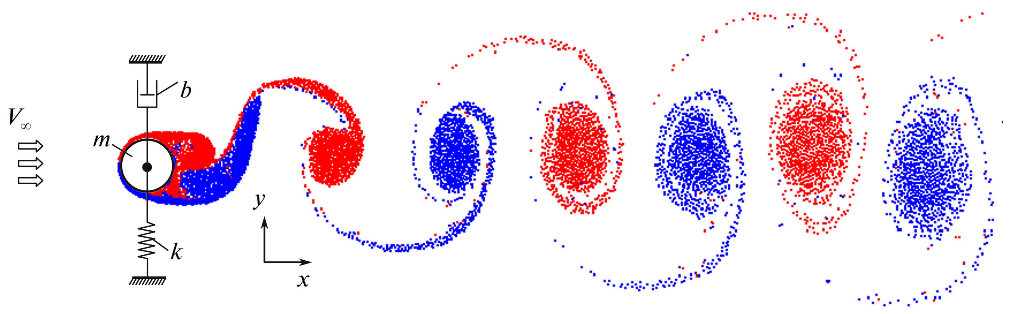

Figure 2: Model problem 1 statement

at rather high Reynolds numbers (number of vortices $n_{p0} = 200$ is considered to be more or less enough only for $\mathrm{Re} \le 10^3$), number of vortex elements on the airfoils surfaces n_p should be increased. Total number of vortices in the flow domain N we assume to be proportional to n^2, time step should be proportionally decreased and number of steps — proportionally increased:

$$n = n_p, \quad N = N_0 \cdot \left(\frac{n_p}{n_{p0}}\right)^2, \quad T = T_0 \cdot \left(\frac{n_p}{n_{p0}}\right). \tag{1}$$

Problem 2. *Circular cylinder hydroelastic oscillations simulation in the presence of the screen.*

The mentioned problem is considered in [16, 17]. In numerical simulation, the flow around the airfoil which models the screen, can be considered as flow without separation, that makes it possible to decrease number of vortices in the flow domain due to vorticity flux simulating only on the cylinder surface (vortex sheet on the screen surface is attached). As basic parameters of the numerical scheme, we consider the following: vortex sheet on the cylinder surface is modeled with $n_{p0} = 200$ vortex elements, vortex sheet on the screen surface — with $n_{e0} = 3n_{p0} = 600$ vortex elements, number of vortices in the wake in flow domain $N_0 = 10\,000$.

Figure 3: Model problem 2 statement

As the result, for arbitrary value of n_p we obtain

$$n = 4n_p, \quad N = N_0 \cdot \left(\frac{n_p}{n_{p0}}\right)^2, \quad T = T_0 \cdot \left(\frac{n_p}{n_{p0}}\right).$$

Computational complexities of the considered model problems for $n_p = 200$ are the following:

$$S_1(200) = \sum_{r=1}^{3} Q_r(n,\, N) + \sum_{r=4}^{7} Q_r(n,\, N) \cdot T_0 = 5.0 \cdot 10^{13},$$

$$S_2(200) = \sum_{r=1}^{7} Q_r(n,\, N) \cdot T_0 = 7.1 \cdot 10^{13}.$$

Estimations for computational complexities of these problems at different values of n_p (being normalized to $S(n_p = 200)$) are shown in Table 1.

Table 1: Computational complexity of the algorithm for different values of n_p in comparison with $S(200)$

n_p	100	200	400	600	800	1 000
$\frac{S_1(n_p)}{S_1(200)}$	0,03	1	32	240	1000	3050
$\frac{S_2(n_p)}{S_2(200)}$	0,05	1	26	188	766	2293

The diagrams in the Fig. 4 and Fig. 5 show how the shares of separate operations vary for the considered problems with different n_p.

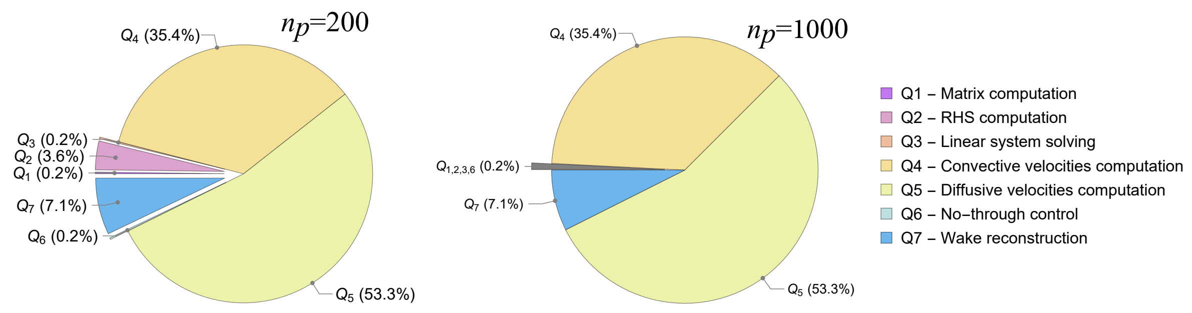

Figure 4: The shares of separate operations for the Problem 1 with $n_p = 200$ and $n_p = 1000$

It is clear from the diagrams, that for different problems with different parameters the distribution of the total computational complexity over the operations can vary significantly. Fig. 4 shows the ratio of the complexities of the operations for Problem 1: it is evident that in the case of $n_p = 1000$ panels, only operations 4, 5 and 7 have significant complexities, while all the other operations takes only about 0.2 %. However, for $n_p = 200$, the operation 3 also becomes significant. In the case of Problem 2, when there is large number of panels on the airfoils, some of which do not shed to the flow, for $n_p = 1000$, complexities of operations 2, 3, 4, 5, 7 are essential, while it seems that complexities of the 1 and 6 operations still can be neglected. However, if we consider the same problem with $n_p = 200$, all the operations are essential, and in order to obtain an efficient acceleration of computations, it is necessary to perform parallelization of all the listed operations.

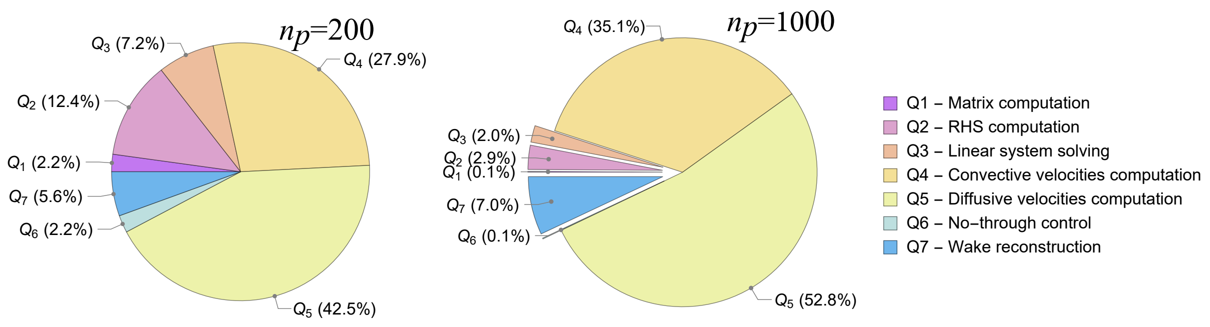

Figure 5: The shares of separate operations for the Problem 2 with $n_p = 200$ and $n_p = 1000$

4 PARALLEL TECHNOLOGIES IMPLEMENTED IN THE VM2D CODE

In the VM2D code parallel algorithms based on the MPI, OpenMP and Nvidia CUDA technologies are implemented, which allow one to perform simulations on multiprocessor computers with different architectures: on CPUs (with distributed and shared memory), and of hybrid systems with CPUs + GPUs. To achieve high acceleration and high efficiency, all the operations listed above are parallelized for all the mentioned parallel technologies.

4.1 MPI + OpenMP parallelization

Parallelization for distributed memory systems is performed using MPI technology, computations within one node are parallelized using OpenMP technology. To estimate the efficiency of parallelization, the test simulations for four problems similar to Problem 1 have been performed with the following parameters: 1) $n_p = 1000$, $N_1 = 80\,000$; 2) $n_p = 2000$, $N_2 = 160\,000$; 3) $n_p = 4000$, $N_3 = 320\,000$; 4) $n_p = 6000$, $N_4 = 480\,000$.

The computations have been performed on two cluster systems:

1. Cluster with HP Blade Servers BL2x220c G7, Ivannikov Institute for System Programming of the RAS (Infiniband QDR, $2 \times$ Intel Xeon X5670 (6 cores), 2.93 GHz).

2. Cluster HPC1 in National Research Center "Kurchatov Institute" (Infiniband QDR, $2\times$ Intel Xeon E5345 (4 cores), 2.33 GHz).

Fig. 6 shows the acceleration of computations on the BL2x220c G7 system. The black line shows the acceleration level for the "ideally parallelized" code with 0.1 % of non-parallel (sequential) code calculated according to the Amdahl law. It can be seen, that parallelization is the more efficient, the more vortex elements are in the vortex wake. Note, that for the considered problems we reach absolute efficiency $0.75 \ldots 0.79$ at 96 cores and $0.62 \ldots 0.75$ at 228 cores. If we normalize our acceleration to the acceleration of the code with 0.1 % of sequential code, we obtain efficiency $0.82 \ldots 0.92$ at 96 cores and $0.75 \ldots 0.88$ at 228 cores. Finally, we reach $140 \ldots 170$ times acceleration of the algorithm at 228 nodes.

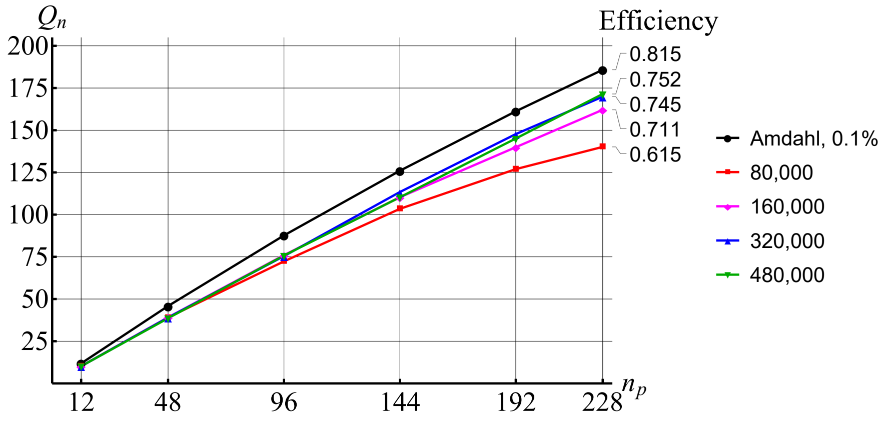

Figure 6: Acceleration of the computations obtained for different number of cores for problems with different parameters on BL2x220c G7

Fig. 7 shows the acceleration of computations on the Claster HPC1 in National Research Center "Kurchatov Institute". As earlier, the black line shows the acceleration of the "ideally parallelized" code with 0.1 % sequential code calculated according to the Amdahl law. For the considered problems we reach absolute efficiency 0.68 . . . 0.76 at 256 cores and 0.30 . . . 0.41 at 1280 cores. If we normalize our acceleration to the acceleration of the code with 0.1 % sequential code, we obtain efficiency 0.84 . . . 0.94 at 256 cores and 0.68 . . . 0.93 at 1280 cores. Finally, we reach 380 . . . 520 times acceleration of the algorithm at 1280 nodes.

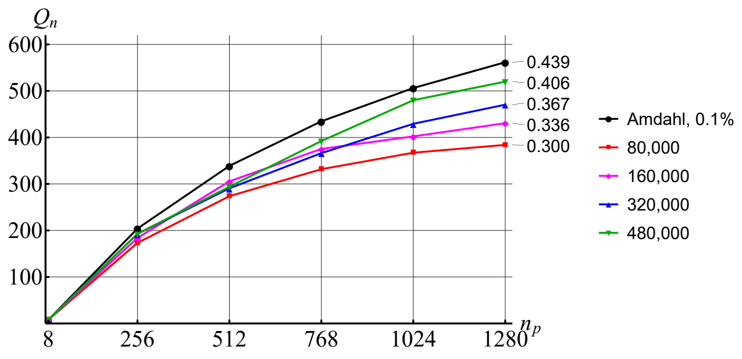

Figure 7: Acceleration of the computations obtained by different number of cores for problems with different parameters on HPC1

4.2 Nvidia CUDA and MPI + CUDA parallelization

All the algorithms for the described operations have been adapted for hybrid computer systems with GPU accelerators. Test simulations were performed by using two GPUs:

	GeForce GTX 970	Tesla K40c
Number of multiproc./cores	13/1664	15/2880
DRAM Memory	4 Gb (3.5 Gb)	12 Gb

Tables 2 and 3 show a comparison of the VM2D code performance on GPUs with its performance on CPUs. It can be seen that in the framework of VM2D one GPU GeForce GTX 970 can replace dozens or even hundreds of CPU cores. GPU Tesla K40c replaces hundreds or even more than 1000 CPU cores.

Table 2: Comparison of performances of GPUs and BL2x220c G7

Acceleration	1	12	48	G970	96	144	192	228	K40
$N = 80\,000$	1	10.2	39.1	**58.7**	72.3	103.4	127.0	140.1	**158.9**
$N = 480\,000$	1	10.2	38.6	**66.1**	75.8	110.3	145.1	171.5	**162.6**

Table 3: Comparison of performances of GPUs and HPC1

Acceleration	1	8	G970	256	512	768	1024	K40	1280
$N = 80\,000$	1	7.2	**127.9**	173.3	273.7	331.7	367.2	**467.6**	468.6
$N = 480\,000$	1	7.3	**190.8**	192.8	294.0	392.3	463.0	**469.0**	565.9

It is also possible to perform calculations on several video cards that are located on different nodes, their communication takes place with the help of MPI. Test calculations show that if you use two video cards, you can get an acceleration of 1.6 times, and for three cards — 2.2 times compared with the calculation on a single video card.

5 CONCLUSIONS

Parallel algorithms for VM2D open-source code were implemented using three technologies: MPI, OpenMP and CUDA. It is shown that ratio between computational complexities of the operations of the algorithm can vary significantly for different problems statements. Test problems shown that VM2D is efficiently parallelized: we achieve $75\ldots79\%$ efficiency of parallelization at 96 CPU cores, $0.68\ldots0.76\%$ efficiency at 256 cores and $0.30\ldots0.41\%$ efficiency at 1280 cores. It is shown that the acceleration achieved on one GPU is comparable to acceleration on hundreds and even thousands of CPU cores. At the same time, the acceleration of calculations on GPU can be further increased due to the fact that it is possible to perform calculations on several cards simultaneously.

ACKNOWLEDGEMENTS

The research is supported by the Ministry of Education and Science of the Russian Federation (project no. 9.2422.2017/PCh), and the Grant of the President of the Russian Federation for Young Ph.D (project no. MK-743.2018.8).

The authors express their gratitude to Ivannikov Institute for System Programming of the Russian Academy of Sciences and to National Research Center “Kurchatov Institute” for providing the computational resources.

REFERENCES

[1] Mittal, R. and Iaccarino, G. Immersed boundary methods. *Annu. Rev. Fluid Mech.* (2005) **37**:239–261.

[2] Marchevskii, I.K. and Puzikova, V.V. Numerical simulation of the flow around two fixed circular airfoils positioned in tandem using the LS-STAG method. *J. Mach. Manuf. Reliab.* (2016) **45**:130–136.

[3] Cottet, G.-H. and Koumoutsakos, P.D. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000.

[4] Saffman, P.G. *Vortex Dynamics*. Cambridge University Press, 1992.

[5] Lifanov, I.K. *Singular Integral Equations and Discrete Vortices*. Utrecht: VSP, 1996.

[6] Lighthill, M. *Boundary Layer Theory*. ed. J. Rosenhead, Oxford: OUP, Introduction, 1963, pp. 54–61.

[7] Dynnikova, G.Ya. Vortex motion in two-dimensional viscous fluid flows. *Fluid Dynamics.* (2003) **38:5**:670–678.

[8] Kuzmina, K.S., Marchevsky, I.K. and Moreva, V.S. Parallel Implementation of Vortex Element Method on CPUs and GPUs. *Procedia Comp. Science.* (2015) **66**:73–82.

[9] Kuzmina, K.S., Marchevsky I.K. and Ryatina E.P. Open Source Code for 2D Incompressible Flow Simulation by Using Meshless Lagrangian Vortex Methods. *IEEE Xplore Digital Library. Ivannikov ISPRAS Open Conference.* (2017) 97–103.

[10] Kuzmina, K.S., Marchevskii, I.K., Moreva, V.S. and Ryatina, E.P. Numerical scheme of the second order of accuracy for vortex methods for incompressible flow simulation around airfoils. *Russian Aeronautics.* (2017) **60:3**:398–405.

[11] Kuzmina, K.S., Marchevskii, I.K. and Moreva, V.S. Vortex Sheet Intensity Computation in Incompressible Flow Simulation Around an Airfoil by Using Vortex Methods. *Mathematical Models and Computer Simulations.* (2018) **10:3**:276–287.

[12] Kuzmina, K.S., Marchevsky, I.K. and Ryatina, E.P., Exact analytical formulae for linearly distributed vortex and source sheets influence computation in 2D vortex methods. *Journal of Physics: Conference Series.*, (2017) **918**:012013.

[13] Kuzmina, K.S., Marchevsky, I.K., Milani, D. and Ryatina, E.P. Accuracy comparison of different approaches for vortex sheet discretization on the airfoil in vortex particles method. *Proc. of 5th Int. Conf. on Particle-Based Methods, Particles 2017 (Hannover, Germany).* (2017) 691–702.

[14] Kempka, S.N., Glass, M.W., Peery, J.S. and Strickland, J.H. *Accuracy considerations for implementing velocity boundary conditions in vorticity formulations.* Sandia Report Sand 96-0583 UC-700, 1996.

[15] Dynnikova, G.Ya. An analog of the Bernoulli and Cauchy – Lagrange integrals for a time-dependent vortex flow of an ideal incompressible fluid. *Fluid Dynamics.* (2000) **35:1**:24–32.

[16] Bearman, P.W. and Zdravkovich, M.M. Flow around a circular cylinder near a plane boundary. *Journal of Fluid Mechanics.* (1978) **89:1**:33–47.

[17] Lei, C., Cheng, L., and Kavanagh, K. Re-examination of the effect of a plane boundary on force and vortex shedding of a circular cylinder. *Journal of Wind Engineering and Industrial Aerodynamics.* (1999) **80:3**:263–286.