Two Dimensional Vortex-in-Cell Method

David Akinyemi

Bates College

Lawrence Berkeley National Laboratory Visiting Faculty Program Summer 2019

This paper documents the process my team and I went through to gain an understanding of fluid dynamics using a two-dimensional vortex particle in cell method. Utilizing ordinary and partial differential equations, we took an in-depth look at the interaction between particles based off their vorticity strength in two dimensions. This report examines three tests to determine the effectiveness of our code and concludes with an analysis of the results of each of the tests and then details why specific tests worked better than others.

Mathematics and its application to how people interact with the world has always been a topic of excitement and discovery. Scientists and engineers frequently use mathematics, physics and computer science to examine and gain a better understanding of how the world around them works. An area that we have been able to further study is that of fluid dynamics; learning the mechanics of liquids and gases and more specifically how they flow. This field has many applications with a popular one being aerodynamics and the study of how air flows around airplanes and automobiles. This paper will detail the process the team took to learn about and implement vortex-in-cell methods, a Lagrangian particle method similar to particle-in-cell that can be used to understand how fluids operate. This research is a subset of the work being completed by UC Berkeley graduate student Colin Wahl, focused on developing efficient high-order particle-in-cell methods for plasma physics and astrophysics.

This paper explains the process taken to examine the interaction between vortices and the forces acting between them in two-dimensional space. This elementary version of Wahl's research gave us more knowledge about this specific combination of physics and mathematics and a foundational understanding of the work that he is doing. The paper will discuss the assigned tasks designed to teach us about numerical methods for fluid dynamics based on Lagrangian particle methods. Our work was able to provide a test bed for the adaptive remapping ideas that are currently under development in the Lawrence Berkeley National Laboratory Applied Numerical Algorithms Group. We were ultimately able to implement a two-dimensional version of the vortex in cell method in preparation for the implementation of a global-in-space remapping step.

Materials and Methods

This work was completed in Python and began with creating code for particle interpolation and vorticity deposition. In a vortex method, we represent a fluid as a collection of rotating particles of fluid, known as vortices. Each particle had an x and y coordinate for it's location in space and the strength of its vorticity which is the amount that it is spinning. Next we lay down a rectangular grid (or mesh) in space to divide our domain into distinct cells. The size of each grid cell was Δx by Δy. Interpolating distributes each particle to its nearest grid points and divides the vorticity strength to be deposited depending on how close the original particle is to the grid points. Equation 1 below is the one used for particle deposition where p indexes the particles and i indexes the grid points. Equation 2 is the function W_2 required for deposition that distributes vorticity strength based on distance. Using this interpolation and deposition method, we were able to distribute particles to their nearest grid points with their respective strengths and combine those points that ended up with the same interpolated grid point coordinates.

$$\Omega_i = \sum_p \omega_p W_2\left(\frac{x_i - x_p}{\Delta x}\right) W_2\left(\frac{y_i - y_p}{\Delta y}\right) \tag{1}$$

$$W_2(y) = \begin{cases} 1 - |y|, & 0 \leq |y| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Once we had vorticity values at their proper grid points, we used Hockney's algorithm to find the potential at all the grid points. The implementation of this algorithm was provided by Wahl as a black box known as the Poissons solver to help us compute Equation 3. This input to the Poisson solver is the multi-dimensional array containing all the vorticity values at their interpolated grid points. The solver returns an array containing the potentials, ψ, at all the grid points. We next implemented a finite difference method to calculate, U the fields at each of our initial particle points. We implemented the centered difference[3] to compute the fields, shown in Equation 4.

$$\Delta\psi = \Omega \tag{3}$$

$$U = \left(\frac{\partial\psi}{\partial y}, -\frac{\partial\psi}{\partial x}\right) \tag{4}$$

We then interpolated the field back to the particles using the same interpolation function in Equation 2. This equation however, allowed us to calculate an initial x and y velocity for each particle as shown in Equation 5.

$$U_k = \sum_i U_i W_2\left(\frac{x_k - x_i}{\Delta x}\right) W_2\left(\frac{y_k - y_i}{\Delta y}\right) \tag{5}$$

Once we were able to gather these velocities, the next step was to evolve these particles in time. In order to do this, we decided to use a second-order Runge-Kutta method shown in Equation 6. $F(t, x)$ represents the entire process that allowed us to calculate our initial velocities for each particle. Equation 6 details the Runge-Kutta method where we used the current position of particles to compute its velocity, similar to the trapezoid rule for integration. We then moved to new positions by applying the current velocity for half of the time step. We would calculate an intermediate velocity at this point, move the particle again, and apply the velocity over the full-time step. Repeating this process enabled us to see how each particle moved with time based on the initial velocity calculated from the PIC method.

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \mathbf{U}(t, \mathbf{x}) \\ \mathbf{k}_1 &= \Delta t \mathbf{U}(t^n, \mathbf{x}^n) \\ \mathbf{k}_2 &= \Delta t \mathbf{U}(t^n + \frac{1}{2}\Delta t, \mathbf{x}^n + \frac{1}{2}\mathbf{k}_1) \\ \mathbf{x}^{n+1} &= \mathbf{x}^n + \mathbf{k}_2 \end{aligned} \tag{6}$$

Results

After creating our code for the vortex particle in cell method we were given three test cases to use to check its functionality. The test conditions that we chose are shown in Table 1. Test one included one particle with a vorticity strength of $\frac{1}{\Delta x \Delta y}$ placed near the center of the grid. Any self-forcing is nonphysical so over time we should not expect to see this particle move. Test two contained two particles, one particle at $(0.25, 0.5)$ with a vorticity strength of 0 and another at $(0.75, 0.5)$ with a strength of $\frac{1}{\Delta x \Delta y}$. The zero-strength particle should rotate around the other particle while the other particle in the center stays fixed as shown in Figure 1A. Test three involved two particles and allowed us to see if they interact correctly. One particle is located at $(0.25, 0.5)$ and another at $(0.75, 0.5)$ both with a vorticity strength of $\frac{1}{\Delta x \Delta y}$. The expectation is that both particles should rotate around the center $(0.5, 0.5)$ as shown in Figure 1B.

N, number of iterations	x grid length	y grid length	Δx	Δy	Δt
2^5	1	1	$\frac{\text{x grid length}}{N}$	$\frac{\text{y grid length}}{N}$	$\frac{3.5}{N}$

Table 1: Testing conditions

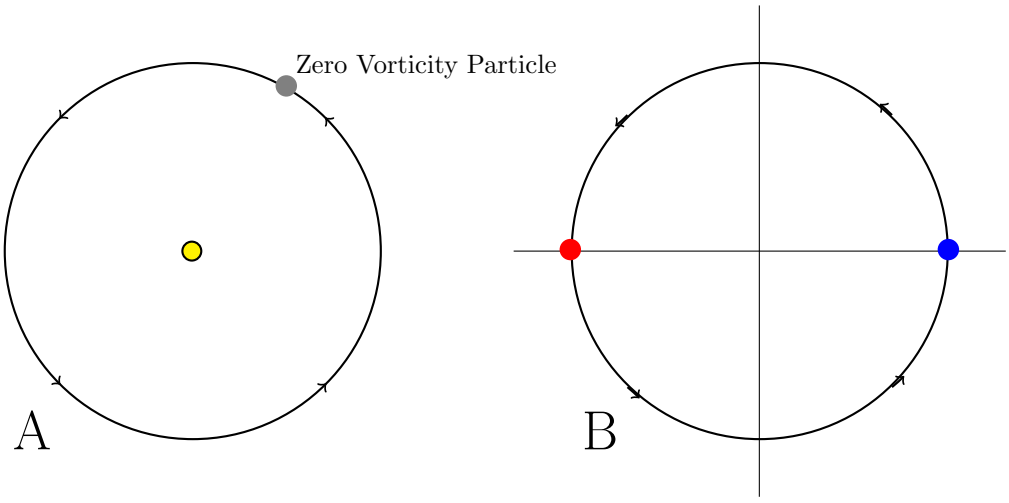

Figure 1: Test Cases 2 and 3: A, shows the way the zero strength particle should rotate around the non-zero strength particle in test case 2. B, shows the way both non-zero strength particles should rotate around each other relative to their center.

In an attempt to apply our code to these test cases, we were successful in our first and second attempts. While running test one, we were able to observe our particle position remain stable over time. In test two we watched as one of our particles orbited around the other. Figure 2 shows test two with conditions that worked to create the proper orbit. However, we found that there was a delicate relationship between how much time we moved our particles, the number of iterations that our Runge-Kutta used, N, and our grid spacing, Δx, Δy. Initially we thought this relationship created the issue with test three, causing our particles to not orbit in uniform circles.

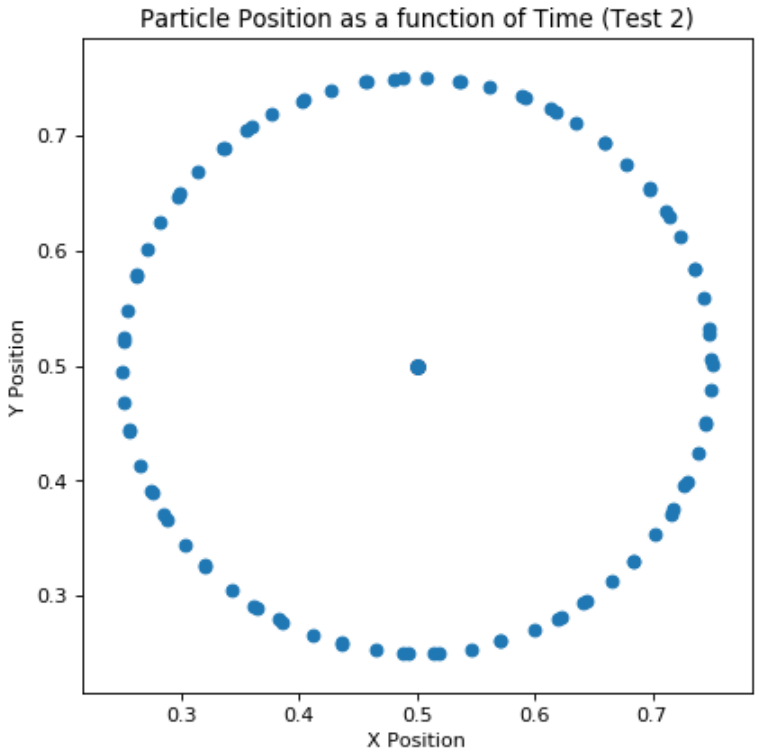

Figure 2: Test 2: Running the code for 10 second and a particle with a vorticity strength of $\frac{1}{\Delta x \Delta y}$ we were able to produce a correct orbit.

After taking more time to look into test three we found more information concerning time step limits for Lagrangian Particle Methods. The fact that the time step and grid sizes are bounded by properties of the distribution is what makes Lagrangian particle methods more appealing when compared to grid-based solvers such as finite difference, finite element and finite volume. With these solvers time steps are typically related to the grid size usually between Δt and Δx. For vortex methods however, the time step is usually proportional to the fundamental length scale for the problem.

Taking this into account and implementing a new Poisson solver we were able to run test three to obtain satisfactory results.

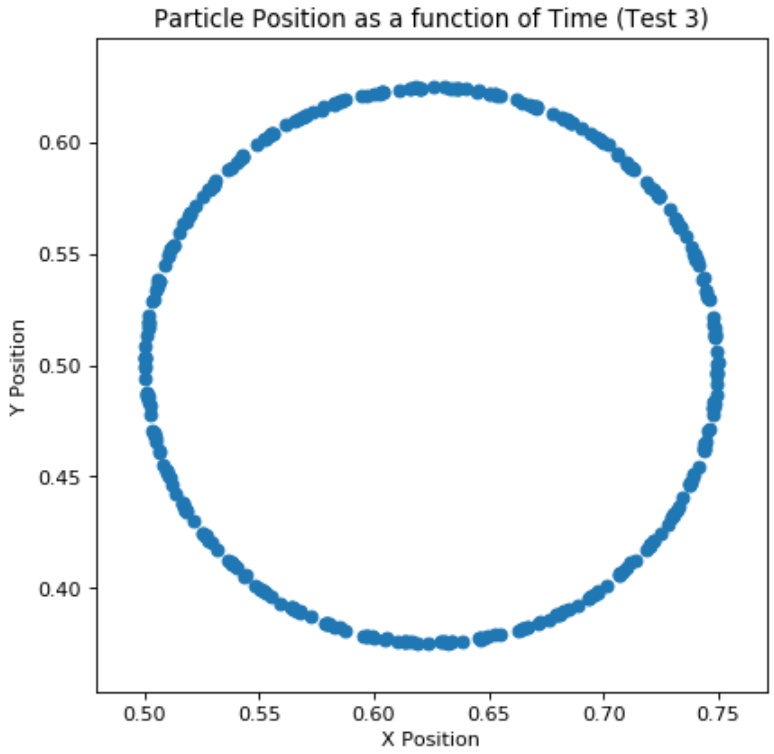

Figure 3: Test 3: Running the code for 10 seconds with both particles having a vorticity strength of $\frac{1}{\Delta x \Delta y}$ we were able to produce a proper orbit around their center.

Conclusion

From my time at LBNL, I have gained an understanding of fluid dynamics and how particles interact based on the vortices acting on them. As a result, I have been able to create a solver for ordinary and partial differential equations and implement a particle-in-cell method. We deemed our results satisfactory for our 3 tests and were able to gain insight into the relationship between time and grid spacing limits when it comes to Lagrangian Particle Methods. This experience has improved my scientific computing skills and expanded my knowledge in fluid dynamics and particle-in-cell methods; continued work and improvement in these skills will take me far in my desired field of engineering in the future.

References

[1] Colella, Phillip, " CS 294-73 Software Engineering for Scientific Computing." Berkeley, California. Lecture 13: Particle Methods; Homework 3.

[2] Heinzl R., Schwaha P., Stimpfl F., Selberherr S. (2008) Concepts for High-Perfomance Scientific Computing. In: Filipe J., Shishkov B., Helfert M., Maciaszek L.A. (eds) Software and Data Technologies. ICSOFT 2007, ENASE 2007. Communications in Computer and Information Science, vol 22. Springer, Berlin, Heidelberg

[3] Olver, Peter J. "Introduction to Partial Differential Equations: 9783319020983."

Acknowledgements

I would like to thank Professor Boateng for inviting me back to continue research with him, if it were not for him I would not be here and be able to experience this wonderful opportunity.

I would like to thank my research partner Hannah Beams and graduate student Colin Wahl for working with me throughout every part of the process and the patience and perseverance they both had when things became difficult.

I would like to thank my mentors Dr. Phillip Colella and Dan Martin for welcoming me with open arms and giving me a space where I could develop as a scientist and as a person.

Thank you to Berkeley Lab for creating programs like this and giving students like me the opportunity to learn and see potential career opportunities.

Thank you, Bates College, for supporting me in my endeavors to learn off campus this summer.

Thank you to my friends and family for supporting me and keeping in touch with me despite the time difference.

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Visiting Faculty Program (VFP).