Vortex-in-cell method for fluid dynamics simulations

Jonas Meier, José Gómez Aleixandre, Anna Hutter,
Carla López Zurita, Daniele Hering

Supervisor 1: Dr. Sriramkrishnan Muralikrishnan (FZ Jülich)
Supervisor 2: Dr. Andreas Adelmann (ETH/PSI)

May 31, 2024

Abstract

In this computational statistical physics project we implemented the vortex in cell (VIC) method, which is a hybrid particle-mesh method used for the simulation of vortex dynamics in fluid dynamic simulations. We started from the incompressible Navier-Stokes equations and we implemented the 2D and the 3D method. The two dimensional method did not include the vortex stretching term as well as the viscous one, whereas the 3D case did include those terms. For the implementation we used the performance portable, C++ library IPPL [1] in 2D and 3D and we validated the method by means of test cases. The results we obtained showed that our 2D version was in agreement with the results in Christiansen's work [2] and our implementation[3] is able to reproduce their results accurately.

Contents

1 Introduction

1.1 Theoretical background

Vortex-in-cell (VIC) is a particle-mesh method used in fluid dynamics simulations. It is often used to model incompressible flows based on the Navier-Stokes equations (NSE) in velocity-vorticity form. The NSE for an incompressible flow read

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\, \vec{u} = -\frac{1}{\rho} \nabla P + \nu \Delta \vec{u}\,,$$
$$\nabla \cdot \vec{u} = 0\,,$$

where $\vec{u}$ is the kinematic velocity vector, ρ the fluid density, P the pressure, and ν the kinematic viscosity. By defining the vorticity field as $\vec{\omega} \coloneqq \nabla \times \vec{u}$, and using the incompressibility condition $\nabla \cdot \vec{u} = 0$, the NSE can be rewritten in the so-called velocity-vorticity formulation:

$$\frac{\partial \vec{\omega}}{\partial t} + (\vec{u} \cdot \nabla)\, \vec{\omega} = (\vec{\omega} \cdot \nabla)\, \vec{u} + \nu \Delta \vec{\omega}\,, \tag{1}$$
$$\nabla \cdot \vec{u} = 0\,. \tag{2}$$

The reason to study the time evolution of the vorticity instead of the velocity is that vortices are the sources of the incompressible flow field. Hence, by knowing the vorticity distribution at any instant of time, both the current state of the system and its future evolution can in principle be determined subject to appropriate boundary conditions [2].

The above formulation of Eq. (1) is an Eulerian description of the vorticity dynamics. It describes the development of the flow field as it is observed at a fixed point of the domain. The Lagrangian description considers the evolution of the vorticity of a moving fluid element. The corresponding formulation can be obtained by substituting the left-hand side of Eq. (1) with the material derivative [4].

The VIC scheme is a hybrid method because it combines both an Eulerian and a Lagrangian approach. The former corresponds with the use of a fixed grid to calculate the evolution of the fields (vorticity, velocity, and stream function), whereas the latter is obtained by tracking the positions and vorticity strengths of moving particles[1]. Hence, the dynamics of a vortex can be obtained by solving the following equations:

$$\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = \vec{u}\,, \tag{3}$$
$$\frac{\mathrm{d}\vec{\omega}}{\mathrm{d}t} = (\vec{\omega} \cdot \nabla)\, \vec{u} + \nu \Delta \vec{\omega}\,. \tag{4}$$

As mentioned before, the only required initial conditions for a full description of the dynamics are starting values for the positions and vorticity strengths of the particles. To compute the velocities needed for Eqs. (3) and (4), we note that a further consequence of Eq. (2) is the existence of a vector potential $\vec{A}$ (or stream function), which satisfies the relation

$$\vec{u} = \nabla \times \vec{A}\,. \tag{5}$$

[1] These "particles" represent point vortices, i.e., vortex elements or constituents of the full vortex.

Without loss of generality, we can assume that $\vec{A}$ is also an incompressible field, i.e. , it satisfies $\nabla \cdot \vec{A}$. Then, its i-th component is related to the i-th component of the vorticity field via the Poisson equation

$$\Delta A_i = -\omega_i \,. \tag{6}$$

1.2 Project description

The purpose of this project is to implement the VIC method within the framework provided by the performance portable, C++ library IPPL [1]. The implementation shall include both 2D and 3D versions of the inviscid scheme as well as the possibility to take viscosity effects into account. The numerical scheme will feature a hybrid, particle-mesh approach, such that Eqs. (5) and (6) are solved on the grid, while Eqs. (3) and (4) describe the moving particles. The implementation will be tested with a series of cases including several vortex geometries and vorticity distributions.

2 Numerical methods

2.1 2D inviscid scheme

If we consider an inviscid flow $(\nu = 0)$, the second term in the right-hand side of Eq. (4) becomes zero. Furthermore, in a 2D scheme, i.e. , with $\vec{u} = (u_x, u_y, 0)$, the vorticity field has only a non-zero component in the z-direction, since $\vec{\omega}$ and $\vec{u}$ are related by

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

Hence, the first term of the right-hand side of that equation also becomes zero:

$$(\vec{\omega} \cdot \nabla)\, \vec{u} = (\omega_x \partial_x + \omega_y \partial_y + \omega_z \partial_z)\, \vec{u} = \omega_z \partial_z \vec{u} = 0 \,.$$

Therefore, the 2D inviscid scheme reduces to solving the following equations:

$$\begin{aligned} \frac{\mathrm{d}\vec{x}}{\mathrm{d}t} &= \vec{u} \,, \\ \frac{\mathrm{d}\vec{\omega}}{\mathrm{d}t} &= 0 \,. \end{aligned}$$

The computational algorithm for simulating the 2D inviscid scheme with N particles can be described as follows:

i) Initialize particle positions and vortex strengths: $\left\{\left(\vec{x}_\alpha^{(0)}, \vec{\omega}_\alpha^{(0)}\right)\right\}_{\alpha=1,\dots,N}$.

ii) Interpolate the vortex strengths to the grid: $\vec{\omega}_\alpha \mapsto \vec{\omega}_p$.

iii) Solve Eq. (6) to obtain the stream function on the grid: $\omega_{i,p} \mapsto A_{i,p}$.

iv) Obtain the velocity values on the grid using Eq. (5): $A_{i,p} \mapsto u_{i,p}$.

v) Interpolate back the velocity values to the vortices: $\vec{u}_p \mapsto \vec{u}_\alpha$.

vi) Push the particles forward in time with Eq. (3).

The initialization of the particles can be done either deterministically or randomly. The simplest way to deterministically initialize particles is to place one particle either on each of the grid nodes or the center of each grid cell [4]. An alternative to that is to place the particles equally spaced along a defined geometry [2]. The possibility of quasi-random methods (e.g. , Monte-Carlo-based algorithms) will not be further explored here, although other authors have used it [4].

To interpolate the particles between their positions and the grid, a cloud-in-cell (CIC) kernel is used. Thus, the vorticity of the particle is assigned to the four nearest grid points weighted with the volume defined by the particle and the corresponding grid point. If $(\mathrm{d}x, \mathrm{d}y)$ is the relative position of the particle α to the grid point $p = (j, k)$, then we can write

$$\begin{aligned}
\omega(j,k) &= \frac{1}{h_x h_y}(1-\mathrm{d}x)(1-\mathrm{d}y) =: C_1\,,\\
\omega(j+1,k) &= \frac{1}{h_x h_y}\,\mathrm{d}x(1-\mathrm{d}y) =: C_2\,,\\
\omega(j,k+1) &= \frac{1}{h_x h_y}(1-\mathrm{d}x)\,\mathrm{d}y =: C_3\,,\\
\omega(j+1,k+1) &= \frac{1}{h_x h_y}\,\mathrm{d}x\,\mathrm{d}y =: C_4\,,
\end{aligned}$$

where h_x and h_y are the grid spacings in the corresponding directions. The same coefficients are used during the back-interpolation step to transfer the velocities from the grid to the particles.

The solution of the Poisson equation relating the stream function to the vorticities can be found with a variety of methods. Earlier approaches included the Hockney method, a finite-difference (FD) scheme based on a five-point stencil and matrix diagonalization [2]. A more efficient approach, which will be employed here, is the use of a fast Poisson solver (FFTPS), which combines the properties of the Fourier transform with the effectiveness of the FFT algorithm. This technique is particularly suitable in combination with periodic boundary conditions.

The solution of Eq. (5) is obtained by means of a central difference FD scheme, as given by the following equations:

$$\begin{aligned}
u_x(j,k) &= \frac{A(j,k+1)-A(j,k-1)}{2h_y}\,,\\
u_y(j,k) &= -\frac{A(j+1,k)-A(j-1,k)}{2h_x}\,.
\end{aligned}$$

Note that here no component of $\vec{A}$ was specified. As argued above, in two dimensions, $\vec{\omega}$ has only one non-zero component, and hence so does $\vec{A}$. Therefore, they can be treated as scalar quantities.

The time evolution of the particles is calculated using a full-time-step leap-frog scheme, as described by Christiansen's work [2]. This is achieved by calculating the new positions according to the following formula:

$$\vec{x}_\alpha^{\,(n+1)} = \vec{x}_\alpha^{\,(n-1)} + \vec{u}_\alpha^{\,(n)}\left(\vec{x}_\alpha^{\,(n)}\right)\cdot 2\Delta t\,.$$

Therefore, the position of the particle α at the time step $n+1$ is calculated using its position at the time step $n-1$ and the velocity obtained during the n-th iteration of the algorithm described above. The main benefit of employing the leap-frog scheme is that it is symplectic, and hence ensures energy conservation. In the above formula, $\vec{u}_\alpha^{(n)}$ is obtained again with CIC kernel, i.e. , by using the kernel coefficients obtained during the n-th time step:

$$\vec{u}_\alpha^{(n)}\left(\vec{x}_\alpha^{(n)}\right) = C_1^{(n)}\vec{u}_p^{(n)}(j,k) + C_2^{(n)}\vec{u}_p^{(n)}(j+1,k) + C_3^{(n)}\vec{u}_p^{(n)}(j,k+1) + C_4^{(n)}\vec{u}_p^{(n)}(j+1,k+1)\,.$$

2.2 3D inviscid scheme

In the 3D inviscid scheme, as seen above, the first term on the right-hand side of Eq. (4) does not vanish. Hence, by extending all the relevant quantities to three-dimensional vectors and calculating the time evolution of the vorticities, the full 3D scheme can be achieved. The corresponding computational algorithm is very similar to the 2D inviscid case. Now, after having calculated the velocities on the grid from the stream function, the term $\vec{R}_1 \coloneqq (\vec{\omega}\cdot\nabla)\,\vec{u}$ needs to be also calculated on the grid. This can be done as well by applying a central difference FD scheme. The following equation shows how to determine the i-th component of $\vec{R}_1$ at time step n and grid point (j,k,l) :

$$\begin{aligned} R_{1,i}^{(n)}(j,k,l) = {} & \omega_x^{(n)}(j,k,l)\frac{u_i^{(n)}(j+1,k,l) - u_i^{(n)}(j-1,k,l)}{2h_x} \\ & + \omega_y^{(n)}(j,k,l)\frac{u_i^{(n)}(j,k+1,l) - u_i^{(n)}(j,k-1,l)}{2h_y} \\ & + \omega_z^{(n)}(j,k,l)\frac{u_i^{(n)}(j,k,l+1) - u_i^{(n)}(j,k,l-1)}{2h_z}\,. \end{aligned}$$

After this calculation, both the velocities and $\vec{R}_1$ need to be interpolated to the particles, as explained above. Finally, the particles are pushed in the same way that was described for the 2D inviscid scheme, and the time evolution of the vorticities can be simply obtained through an explicit Euler step:

$$\vec{\omega}_\alpha^{(n+1)} = \vec{\omega}_\alpha^{(n)} + \Delta t\cdot\vec{R}_1^{(n)}\,.$$

3 Test cases

3.1 Description of the test cases

The test cases used to evaluate the performance of our implementation of the VIC method can be classified into different geometries. For each geometry[2], both uniform and Gaussian vorticity profiles were used. In all cases, a Cartesian grid with equal grid spacing in all directions was employed[3].

The general procedure to generate the initial conditions can be summarized as follows: given a grid size L_i , and a number of grid points G_i for the i-th direction, choose a number of

[2]The only exception to this is the jet stream, where only uniform vorticities were used.

[3]Note that the implementation allows for different grid sizes, grid spacings as well as number of grid points for each direction.

particles N given by

$$N = \rho \prod_i G_i \,.$$

Thereby, ρ is an integer parameter indicating the density of particles, i.e., the number of particles per grid point. Then set the grid spacing to $h_i := L_i/G_i$, and the time step to

$$\Delta t := \min\{0.05, 0.5 \cdot h\} \,.$$

This choice of time step allows us to fulfill the Courant-Friedrichs-Lewy (CFL) condition. Then, in 2D, we choose among the following geometries:

- Disk: $X = \left\{(x,y) \,|\, x^2 + y^2 \le r^2\right\}$,
- Band: $X = \{(x,y) \,|\, a \le y \le b\}$.
- Two bands: $X = X_1 \times X_2$ with $X_k = \{(x,y) \,|\, y \in [a_k, b_k]\}$.
- Half-plane: $X = \{(x,y) \,|\, y \le a\}$.
- Concentric rings: $X = \prod_{m=1}^{M} X_m$, with $X_m = \left\{(x,y) \,|\, r_{m-1}^2 \le x^2 + y^2 \le r_m^2\right\}$.
- Jet streams: $X = X_1 \times X_2$ with $X_k = \{(x,y) \,|\, (x,y) \in [a_k, b_k] \times [c_k, d_k]\}$, and $b_k - a_k \ll d_k - c_k$ for all k.

Next, the N particles are generated either deterministically or randomly. Deterministic generation consists of placing one particle per $1/\rho$ grid points, whereas randomly generated positions are sampled from a two-dimensional uniform random distribution. Finally, those particles that lie within the selected geometry, as given by the set X, get assigned a non-zero vorticity value. In the case of uniform vorticity, we choose

$$\omega_\alpha = \omega_0$$

for all $\alpha \in \{1, \ldots, N\}$. In the case of a Gaussian vorticity distribution, the vorticity strengths are assigned according to a normal distribution centered on the center of the region defined by the geometry and with amplitude ω_0.

3.2 Test results

In this section, we present some of the results obtained from testing our implementation of the 2D inviscid scheme. Further results, both from the 2D and 3D inviscid schemes, will be added in a later version of this work, either here or as an appendix.

Figures 1 to 5 show the time evolution of the flow for some of the defined geometries. These results demonstrate qualitatively that our implementation of the 2D inviscid scheme can successfully replicate the results presented in [2] (cf. Fig. 5). In particular, Figs. 1-5 simulate a Rankine vortex, a Diochotron-type instability with two circles, the formation of a von Karman vortex street, and the penetration of a jet, respectively.

We observe that the disk-type vortex expands and suffers certain deformation when using a uniform vorticity distribution (Fig. 1). When the vorticity follows a Gaussian distribution (Fig. 2), there is a slight increase in size, but no change in shape can be appreciated. Although

the vortex volume should be an invariant of motion for inviscid regimes [2, 4], a similar behavior was shown in [5].

A singular ring produces a Diochotron-type instability [2], a phenomenon that is also well-known in plasma physics. When choosing the geometry to be two concentric rings, we observe that our obtained pattern is the complement of the one that follows from this type is instability (Fig. 3).

Our implementation also allows for the simulation of geometries without circular symmetry. An example is the two-band geometry shown in Fig. 4. Several tests on rectangular geometries (one band, two bands, and half-plane) have shown that a deterministic sampling of equally spaced particles is too stable. No change in the structure could be observed, even for longer simulation times. We believe that this type of initial conditions might describe well a laminar flow, where no instabilities appear, at least for finite simulation times. The result shown here is a two-band geometry, where a small amount of noise has been added to the initial values of the vorticity for some randomly chosen particles. This little amount of initial instability proves enough to replicate the expected behavior of the Karman vortex street. Similar results were also obtained with uniform vorticity distribution and random position sampling.

The final test (Fig. 5) shows the penetration of two jets perpendicular to a shallow band with vorticity $\omega_{\text{band}} \sim 0$. Again here our results replicate very accurately the test case in Christiansen's work [2].

Further quantitative tests are yet to be performed. For inviscid regimes, several quantities such as the total kinetic energy, linear momentum, and angular momentum are theoretically invariants of the time evolution [4]. The last two can be easily calculated for a discretized, numerical scheme, whereas a precise calculation of the total kinetic energy is not straightforward. Determining it naively as the sum of the velocities squared shows, in most cases, a significant loss in kinetic energy. This behavior is to be expected, according to [4], and it is mostly due to losses during the interpolation steps between the particles and the grid. An exception to this behavior is shown by the jet streams. In that case, the energy initially increases. Although more tests need to be performed to confirm this behavior, we believe it to be a consequence of the geometry, which induces a transformation of potential in kinetic energy.

We would like also to note that all these tests have been performed such that the particles that initially had a zero-valued vorticity, have been removed before running the simulation. A few, not yet conclusive tests have shown that leaving or removing those particles might change the behavior of the system. This aspect still needs to be studied in more detail.

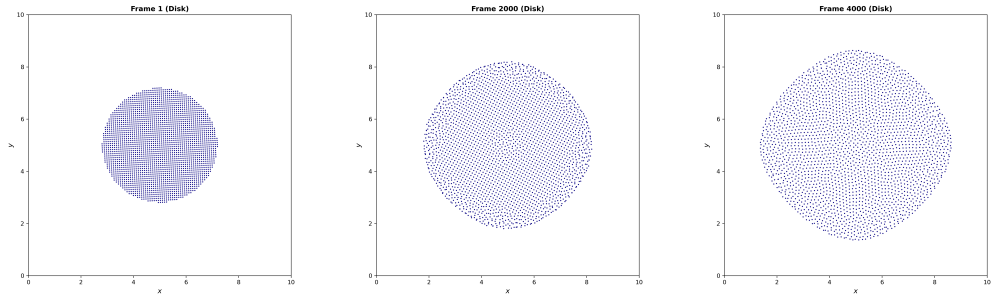

Figure 1: Time evolution of a Rankine-type vortex with uniform vorticity distribution.

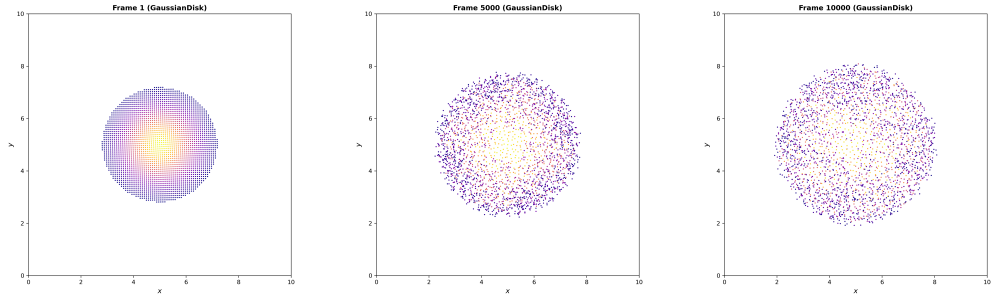

Figure 2: Time evolution of a Rankine-type vortex with Gaussian vorticity distribution.

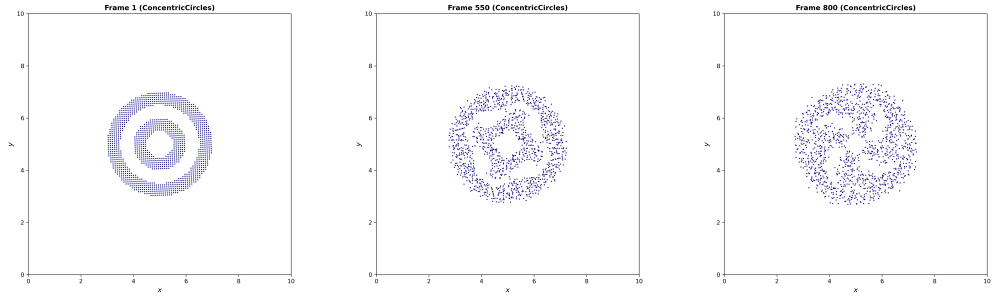

Figure 3: Time evolution of two concentric ring vortices.

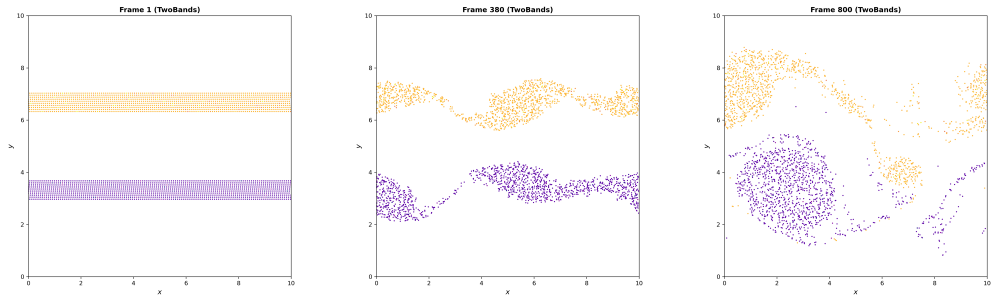

Figure 4: Time evolution of two band jets with opposing vorticity values.

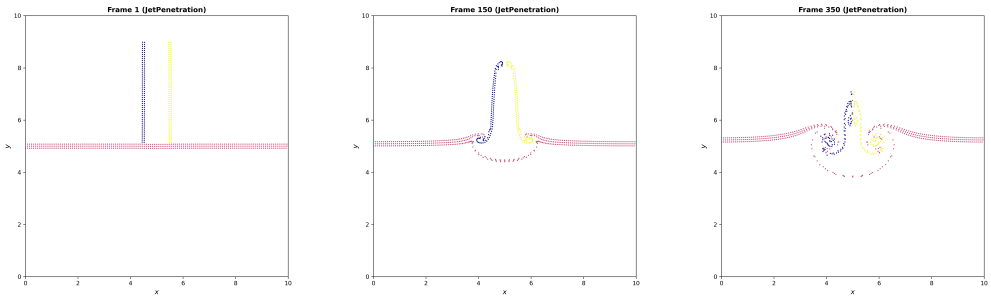

Figure 5: Time evolution of two parallel jet streams penetrating a shallow fluid band.

References

[1] S. Muralikrishnan, M. Frey, A. Vinciguerra, M. Ligotino, A. J. Cerfon, M. Stoyanov, R. Gayatri, and A. Adelmann, "Scaling and performance portability of the particle-in-cell scheme for plasma physics applications through mini-apps targeting exascale architectures", in Proceedings of the 2024 siam conference on parallel processing for scientific computing (pp) (SIAM, 2024), pp. 26–38.

[2] I. Christiansen, "Numerical simulation of hydrodynamics by the method of point vortices", Journal of Computational Physics **13**, 363 (1973).

[3] A. H. C. L. Z. D. H. J. Meier J. Gomez Aleixandre, *Vortex-in-cell method*, `https://github.com/JoMee/ippl_fork`, 2024.

[4] G.-H. Cottet, P. D. Koumoutsakos, et al., *Vortex methods: theory and practice*, Vol. 313 (Cambridge University Press, Cambridge, 2000).

[5] H. Kudela and A. Kosior, "Modeling vortex rings dynamics with vortex in cell method", in Journal of physics: conference series, Vol. 318, 6 (IOP Publishing, 2011), p. 062014.