XXI Fluid Mechanics Conference IOP Publishing
Journal of Physics: Conference Series **530** (2014) 012021 doi:10.1088/1742-6596/530/1/012021

Numerical study of the vortex tube reconnection using vortex particle method on many graphics cards

Henryk Kudela and Andrzej Kosior

Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland

E-mail: `henryk.kudela@pwr.wroc.pl, andrzej.kosior@pwr.wroc.pl`

Abstract. Vortex Particle Methods are one of the most convenient ways of tracking the vorticity evolution. In the article we presented numerical recreation of the real life experiment concerning head-on collision of two vortex rings. In the experiment the evolution and reconnection of the vortex structures is tracked with passive markers (paint particles) which in viscous fluid does not follow the evolution of vorticity field. In numerical computations we showed the difference between vorticity evolution and movement of passive markers. The agreement with the experiment was very good. Due to problems with very long time of computations on a single processor the Vortex–in–Cell method was implemented on the multicore architecture of the graphics cards (GPUs). Vortex Particle Methods are very well suited for parallel computations. As there are myriads of particles in the flow and for each of them the same equations of motion have to be solved the SIMD architecture used in GPUs seems to be perfect. The main disadvantage in this case is the small amount of the RAM memory. To overcome this problem we created a multiGPU implementation of the VIC method. Some remarks on parallel computing are given in the article.

1. Introduction

Understanding the dynamics and mutual interaction of various types of vortical motions is the key ingredient in clarifying and controlling fluid motions. One of the most fundamental 3D vortical interactions is related to vortex tube reconnection. Reconnection is regarded as a one fundamental mechanism that increases the complexity of the flow what is important in the study of the turbulence phenomena [14].

In most of the real life experiments the vorticity evolution is tracked using some kind of passive markers (like paint particles). But in viscous fluid the particles does not follow the vorticity exactly [4]. In this situations the numerical simulation can be used.

As a numerical method we chose the 3D Vortex–in–Cell (VIC) method [1]. In this method particles carry information about vorticity. It is well known that the velocity may be calculated from the vorticity distribution. Next the vortex particles are displaced according to the local velocity field. Particles intensities are then interpolated back to the grid nodes. To simulate the effect of the viscosity the viscous splitting algorithm was used.

Numerical solution of the 3D Navier – Stokes equations for high Reynolds number, using any method, is a very time consuming process. It may be noticed that recently there is not much increase in the computational power of a single processor. We are forced to use multicore architectures and parallel computation.

The VIC method is very well suited for parallel computation. The displacement and redistribution processes, which have to be done at each time step, have a local character and the computations for each particle can be done independently. So the whole set of particles can be divided into independent groups and operations over these groups can be done concurrently.

Graphics Processing Units (GPU) that were developed for video games provide cheap and easily accessible hardware for scientific calculations. It is build of hundreds of simple streaming processors which altogether give a great computational power. GPUs are relatively cheap and commonly available but in order to take advantage of its potential one needs to use proper numerical algorithms.

In the next chapter the equations of the motion and description of the vortex particle method were be given. In chapter 3 we discussed problems connected to the parallel computations. In chapter 4 we presented a numerical example of a head-on vortex rings collision. The last chapter are conclusions.

2. Equations of the motion and description of the vortex particle method

Equations of incompressible and viscous fluid motion have the following form:

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \nu \Delta \vec{u}, \tag{1}$$

$$\nabla \cdot \vec{u} = 0. \tag{2}$$

where $\vec{u} = (u, v, w)$ is velocity vector, ρ is fluid density, p is pressure, ν is kinematic viscosity. Equation (1) can be transformed to the Helmholtz equation for vorticity evolution [18]:

$$\frac{\partial \vec{\omega}}{\partial t} + (\vec{u} \cdot \nabla)\vec{\omega} = (\vec{\omega} \cdot \nabla)\vec{u} + \nu \Delta \vec{\omega}, \tag{3}$$

where $\vec{\omega} = \nabla \times \vec{u}$.

Generally in all vortex particle methods the viscous splitting algorithm is used [3]. The solution is obtained in two steps: first, the inviscid - Euler equation is solved.

$$\frac{\partial \vec{\omega}}{\partial t} + (\vec{u} \cdot \nabla)\vec{\omega} = (\vec{\omega} \cdot \nabla)\vec{u}. \tag{4}$$

Next, the viscosity effect is simulated by solving the diffusion equation

$$\begin{aligned} \frac{\partial \vec{\omega}}{\partial t} &= \nu \Delta \vec{\omega}, & (5)\\ \vec{\omega}(\vec{x}, 0) &= \vec{\omega}_I, & (6) \end{aligned}$$

where $\vec{\omega}_I$ is vorticity distribution obtained after inviscid step.

For the solution of the equations (5) and (6) one can use any suitable method like the Particle Strength Exchange (PSE) method [1] or the Finite Difference method. In the present paper we solved equations (5) and (6) using the implicit finite difference scheme.

In inviscid flow (4), according to the third Helmholtz theorem [18], vorticity lines move as the material particles of fluids. Thus the movement of the vortex particles can be described by the infinite set of ordinary differential equations:

$$\frac{d\vec{x}_p}{dt} = \vec{u}(\vec{x}_p, t), \qquad \vec{x}(0, \vec{c}) = \vec{c}. \tag{7}$$

where $\vec{c} = (c_1, c_2, c_3)$ means the Lagrange coordinates of fluid particles. Solution of the equation (7) gives the particle-trajectory mapping $\Phi(\cdot, t) : R^3 \to R^3; \vec{c} \to \Phi(\vec{c}, t) = \vec{x} \in R^3$ that is one–to–one and onto. Incompressibility implies that

$$det(\nabla_c \Phi(\vec{c}, t)) = 1. \tag{8}$$

The velocity can be express through vorticity distribution using the Biot–Savart law

$$\vec{u}(\vec{x}, t) = \int \vec{K}(\vec{x} - \vec{x}')\vec{\omega}(\vec{x}', t)d\vec{x}' = (\vec{K} * \vec{\omega})(\vec{x}, t), \tag{9}$$

where $*$ denotes convolution and $\vec{K}$ is a 3×3 matrix kernel

$$\vec{K}(\vec{x}) = \frac{1}{4\pi}\frac{1}{|\vec{x}|^3}\begin{bmatrix} 0 & x_3 & -x_2 \\ -x_3 & 0 & x_1 \\ x_2 & -x_1 & 0 \end{bmatrix}, \qquad |\vec{x}| = \sqrt{x_1^2 + x_2^2 + x_3^2}. \tag{10}$$

Equations (9) are the fundamental formulas for direct vorticity methods [12]. By covering the domain of the flow with a numerical mesh $(N_x \times N_y \times N_z)$ with equidistant spacing h, the i-th component of the intensity vector particle $\vec{\alpha}$ is defined by the expression:

$$\alpha_i = \int_{V_p} \omega_i(x_1, x_2, x_3)\mathrm{d}\vec{x} \approx h^3 \omega_i(\vec{x}_p), \quad \vec{x}_p \in V_p, \quad |V_p| = h^3, \tag{11}$$

where V_p is the volume of the cell with index p. The velocity field is found by summing over all of the particles

$$\frac{d\vec{x}_p}{dt} = \vec{u}(\vec{x}_p, t) \quad = \textstyle\sum_{k=1}^{N_p} \vec{K}(\vec{x}_p - \vec{x}_k)\vec{\alpha}_k(t), \tag{12}$$

$$\frac{d\vec{\alpha}_p}{dt} \quad = (\vec{\alpha}_p(t) \cdot \nabla)\, \vec{u}(\vec{x}_p, t). \tag{13}$$

The disadvantage of the direct vortex particle method given by formula (12) is that the method is very computational time consuming. It stems from the fact that one should take into account all of the mutual interactions between particles that are in the flow. In practical calculation one should also introduce regularization in the formulas (12) removing the singularity of the kernel $\vec{K}$ [16, 13, 12]. To overcome those difficulties we replaced the Biot-Savart law for velocity calculation with the grid method.

From incompressibility (2) stems the existence of the vector potential $\vec{A}$, such that:

$$\vec{u} = \nabla \times \vec{A}. \tag{14}$$

The vector potential $\vec{A}$ defines the velocity field with accuracy to some potential field $\nabla\phi$. Adding $\vec{A} + \nabla\phi$ we do not change the the left side of (14). It is convenient to assume that $\mathrm{div}\vec{A} = 0$. The vorticity is related to $\vec{A}$ as follows

$$\nabla \times \nabla \times \vec{A} = \vec{\omega} = \nabla(\nabla \cdot \vec{A}) - \Delta\vec{A}. \tag{15}$$

So assuming that $\nabla \cdot \vec{A} = 0$ the vector potential can be obtained by solving three Poisson equations

$$\Delta A_i = -\omega_i, \qquad i = 1, 2, 3. \tag{16}$$

The Poisson equation can be solved effectively using the numerical gird and finite difference method. The velocity in grid nodes was obtained also by the finite difference using the formula (14).

Solving equation (12) by Runge-Kutta method we need the velocity of the particles that are out of the grid nodes. This is obtained using the interpolation of the velocity onto that new particle positions. Using grid for solution of equations (12) and (16) significantly accelerates ($\sim$ 1000 times [2]) the calculations. For the solution of the algebraic systems obtained form discretisation of the Poisson equations (16) we chose the Multigrid method that is very well suited for parallelization. Multigrid method used in a sequential architecture can be as fast as the Fast Poisson Solvers (FPS) [17]. The disadvantage of the FPS is that it is a sequential algorithm and can not take the full advantage of the parallel architecture. Our parallel implementation of the Multigrid method gave a speed-up up of 10-12 times comparing to FPS running on a single thread of the processor [7]. The system of equations (12) was solved by the Runge–Kutta method of the 4th order. After solution of the equations (12) obtained intensity of the particles was redistributed onto the grid nodes.

We did the redistribution of the intensities of particles onto grid nodes in each time step, before the solution of the Poisson equations (16).

It was done using an interpolation:

$$\omega_j = \sum_p \tilde{\alpha}_{p_n} \varphi\left(\frac{\vec{x}_j - \tilde{\vec{x}}_p}{h}\right) h^{-3}, \tag{17}$$

where j is the index of the numerical mesh node, p is the index of a particle.

Let us assume that $x \in R$. In this article, we used the following interpolation kernel [1]

$$\varphi(x) = \begin{cases} (2 - 5x^2 + 3|x|^3)/2 & \text{for } 0 \le |x| \le 1, \\ (2 - |x|)^2(1 - |x|)/2 & \text{for } 1 \le |x| \le 2, \\ 0 & \text{for } 2 \le |x|. \end{cases} \tag{18}$$

For the 3D case, $\varphi = \varphi(x)\varphi(y)\varphi(z)$.

If φ satisfies the following moment condition [1]:

$$\sum_j (x - x_j)^l \varphi\left(\frac{x - x_j}{h}\right) = 0, \qquad 1 \le |l| \le m - 1, \tag{19}$$

then

$$f(x) = O\left(h^m\right), \tag{20}$$

and the redistribution process will be of the order m. It means that the polynomial functions up to the order m will be exactly represented by this interpolation.

Kernel (18) used in this work is of order $m = 3$ [1].

3. Remarks on parallel computing on GPUs

As can be seen from the description of the VIC method, most calculations are related to the vortex particles displacement and redistribution. These processes are independent of one another and can therefore be done in parallel. For our calculations we selected Graphics Processing Units (GPUs - also called as a device in [15]). The computer in which GPU is mounted is called as a host. The graphics cards have a great computing power-to-cost ratio, but require quite a different approach to programming.

Details of implementation of the VIC method on a single GPU can be found in the authors' paper [7]. Rewriting the code to use the graphics card in calculations allowed for a speed-up of about 46 times. This result was obtained on a GTX480 GPU compared to a single thread of i7 960 @ 3.2GHz CPU.

Program running on a single GPU has its limitations. The most important is the amount of the RAM memory available on a single device. In our vortex particle method implementation, a single GPU used at that time (GTX480 with 1.5GB of RAM) allowed for a numerical mesh not larger than 128x128x128 nodes. To overcome this limitation we had to write a program capable of using multiple graphics cards. This required distribution and exchange of the data between devices.

Our final implementation of the VIC method was supposed to run on the computer cluster - that means a group of computers (called ,,nodes") used as servers connected to each other through local area network (LAN). Each of the nodes is a stand alone computer and can work on its own. Our target hardware configuration was that each node has more than one GPU (in supercomputing centres one can find nodes with even up to 8 GPUs).

To take advantage of the GPUDirect we introduced the so called hybrid MPI-OpenMP parallel programming (also called multilevel parallel programming) in our code. OpenMP is a standard for parallel programming on systems with shared-memory (in opposite to MPI which is dedicated to systems with distributed memory). The main difference between these two standards is the fact that in MPI each process makes its own copy of the data that it needs, and in OpenMP processes may read or modify memory spaces created by other processes (variables can be set as a shared or private). The main idea in hybrid MPI-OpenMP programming is that OpenMP assures communication between the processes running on the same node and MPI transfers the data between the nodes. OpenMP threads can use the GPUDirect for the communication between GPUs on the same node.

4. Collision of two vortex rings ($Re_\Gamma = 1000$)

This test shows the head–on collision of two vortex rings. The experimental results were published by T.T. Lim and T.B. Nickels in [11] and also on the web side of Lim [10] http://serve.me.nus.edu.sg/limtt/.

Distribution of vorticity in the cross section of the ring was assumed as:

$$\omega(r) = \omega_0 e^{-\frac{r^2}{a^2}} \tag{21}$$

where $\omega_0 = 20$ is the maximal value of vorticity in the tubes cross-section at initial time, $a = 0.5$ is a vortex tube radius. The radius of the ring was $R = 1.0$.

Based on this the total circulation Γ and the Reynolds number were calculated as

$$\Gamma = \int_0^a \omega(r) 2\pi r \; dr, \qquad Re_\Gamma = \frac{\Gamma}{\nu} \tag{22}$$

where $\nu = 0.01$ is kinematic viscosity coefficient. Calculated Reynolds number for this test was $Re_\Gamma = 1000$.

The results in the form of an isosurface plot can be seen in the figure 1 and the visualisation with passive markers is presented in figure 2.

When two rings approach each other the velocities induced by them cause both rings to grow in diameter and become flat. At first one can notice that the vorticity distributions around the axis of symmetry create a thin sheet that was called by Lim as a membrane (frame $t = 2$ in figure 1). As was reported in [11] (see also movies obtained in experiments by Lim) when size of the rings are approximately four times their initial diameter, a symmetrical instability in the form of azimuthal waviness begin to developed (frame $t = 3$ in figure 1, see also [10]). That instability lead to the creation, on the circumference, four bulges of vorticity. Our results are related to the experimental data presented on the web side of Lim for Reynolds number

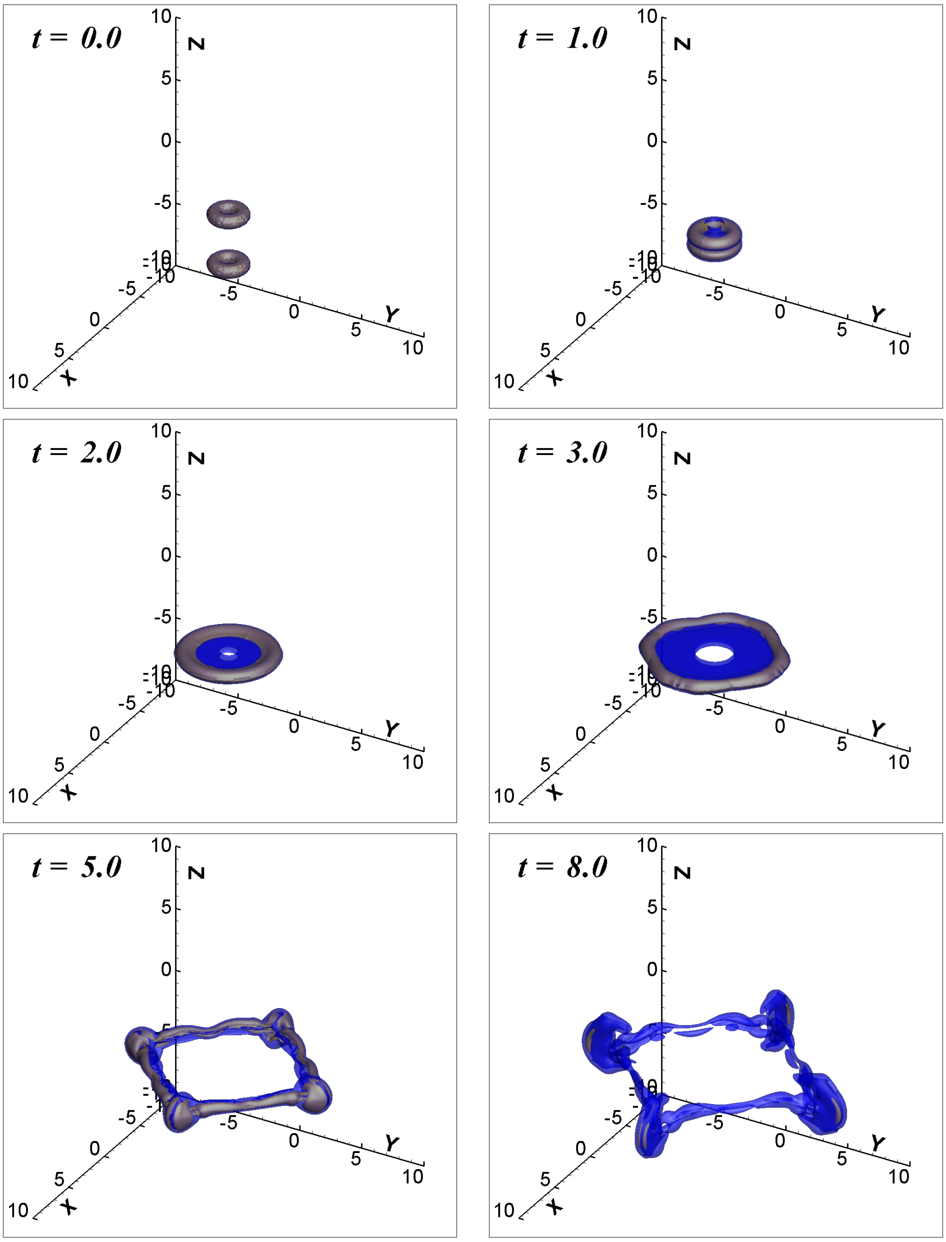

Figure 1. Isosurface plot $|\omega| = 0.05\omega_0$ (blue) and $|\omega| = 0.2\omega_0$ (yellow - look like grey when covered with blue) of collision of two vortex rings. $Re_\Gamma = 1000$.

smaller then 1000. In Lim's experimental results the number of the bulges is five times higher than in our computations. It is hard to compare Lim's results to our computations exactly. We conducted computations on a finite size cube $20 \times 20 \times 20$ with the periodic boundary conditions. Lim related the Reynolds number to the diameter of the hole by which the rings were produced and not to the circulation of the rings as we did. Smaller number of bulges on circumference of vortex structure and rectangular shape of the structure in the end of our computation may results from that. Also the size of our computational domain and periodic boundary conditions impact on our results.

It is known that vorticity is not carried by the fluid particles in the viscous flow and that the

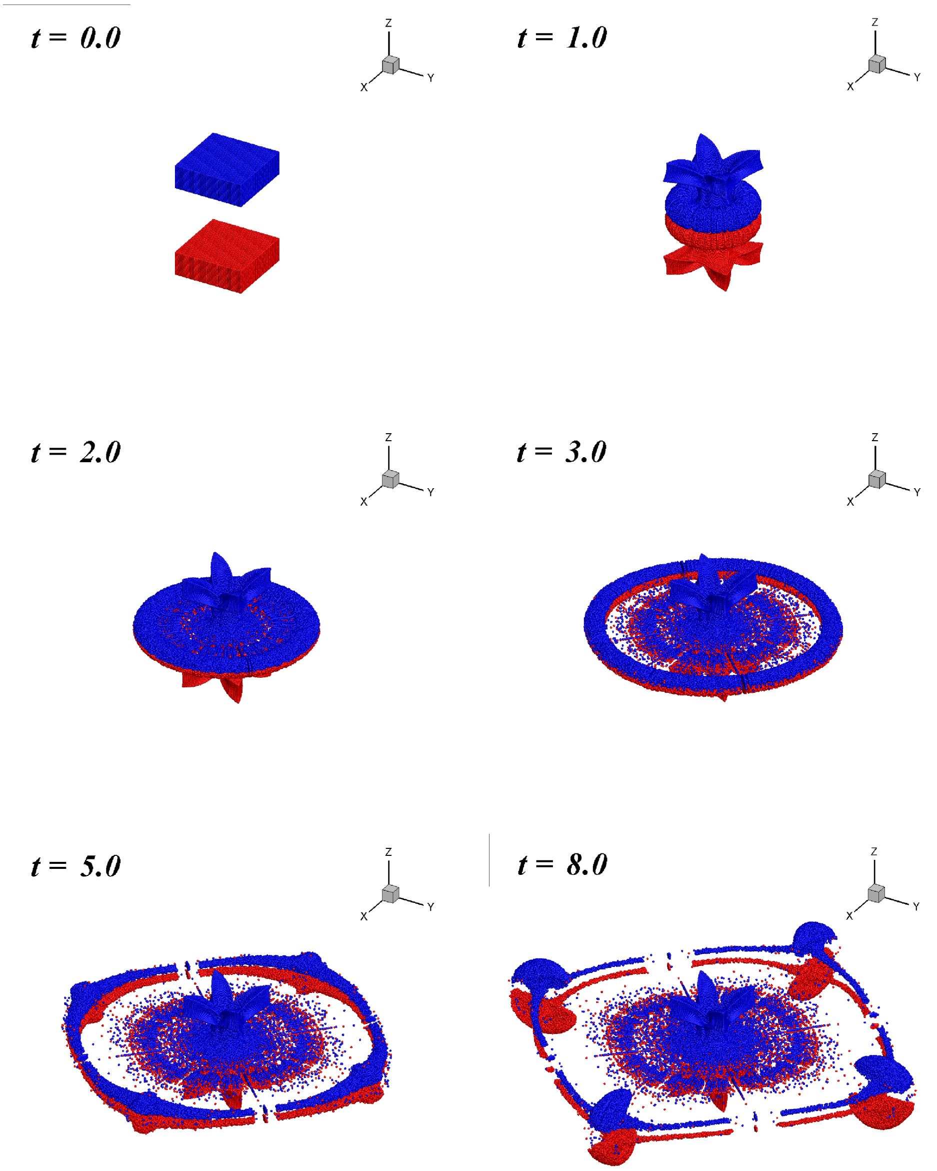

Figure 2. Collision of two vortex rings visualized by the passive marker. On the frame $t = 0$ the vortex rings are merged inside of the cuboids.

vorticity is not the prefect mean for visualization of the flow. Due to this, in order to better compare our results to the real flow visualization we simulated movement of the passive markers carried by the fluid. The vortex rings were merged in boxes of passive markers (colored with the initial position: red - lower ring, blue - upper ring). The size of the cuboids was equal to the diameter of the vortex rings and their thickness. Inside of each box we put a grid of $100 \times 100 \times 50$ passive markers. It is clear that some passive markers laid outside of the rings. Our numerical results were presents in figure 2. One can see that in frame $t = 1$ a tail created by the particle initially placed outside of the rings is formed. In the frame $t = 3$ one can see formation of the membrane. The structure that can be seen in the frame $t = 5$ very much resembles the experimental results by Lim presented in [10]. From the frame $t = 8$ it is clear that reconnection did not take place. The particles did not mix. The reconnection phenomena that would appear in the vorticity bulges could be responsible for the creation of the small rings on a circumference as was reported in [11].

5. Closing remarks

We showed that our implementation of vortex particle method is capable of reproducing result obtained in experiments. Because vorticity is not carried by the fluid particles in the viscous flow numerical simulation seems to give better understanding of the processes behind the reconnection phenomena. In this paper we showed comparison with the real experiment with the use of isosurfaces of vorticity and passive markers.

Nowadays it is not difficult to notice that the computational power of a single processor has stopped rising. Parallel architectures need to be used to deliver the means to speed up computation. Developing programs on GPUs is an interesting alternative to using the CPU. Thanks to hundreds of streaming processors working in parallel we can get the results faster. The processors also quite cheap and easily accessible.

It is obvious that if one wants to have a good resolution of the physical phenomena, one has to use a fine numerical mesh in computations. That requires greater memory and computational time. To overcome this problems, one can use many graphics cards. Introducing of multi GPU computation is our aim in the nearest future. Properly used GPUs (memory management, parallel algorithms, etc.) allows programs to be executed much faster at relatively low cost.

References

[1] G.-H. Cottet, P. D. Koumoutsakos, *Vortex Methods: Theory and Practice*, Cambridge University Press, (2000)
[2] G.-H. Cottet, P. Poncet, *Advances in direct numerical simulations of 3D wall-bounded flows by Vortex-in-Cell methods*, Journal of Computational Physics **169**, 136–158 (2004)
[3] H. Holden, K. H. Karlsen, K.-A. Lie, W. H. Risebro, *Splitting Methods for Partial Differential Equations with Rough Solutions*, European Mathematical Society, (2010)
[4] S. Kida, M. Takaoka, *Breakdown of frozen motion of vorticity field and vorticity reconnection*, Journal of Physical Sociecty of Japan, **60**(7), 2184–2196 (1991)
[5] S. Kida, M. Takaoka, *Vortex Reconnection*, Annual Review of Fluid Mechanics **26**, 169–189 (1994)
[6] S. Kida, M. Takaoka, F. Hussain, *Collision of two vortex rings*, Journal of Fluid Mechanics **230**, 583–646 (1991)
[7] A. Kosior, H. Kudela, *Parallel computations on GPU in 3D using the vortex particle method*, Computers & Fluids **80**, 423–428 (2013)
[8] H. Kudela, P. Regucki, *Vorticity particle method for simulation of 3D flow*, in Computational Science - ICCS 2004 ed by M. Bubak, G. D. van Albada, P. M. A. Sloot and J. J. Dongarra, 356–363 (2004)
[9] T. T. Lim, *A note on the leapfrogging between two coaxial vortex rings at low Reynolds numbers*, Phys. Fluids **9**, 239–241 (1997)
[10] T.T. Lim webpage, `http://serve.me.nus.edu.sg/limtt/`
[11] T. T. Lim, T. B. Nickels, *Instability and Reconnection in the Head-On Collision of Two Vortex Rings*, Nature **357**, 225–227 (1992)
[12] A. J. Majda, A. L. Bertozzi, *Vorticity and Incompressible Flow*, Cambridge University Press, (2002)
[13] E. Meiburg, *Three-dimensional vortex dynamics simulations*, in Fluid Vortices ed by S. I. Green, Kluwer Academic Publishers, 651–685 (1995)
[14] V. M. Melander, F. Hussain, *Cross-linking of two antiparallel vortex tubes*, Phys. Fluids A **1**, 633–636 (1989)
[15] *NVIDIA CUDA C Programming Guide*, `http://www.nvidia.com/cuda`
[16] E. G. Puckett, *Vortex Methods: An Introduction and Survey of Selected Research Topics*, in Incompressible Computational Fluid Dynamics Trends and Advances ed by M. D. Gunzburger, R. A. Nicoladies, Cambridge University Press, 335–407 (1993)
[17] U. Trottenberg, C.W. Oosterlee, A. Schuller, *Multigrid*, Academic Press, (2001)
[18] J. Z. Wu, H. Y. Ma, M. D. Zhou, *Vorticity and Vortex Dynamics*, Springer, (2006)
[19] N. J. Zabusky, M. V. Melander, *Three-dimensional Vortex Tube Reconnection: Morphology For Orthogonally - Offset Tubes*, Physica D **37**, 555–562 (1989)