ParCFD'2020
32^{nd} International Conference on Parallel Computational Fluid Dynamics
May-11-13 2020, Nice FRANCE

CPU AND GPU PARALLEL ALGORITHMS FOR 2D FLOW SIMULATION USING THE VORTEX PARTICLE METHOD IN THE VM2D CODE

Kseniia S. KUZMINA[*,†] **AND Ilia K. MARCHEVSKY**[*,†]

[*]Bauman Moscow State Technical University
Department of Applied Mathematics
105005, 2-nd Baumanskaya st., 5, Moscow, Russia

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

e-mail: kuz-ksen-serg@yandex.ru

Key words: VM2D code, Viscous vortex domains method, Nvidia CUDA

Abstract. The efficiency of parallel algorithms for 2D flows simulation using vortex particle method is discussed. Original open-source code VM2D is developed and freely available for viscous flows around 2D airfoils simulation and coupled hydroelastic problems solving. The code implements in general the viscous vortex domains method, while some for the boundary integral equation numerical solution some new approaches developed by the authors are used, that allows to improve the accuracy significantly. Three most popular technologies of parallel computing: OpenMP, MPI and Nvidia CUDA are supported in VM2D. The high efficiency of the parallelization is connected with the fact that the viscous vortex domains method is a purely Lagrangian particle method.

1 VORTEX PARTICLE METHOD AND VM2D CODE

In opposite to well-known mesh-based Eulerian CFD methods, in the vortex particle methods (VPM) vorticity is considered as a primary variable. Its distribution in the flow domain (vortex wake) is represented with a set of vortex elements — vortex particles with circulations Γ_i and positions $\boldsymbol{r}_i$ (Figure 1).

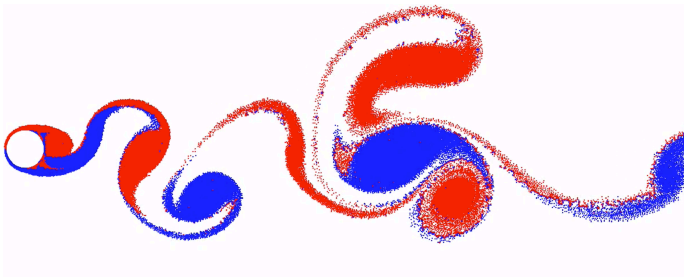

Figure 1: Vortex wake around circular airfoil

Modern modifications of the VPM allows for simulating viscous incompressible flows, described by the Navier – Stokes equation. The viscosity effect can be taken into account in different ways, in particular, in the framework of the viscous vortex domains method (VVD) [2], which is considered in this paper, the so-called diffusive velocity is introduced. It is expressed through the vorticity distribution and its gradient. The resulting velocity field, which is superposition of the flow velocity and diffusive velocity is considered as the transfer velocity for vorticity distribution in the flow domain. Solid walls are the surfaces where vorticity flux is modelled and new vortex particles are introduced in the flow (Figure 1).

The `VM2D` open-source code implements the VVD method, it is freely available on `github` [1]. It is being developed for 3 years by the authors for viscous flows simulation and for fluid-structure interaction modelling, including coupled hydroelastic problems with moving boundaries.

The `VM2D` code is written on C++ and has a modular structure. It is cross-platform and can be compiled under Windows, Linux and MacOS by using MSVC, GCC, Intel C++, Clang compilers. The `Eigen` library is used in `VM2D` for the numerical solution of linear equations systems. The OpenMP, MPI and Nvidia CUDA technologies are used for computations speedup on multi-core and multiprocessor cluster systems, including hybrid systems with graphic accelerators.

2 BRIEF DESCRIPTION OF THE ALGORITHM

A time-step of the VVD algorithm implemented in the `VM2D` code can be divided into 7 main blocks. For all time-consuming operations OpenMP, MPI and CUDA-based algorithms are developed, that can be used separately or together.

1. Vorticity generation on the airfoil surface line. The vorticity generated within a time-step is modelled by a thin vortex sheet at the airfoil surface line K. Its intensity $\gamma(\boldsymbol{r})$ can be found from the no-slip boundary condition, which leads to a boundary integral equation (BIE). In the original VVD method, a singular BIE of the 1st kind is considered [2]; the authors have developed another approach [3, 4] which seems to be more suitable to VVD method due to improved accuracy and leads to the Fredholm-type BIE of the 2nd kind with bounded (or integrable) kernel. The hierarchy of numerical schemes is developed for its discretization, based on Galerkin approach; its discrete analogue is a linear system which should be solved at every time step. The matrix of the linear system remains constant if a rigid airfoil (immovable or moving) is considered, or a system of airfoils, where each of them is immovable relative to the others; in this case the matrix is computed and inverted once at the first time step. The matrix should be computed once again at every time step for flow simulation around deformable airfoils or system of moving airfoils; in this case iterative methods seem to be more efficient for system approximate solving. The right-hand side is computed at every time step.

2. Convective velocities computation is performed for all the vortex particles, and also for the specified points in the flow domain, where velocity should be computed

in the problem being solved. It is reconstructed according to the Biot – Savart law

$$\boldsymbol{V}(\boldsymbol{r}) = \boldsymbol{V}_\infty + \sum_{i=1}^{N} \frac{\Gamma}{2\pi} \frac{\boldsymbol{k} \times (\boldsymbol{r} - \boldsymbol{r}_i)}{|\boldsymbol{r} - \boldsymbol{r}_i|^2} + \oint_K \frac{\gamma(\boldsymbol{\xi})}{2\pi} \frac{\boldsymbol{k} \times (\boldsymbol{r} - \boldsymbol{\xi})}{|\boldsymbol{r} - \boldsymbol{\xi}|^2} dl_\xi,$$

where the summation is performed for all the vortex particles, and integration — over all the airfoil boundaries (for moving airfoils there some additional terms arise, having similar structure). This part of the algorithm is one of the most time-consuming due to $O(N^2)$ computational cost arising from the mutual interaction taking into account.

Note, that specific modifications of this algorithm is developed in order to estimate with high accuracy the velocity of 'new' vortex particles, being formed from the vortex sheet on the airfoils boundaries.

3. Diffusive velocities, according to the VVD method, should be computed for all the vortex particles as

$$\boldsymbol{W}_i = -\nu \frac{\nabla \Omega(\boldsymbol{r})}{\Omega(\boldsymbol{r})}\Big|_{\boldsymbol{r}=\boldsymbol{r}_i}, \quad i = 1, \ldots, N,$$

where special smoothing technique is used for the vorticity field reconstruction and its gradient estimation [2].

As for the previous operation, this step is also time-consuming due to mutual interactions of all the vortex particles, and special algorithms are required to take into account the influence of vortex sheets as well as to estimate the diffusive velocities of the vortex particles, being formed from such sheets.

4. Vortex wake evolution simulation. According to the VVD method, vortex particles move with the velocity $(\boldsymbol{V} + \boldsymbol{W})$:

$$\frac{d\boldsymbol{r}_i}{dt} = \boldsymbol{V}(\boldsymbol{r}_i) + \boldsymbol{W}(\boldsymbol{r}_i), \quad i = 1, \ldots, N. \tag{1}$$

This ODE system is solved with explicit Euler or Runge – Kutta method. This operation becomes complicated because of the need to control the penetration of the vortex particles inside the airfoils. Note, that the airfoils boundaries in current version of the VM2D code are modelled as polygons.

5. Hydrodynamic loads calculation. In order to reconstruct the pressure distribution in the flow domain, an analogue of the Cauchy – Lagrange integral can be used. The values of integral hydrodynamic forces acting the airfoils, can be easily calculated according to the corresponding formulae [2].

6. Coupling scheme for FSI problems can be implemented according to different approaches: from the simplest weak-coupling scheme, to the monolithic one. The last way, despite of being extremely attractive from absolute numerical stability guaranteing, is too complicated and non-robust, so the 'average' iterative scheme, based on splitting hydrodynamic forces into explicit and implicit part, seems to be the most suitable.

7. Vortex wake restructuring is performed to reduce number of vortex particles in the flow by merging closely placed ones and removing the vortices, which are far enough from the airfoil. The most complicated step is the neighbouring particles search.

3 EFFICIENCY OF PARALLEL TECHNOLOGIES IN `VM2D`

Typical number of vortex particles in problems solved by vortex method, varies form several thousand to hundreds of thousand. Typical number of panels on the polygons modelling the airfoils, usually is in range from several hundreds to some thousands.

The implemented OpenMP and MPI technologies provide close to linear speedup up to hundreds of cores, however their efficiency is much smaller in comparison with CUDA technology, especially with modern graphical cards (such as Tesla V100). GPUs are extremely suitable for this algorithm due to the fact that the operations being processed on the particles are the same, rather simple and can be performed independently.

In order to show the parallelization efficiency of `VM2D`, computations were performed for a model problem with 0.5 million of vortex particles and airfoils with 6000 panels (totally) on a cluster system (BL2x220c G7 in the Ivannikov Institute for System Programming of the RAS, Infiniband Network QDR, 2 Intel Xeon X5670 6-core CPU per node, 2.93 GHz) and three graphic accelerators: GeForce GTX 970 (13/1664 multiprocessors/cores), Tesla K40 (15/2880 multiprocessors/cores) and Tesla V100 (84/5386 multiprocessors/cores).

Table 1: Speedup for different number of cores for BL2x220c G7 cluster and GPUs

Cores or GPU	1	12	48	G970	96	144	192	228	K40	V100
Speedup	1	10.2	38.6	**66.1**	75.8	110	145	170	**160**	**900**

It is seen that when using `VM2D`, one GPU GeForce GTX 970 can replace dozens of CPU cores. GPU Tesla V100 replaces up to thousand of CPU cores. The VM2D also makes it possible also to perform computation on multiple graphical cards by using MPI; i.e., two cards provide speedup ratio 1.6, three cards — up to 2.2.

4 CONCLUSIONS

The efficiency of parallel algorithms in VPM implemented in the `VM2D` code is considered. The parallelization efficiency of the whole algorithm for a considered problem is 75 % on 228 cores. Note that one GPU Tesla V100 can replace hundreds of CPU cores.

REFERENCES

[1] VM2D code. URL: https://github.com/vortexmethods/VM2D.

[2] Dynnikova, G.Ya.: The lagrangian approach to solving the time-dependent navier-stokes equations. *Doklady Physics*. (2004) **49:5**:648–652.

[3] Kempka, S.N., Glass, M.W., Peery, J.S., Strickland, J.H. & Ingber, M.S.: Accuracy considerations for implementing velocity boundaryconditions in vorticity formulations. *SANDIA report*. (1996) SAND96-0583, UC-700.

[4] Kuzmina, K.S. & Marchevskii, I.K.: On the Calculation of the Vortex Sheet and Point Vortices Effects at Approximate Solution of the Boundary Integral Equation in 2D Vortex Methods of Computational Hydrodynamics. *Fluid Dynamics*. (2019) **54:7**:991–1001.