FLEKS: A Flexible Particle-in-Cell code for Multi-Scale Plasma Simulations

Yuxi Chen[1a], Gábor Tóth[a], Hongyang Zhou[a], Xiantong Wang[a]

[a] *Center for Space Environment Modeling, University of Michigan, Ann Arbor, Michigan 48109, USA*

Abstract

The Magnetohydrodynamics with embedded particle-in-cell (MHD-EPIC) model has been successfully applied to global magnetospheric simulations in recent years. However, the PIC region was restricted to be a box, and it is not always feasible to cover the whole physical structure of interest with a box due to the limitation of the computational resources. The FLexible Exascale Kinetic Simulator (FLEKS), which is a new PIC code and allows a PIC region of any shape, is designed to break this restriction and extend the capabilities of the MHD-EPIC model.

FLEKS uses the Gauss's law satisfying energy-conserving semi-implicit method (GL-ECSIM) as the base PIC solver. We have also designed extra numerical techniques, such as the adaptive time stepping and particle resampling algorithms, to further improve the accuracy and flexibility of the PIC solver. The grid of FLEKS has to be Cartesian, but the active PIC region is not necessarily to be a box anymore since any Cartesian cells can be turned off. Furthermore, FLEKS supports switching on or switching off grid cells adaptively during a simulation. The initial conditions and boundary conditions of the active PIC region are provided by the coupled MHD code. FLEKS and the coupled MHD code constitute the MHD with adaptively embedded particle-in-cell (MHD-AEPIC) model.

Keywords: particle-in-cell; particle merging

[1]Corresponding author. Email address: yuxichen@umich.edu

1. Introduction

Multi-scale plasma simulations are challenging due to the limitation of computational resources. Fluid models are efficient for global simulations, but kinetic-scale physics is missing. Fully kinetic codes, such as particle-in-cell (PIC) codes and Vlasov solvers, contain electron and ion scale physics. However, it is extremely computationally expensive to resolve the global scale and the electron scale at the same time for three-dimensional (3D) global simulations. Traditional hybrid models, which usually treat electrons as a massless fluid and simulate ions with a PIC method or a Vlasov solver, incorporate ion-scale physics into global simulations by sacrificing electron-scale kinetic physics. Another class of hybrid methods embeds a kinetic code into a global fluid model so that the kinetic code can resolve the regions where the kinetic physics is important, and the fluid model handles the rest of the domain efficiency. Recently, independent groups have developed models that couple either a PIC code [1] or a Vlasov solver [2] with a fluid model.

Sugiyama and Kusano [3] demonstrated the concept of coupling a PIC code with a fluid code. The magnetohydrodynamics (MHD) with embedded particle-in-cell (MHD-EPIC) model developed by Daldorff et al. [1] is the first mature coupled model that is capable of running 3D global simulations. The MHD-EPIC model usually covers the dayside or/and the tail reconnection sites with the PIC code when it is applied to simulate the dynamics of magnetospheres [4, 5, 6, 7]. Multiple isolated PIC domains are supported so that a few regions of interest can be covered by the PIC code in one simulation [4]. However, in a MHD-EPIC simulation, a PIC region is restricted to be a box, which is not always feasible to cover the whole physical structure of interest due to either the limitation of computational resources or the complexity of the physical region. Recently, Shou et al. [8] developed the magnetohydrodynamics (MHD) with adaptively embedded particle-in-cell (MHD-AEPIC) model, which allows changing the location of an active PIC region dynamically.

In this paper, we introduce a brand new code, the FLexible Exascale Kinetic Simulator (FLEKS), which is designed and implemented as the PIC component of the MHD-AEPIC model. FLEKS shares some similarities with the work by Shou et al. [8], but FLEKS provides a more flexible grid design. FLEKS uses the parallel data structures provided by the AMReX library [9, 10]. The grid of FLEKS has to be uniform and Cartesian so far, but the active PIC region is not limited to be a box anymore since the PIC cells can

be turned off to fit the region of interest. Furthermore, FLEKS also supports switching on or switching off grid cells dynamically for MHD-AEPIC simulations. The Gauss's law satisfying energy-conserving semi-implicit method (GL-ECSIM) [11] is implemented as the base PIC solver. The time step of the semi-implicit PIC methods is usually limited by a Courant–Friedrichs–Lewy (CFL) condition in order to be accurate [12]. Since the plasm maximum characteristic speed may change significantly during a long MHD-AEPIC simulation, the simulation will be either too slow or inaccurate with a fixed time step. To keep the simulation efficient and accurate at the same time, FLEKS uses an adaptive time-stepping algorithm, which still satisfies the requirement of the energy-conserving semi-implicit method (ECSIM) [13] to keep energy conservation. Section 2 describes the adaptive grid and temporal discretization of FLEKS.

The statistical noise of macro-particles is a primary source of the numerical errors for typical PIC simulations. Dozens to hundreds of particles per cell are usually used to achieve a balance between the accuracy and the computational cost. Since there are much more macro-particles than grid cells in a kinetic PIC simulation, particle-related calculations, such as updating particle positions and velocities, usually dominate the total computational time. What is worse is that a massive parallel simulation can be further slowed down gradually due to the imbalance of macro-particle numbers among the CPU or GPU cores. On the other hand, the decreasing of the macro-particle numbers in some cells increases the statistical noise and reduces the accuracy. A particle resampling algorithm that is able to control the macro-particle number per cell is crucial for improving both the simulation efficiency and accuracy. More macro-particles should be populated into the cells, which contain fewer macro-particles than required, to represent the plasma velocity-space distributions more accurately. This goal is usually achieved by splitting particles. In the cells with more macro-particles than a threshold, a particle merging algorithm should be applied to reduce the macro-particle number to speed up simulations. A particle resampling algorithm is even much more crucial for a PIC code with adaptive mesh refinement, where the motion of macro-particles between the coarse and fine cells alters the macro-particle number per cell dramatically [14, 15].

Both the particle splitting and particle merging processes replace the original particles with a set of new particles. Lapenta [16] suggested that the replacement should maintain the following properties:

- The plasma moments on the simulation grid, which are used to update electric and magnetic fields, should not be changed by the replacement.
- The replacement should keep the original particle phase-space distributions.

It is more challenging to achieve these two goals for a particle merging algorithm than a particle splitting algorithm because it is inevitable to lose information when replacing original particles with fewer particles. A few algorithms have been designed to merge particles. Lapenta [16] introduced two algorithms to merge particles that are close to each other in the phase space. The algorithm C1 merges two particles into one, and the algorithm C2 merges three particles into two. The algorithm C2 conserves the mass, momentum, and energy of the particles, and also the charge densities on the grid, but it is not straightforward to extend to 2D and 3D. Vranic et al. [17] also proposed an algorithm to merge particles into two new particles while conserving the overall mass, momentum, and energy, and the original particles are chosen by binning particles in the momentum space. Instead of merging a few particles into one or two, the algorithms designed by Assous et al. [18], Welch et al. [19], Pfeiffer et al. [20], and Faghihi et al. [21] use a set of particles to replace the old ones. Assous et al. [18] and Welch et al. [19] focused on the conservation of the grid quantities, but the fine structures in the velocity space may not be well preserved. Pfeiffer et al. [20] generated the new particle velocities from a distribution function and adjusted the velocities to conserve energy afterward. Faghihi et al. [21] created new particles with a uniform distribution inside a phase-space bin, and adjusted the weights to conserve the moments. As a general rule, the particles selected for merging should be close to each other in the phase-space to minimize the error that is introduced by merging. Besides the method of binning the velocity space [17, 21], Teunissen and Ebert [22] applied a k-d tree to find the particles that are closest to each other, and Luu et al. [23] showed how to partition particles with the Voronoi diagram.

When FLEKS is applied to simulate global phenomena as part of the MHD-AEPIC model, the simulation time is usually long enough so that the local macro-particle numbers may reduce or increase significantly. Splitting particles in the low particle number cells improves statistical representation and reduces noise. Merging particles alleviates load imbalance and speeds up simulations. Our particle resampling algorithms are described in section 4.

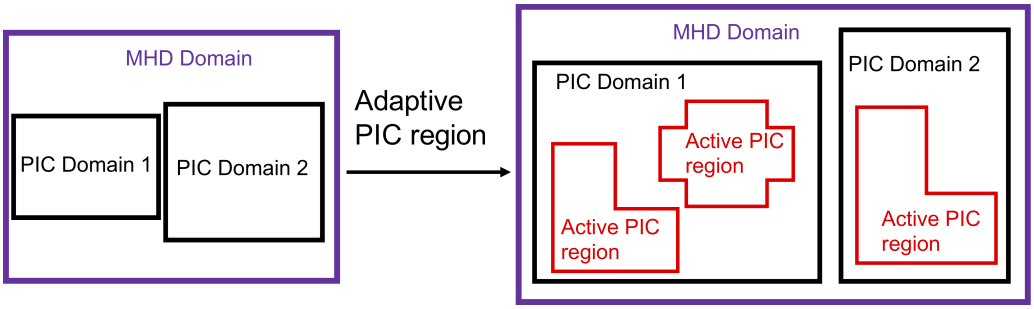

Figure 1: A schematic shows the grid structure of the MHD-AEPIC model.

Tracking the motion of macro-particles is useful for learning the trajectory and energization of plasma, so FLEKS provides a parallel test particle module to follow the motion of macro-particles and save the massive data to disk. The test particle module can be used either as a module of the PIC code, or as an independent component to directly couple to the MHD model. Section 5 describes the implementation details of the test particle module.

The paper is organized as follows. Section 2 describes the grid design of FLEKS. Section 3 introduces the adaptive time time step. Section 4 focuses on the particle merging algorithm. Section 5 presents the implementation of the test particle module. Section 6 shows the numerical tests to demonstrate the capability of the adaptive active PIC regions, the role of the particle resampling algorithms, and the parallel efficiency of FLEKS. Finally, section 7 presents the conclusions.

2. Adaptive grid

Since the MHD-EPIC model was developed by Daldorff et al. [1], we have developed new features to make it more flexible to use. It supports multiple independent PIC domains to cover several regions of interest [4], and it also allows rotating a PIC box domain to align with the simulation objective [24]. However, a box is not always feasible and efficient to cover a region of interest. For example, if a PIC box is used to cover the whole dayside magnetopause, which is close to a paraboloid, the box will cut through the planet and introduce extra difficulties, and the PIC box will also contain a large portion of cells where the kinetic effects are not important to slow down the simulation. A flexible grid that allows an active PIC region of

a paraboloid to fit the magnetopause solves the problems. A dynamically adaptive grid is also useful to improve the efficiency of some simulations. For instance, the near-Earth X-line may move from the inner magnetotail to the middle or even far magnetotail [25], and an adaptive grid that only covers the environment around the X-line is much more efficient compared to a large PIC box that covers the whole magnetotail. The MHD-AEPIC model is designed to solve these problems, and FLEKS is the key component. Figure 1 is a carton that shows the concept of the MHD-AEPIC model.

FLEKS still requires the shape of a PIC domain to be a box, and the grid has to be uniform. But it allows switching off part of the cells to fit a region of any shape. The most straightforward approach is using a bit-wise array to switch on/off each cell. However, we make the algorithm a bit more sophisticated. We divide the whole PIC domain into patches (Figure 2(a)). Each patch contains N cells in each direction, and one can turn on or turn off each patch. The patch size N is required to be larger or equal to 2. FLEKS does not allow switching on/off each cell independently (N=1) for the following reason. FLEKS requires two ghost cell layers. If N=1, the boundary ghost cells of an active region may overlap with the physical cells of another active region, and hence introduces more difficulties to handle the boundary ghost cells. A large patch size also benefits the coupling efficiency. In MHD-AEPIC simulations, the fluid model controls the status of the patches based on either geometric or physical criteria. The fluid model passes the patch status bit-wise array to FLEKS through the Message Passing Interface (MPI), and the size of this array is reduced significantly with a large patch size N. In this paper, we use 'active' to describe the patches or cells that are switched on. The active cells do not have to be connected, and the boundary ghost cells of the active regions are filled in with the information obtained from the fluid model [1]. Figure 2(a) shows an example that contains two separated active regions.

FLEKS uses the data structures provided by the AMReX library to store the fields and also the particles. After the patch status array is obtained from the fluid model, FLEKS uses the functions provided by the AMReX library to divide the active regions into blocks. AMRex does not require all the blocks to have the same size. We note that the patch and the block are two independent concepts. The patches are only used to activate or deactivate cells. For example, the 'L' shape active region in Figure 2(a) consists of 3 patches and it can be divided into 2 blocks (Figure 2(b)).

FLEKS allows activating or deactivating patches during a simulation. If

the active regions change, FLEKS will produce a new set of blocks to cover the new active regions. With the function provided by AMReX, FLEKS copies the fields and particles from the old blocks to the new ones for the cells that are already active and deletes the information of the newly deactivated cells. The newly activated cells are filled in with the information obtained from the fluid model as what is done for FLEKS initialization.

FLEKS has two ghost cell layers, but the outer layer is only used to receive and store the magnetic fields, which are necessary for calculating currents on the nodes of the inner ghost cell layer from $\vec{J} = \nabla \times \vec{B}$. The currents are used to generate particles with correct velocities in the inner layer ghost cells. To simplify the description, we ignore the outer layer in Figure 2(c) and also in the rest of the paper unless otherwise specified. The principle of setting boundary conditions of the electromagnetic fields and the particles is still the same as the MHD-EPIC algorithm with a box PIC region [1]. However, the non-box shape of an active region introduces some extra implementation difficulties. They are three types of ghost cells for a block: the internal ghost cells (blue cells in Figure 2(c)), the exclusive boundary ghost cells (gray cells in Figure 2(c)) and the shared boundary ghost cells (cyan cells in Figure 2(c)). The internal ghost cells are not boundary cells, and there is no need to apply boundary conditions. The exclusive boundary ghost cells are not overlapped with any cells of the neighbor blocks, and they should be filled in with macro-particles as the particle boundary condition. The shared boundary ghost cells are overlapped with the boundary ghost cells of the neighbor blocks. Only one of these blocks should generate boundary particles. Here is the algorithm to choose the block. The first step is to distinguish the boundary ghost cells from the internal ghost cells. Then, for each boundary ghost cell, either the exclusive type or the shared type, we loop through its at most 26 neighbor cells (3D) in a certain order (we choose to loop through all the face neighbors first, then the edge neighbors, and finally the corner neighbors), skip the cells does not exist and find out the first neighbor cell that is either physical cell or internal ghost cell. If the location of this neighbor cell is inside the physical domain of this block, this block should generate particles inside this boundary ghost cell. For example, in Figure 2(c)), C1 and C3 are overlapped with each other. We loop through the neighbor cells of C1 (C3) and find C2 (C4) is its first neighbor cell that is either a physical cell or an internal ghost cell, and block-1 (block-2) should (not) generate particles in C1 (C3) since C2 (C4) is (not) inside block-1 (block-2).

The electric fields are node-based in FLEKS. For a node that is shared by multiple blocks, such as the one indicated by red-cross in Figure 2(c)), only one block should take care of the shared node when solving the linear equation of the electric fields. The algorithm described in the previous paragraph is also applied to choose the proper block for the shared nodes.

3. Adaptive time-stepping

The time step of the energy-conserving semi-implicit method (ECSIM) is subject to the accuracy condition $v_{rms}\Delta t/\Delta x < 1$ just as other semi-implicit PIC methods [12], where v_{rms} is the maximum root mean square of macro-particle velocities. For a long MHD-AEPIC simulation, v_{rms} may vary significantly, so an adaptive time-stepping algorithm that adjusts time-step accordingly will improve the efficiency and accuracy simulations. However, the energy conservation property of ECSIM is sensitive to the temporal discretization scheme, and the adaptive time-stepping algorithm should not break the conservation.

Our adaptive time-stepping algorithm is summarized in Figure 3. At the end of one cycle, Both the electromagnetic fields and the particle velocities are at time stage t^n, and the particle locations are at the staggered stage $t^{n+1/2}$. The difference between $t^{n+1/2}$ and t^n is $t^{n+1/2} - t^n = \Delta t^n/2$.The maximum speed v_{rms} can be obtained with the particle velocities at t^n, and a new time step Δt^{n+1} can be calculated from $\Delta t^{n+1} = \text{CFL} \cdot \Delta x/v_{rms}$. However, during the next cycle to update the electromagnetic fields and particle velocities from t^n to t^{n+1}, the time step should be t^n instead of t^{n+1}, so that the particle locations $X^{n+1/2}$ are still at the middle of t^n and t^{n+1} and the energy conservation property of ECSIM is preserved. In order to adjust the time step for the next cycle, we use the time step $\Delta t^n/2 + \Delta t^{n+1}/2$ for updating the particle locations from $X^{n+1/2}$ to $X^{n+3/2}$. Since the velocities V^{n+1} are not at the middle of $X^{n+1/2}$ of $X^{n+3/2}$ anymore, the second-order accuracy of updating particle locations is not satisfied. In practice, the speed v_{rms} varies gradually and the difference between Δt^n and Δt^{n+1} is very small, so the lose of accuracy is negligible.

4. Particle resampling

We implemented particle resampling algorithms to control the macro-particle number of each cell. Every computational cycle, a particle splitting

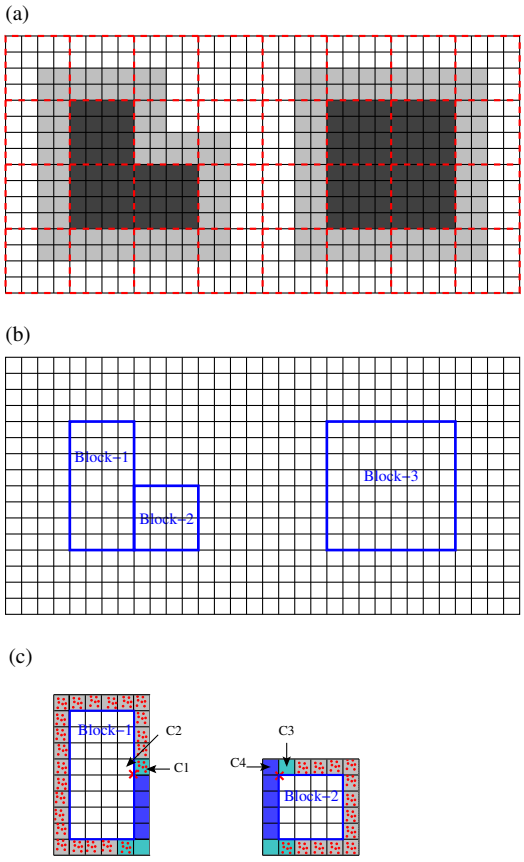

Figure 2: The black lines represent the cells of a PIC domain. The red dashed lines in (a) show the patches, and one patch contains 4×4 cells in this example. In (a), the active patches/cells are colored by dark gray, and light gray colors the ghost cells of the active cells. (b) shows the blocks of the active regions. (c) shows the inner layer of the ghost cells of two blocks, and the red dots represent the macro-particles that are generated in the ghost cells as the particle boundary condition. Blue ghost cells are internal ghost cells, which are overlapped with the physical cells of the neighbor blocks. The gray cells are exclusive boundary ghost cells, and they should be filled in with macro-particles as the boundary condition. The cyan cells are also boundary ghost cells, but they are overlapped with the boundary ghost cells of the neighbor blocks. One of the blocks should generate boundary particles.

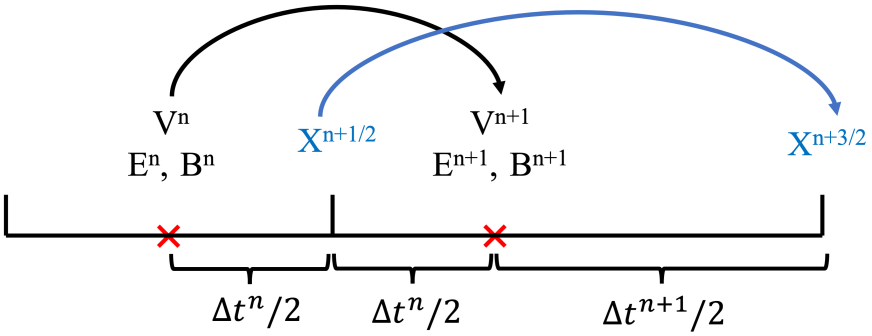

Figure 3: The adaptive temporal discretization.

(merging) algorithm is applied to produce (remove) macro-particles for the cells that contain fewer (more) particles than the low (high) threshold. The goal of splitting and merging is to keep the number of particles per cell (PPC) close to a constant in the whole computational domain. This constant number is the initial PPC in FLEKS. Essentially, the particle resampling algorithms use a new set of particles to replace the old ones. Our guiding principle of designing the algorithms is that the replacement should keep the original particle phase-space distribution as much as possible. In order to conveniently apply the resampling algorithms, FLEKS stores the particle data cell by cell.

4.1. Particle splitting

Our particle splitting algorithm is essentially the same as the one introduced by Lapenta[16], in which one particle is split into two children particles. The children particles have the same velocity as their parent particle, but their location is oppositely displaced slightly along the velocity direction. We choose to displace the new particles along the velocity direction so that the orbits of the new particles are still close to the orbit of the old particle.

Initially, the particles that are close to each other have similar weights, but the weights may become quite different later due to the transport of particles and the effect of particle resampling. We choose to split the heaviest particles to minimize the particle weight variance.

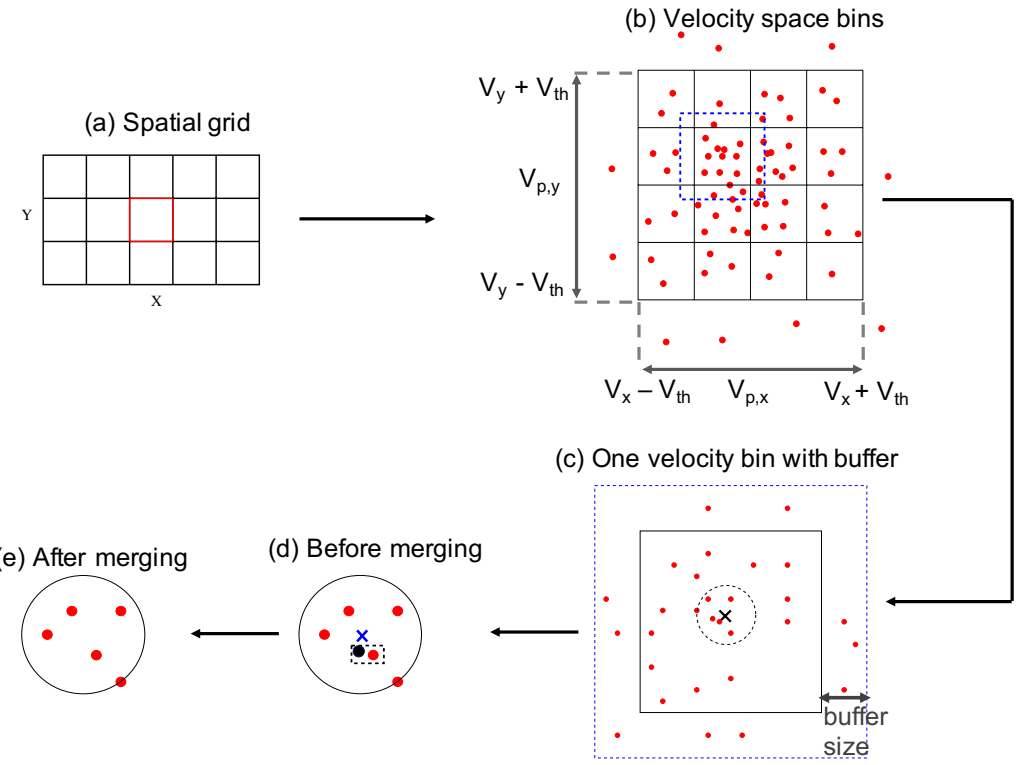

Figure 4: The steps of merging macro-particles.

4.2. Particle merging

The essence of particle merging is replacing a set of particles with a new set, and the new set contains fewer particles than the old one. Particle merging reduces the particle number in some cells and improves load balancing. Particle resampling has a negative impact on the accuracy of a simulation because (1) the replacement introduces errors, and (2) fewer particles lead to larger statistical noise in the subsequent simulation. The increasing of statistical noise is inevitable, but the errors caused by the replacement can be minimized with a proper merging algorithm.

5. Test particle module

An independent test particle (TP) module is designed to track the motion of the macro-particles for FLEKS. It can be used either as an auxiliary component of the PIC algorithm or as an independent component, which is directly coupled with the MHD model. Similar to the macro-particles of the PIC code, the test particles are also generated based on local plasma fluid quantities that are obtained from the MHD model. The TP module also uses the same algorithm to move particles as the GL-ECSIM algorithm. When the TP module is used with the PIC component together, the TP module shares the same grid layout as the PIC component and uses the electromagnetic fields calculated by PIC to update test particles. When the PIC component is turned off, the TP module can directly obtain the grid structure and electromagnetic fields from the MHD model. FLEKS is a pure test particle code for this case. Compared to the embedded PIC simulations, the test particle simulations are only one-way coupled, i.e., the MHD model provides the electromagnetic fields for FLEKS, but there is not any feedback from FLEKS to the MHD model.

In a three-dimensional (3D) simulation, it is common to track the motion of millions of test particles, and a few thousand steps of the update will easily produce a few hundred Gigabytes of particle trajectory data. The test particle module should organize the data properly to improve both the IO performance of writing data to disk and also the efficiency of finding all the data of a certain particle for data analysis. To reduce the IO frequency, the TP module of FLEKS saves the particle trajectory data every 100 cycles, and all the processors write to the same file with MPI-IO APIs. We note that if a test particle moves from one processor to another in the middle of two IO operations, its trajectory data should also be transferred to the destination processor. Besides the particle trajectory data file, a particle ID list file, which maps a particle ID to its data location in the particle data file, is also created. An example of these two files is shown in Figure 5. With this file structure, it is efficient to find all the trajectory data of a certain particle for data analysis.

6. Numerical tests

6.1. Two-dimensional double-current-sheet magnetic reconnection

The two-dimensional magnetic reconnection problem is widely used to test plasma simulation codes. The double-current-sheet setup allows periodic

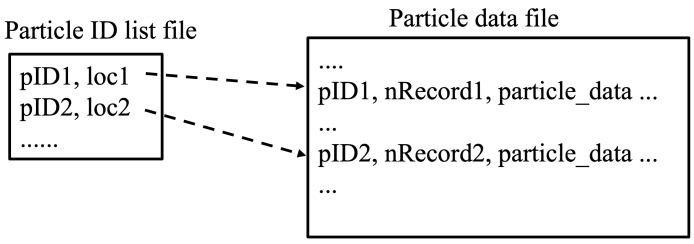

Figure 5: The file structures for storing test particles.

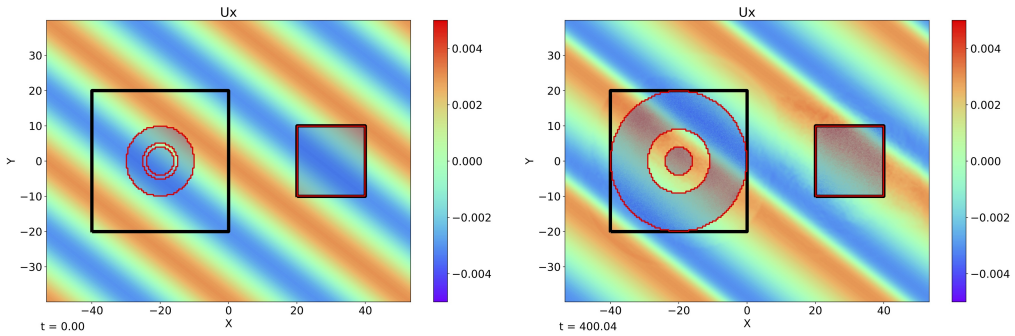

Figure 6: 2D fast wave test.

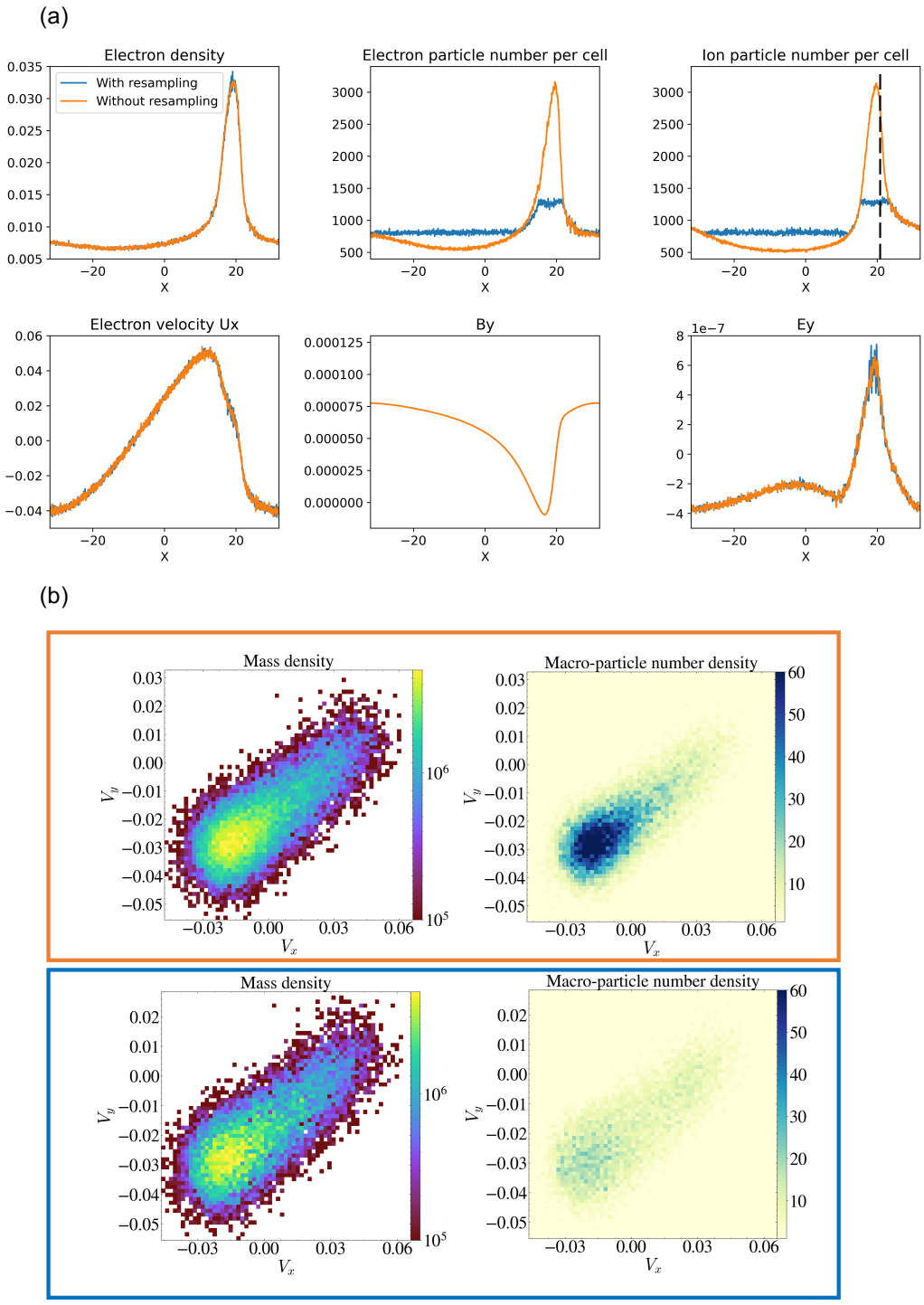

Figure 7: 1D fast wave.

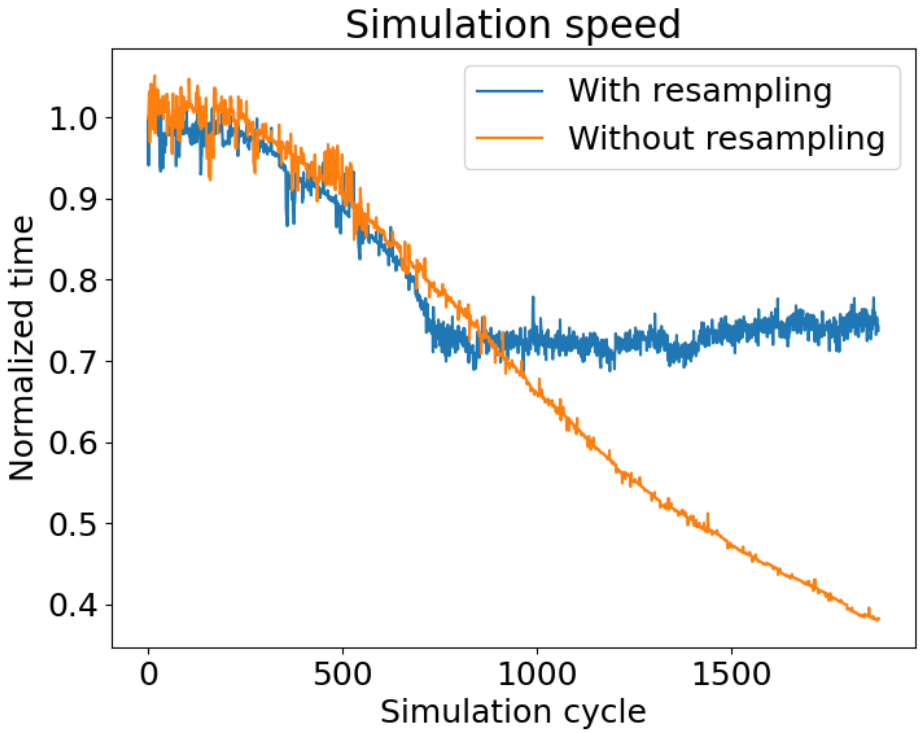

Figure 8: Simulation speed.

boundary conditions for both directions. Here we use a setup based on the GEM-challenge [26].

The initial condition is set to satisfy the fluid force balance for both electrons and ions [27]. The simulation domain is $-12.8 < x < 12.8$ and $-6.4 < y < 6.4$ in normalized CGS unit. The speed of light is set to be $c = 1$. The ion density is uniform and $n_i = 0.0975$. The ion plasma frequency is $\omega_{pi} = \sqrt{\frac{4\pi n_i e^2}{m_i}} = 1.107$ and the ion inertial length $d_i = c/\omega_{pi} = 0.903$ since $m_i = 1$ and $q_i = -q_e = 1$. A reduced ion-electron mass ratio $m_i/m_e = 25$ is used, so the electron skin depth is about $d_e = d_i/5 = 0.18$. Initially, there is no charge separation, $n_e = n_i$, and the electric field is $\mathbf{E} = 0$.

The background magnetic filed is

$$B_x = B_0 \left(-1 + \tanh \frac{y - y_B}{\delta} + \tanh \frac{y_T - y}{\delta} \right) \tag{1}$$

where $B_0 = 0.07$, the positions of the two current sheets are $y_B = -3.2$ and $y_T = 3.2$, respectively, and the width of the currents sheets are controlled by $\delta = 0.5$. The electrons have a velocity in the z-direction to generate current equal to the curl of the magnetic field, i.e, $J_z = n_e q_e u_{e,z} = -\partial B_x/\partial y$. The ion pressure p_i is uniform in the whole domain. Far away from the current sheets, the ion plasma beta is 1, and the electron pressure is 1/5 of the ion pressure. Near the current sheet, the electrons are heated to balance the magnetic field gradient force, which is the same as the Lorentz force $-n_e q_e u_{e,z} B_x$. This unperturbed initial condition is in fluid force balance [27].

A perturbation is added to excite the reconnection. The magnetic field perturbation vector potential is $A_x = 0$, $A_y = 0$ and:

$$\begin{aligned} A_z = A_0 B_0 \Big\{ & - e^{-\frac{(x-x_T)^2}{G_x^2} - \frac{(y-y_T)^2}{G_y^2}} \cos\left[k_x(x - x_T)\right] \cos\left[k_y(y - y_T)\right] \\ & + e^{-\frac{(x-x_B)^2}{G_x^2} - \frac{(y-y_B)^2}{G_y^2}} \cos\left[k_x(x - x_B)\right] \cos\left[k_y(y - y_B)\right] \Big\} \end{aligned} \tag{2}$$

where the perturbation amplitude is set by $A_0 = 0.1$, the locations along the top and bottom current sheets are $x_T = 6.4$ and $x_B = -6.4$, respectively, the width of Gaussian profiles are $G_x = G_y = 0.5$, and the wave vectors are $k_x = 2\pi/25.6$ and $k_y = 2\pi/12.8$. Since these two reconnection sites, i.e., the bottom left one at (x_B, y_B) and the top right one at (x_T, y_T), produce the same signatures, we only plot and discuss the bottom left reconnection site for simplicity.

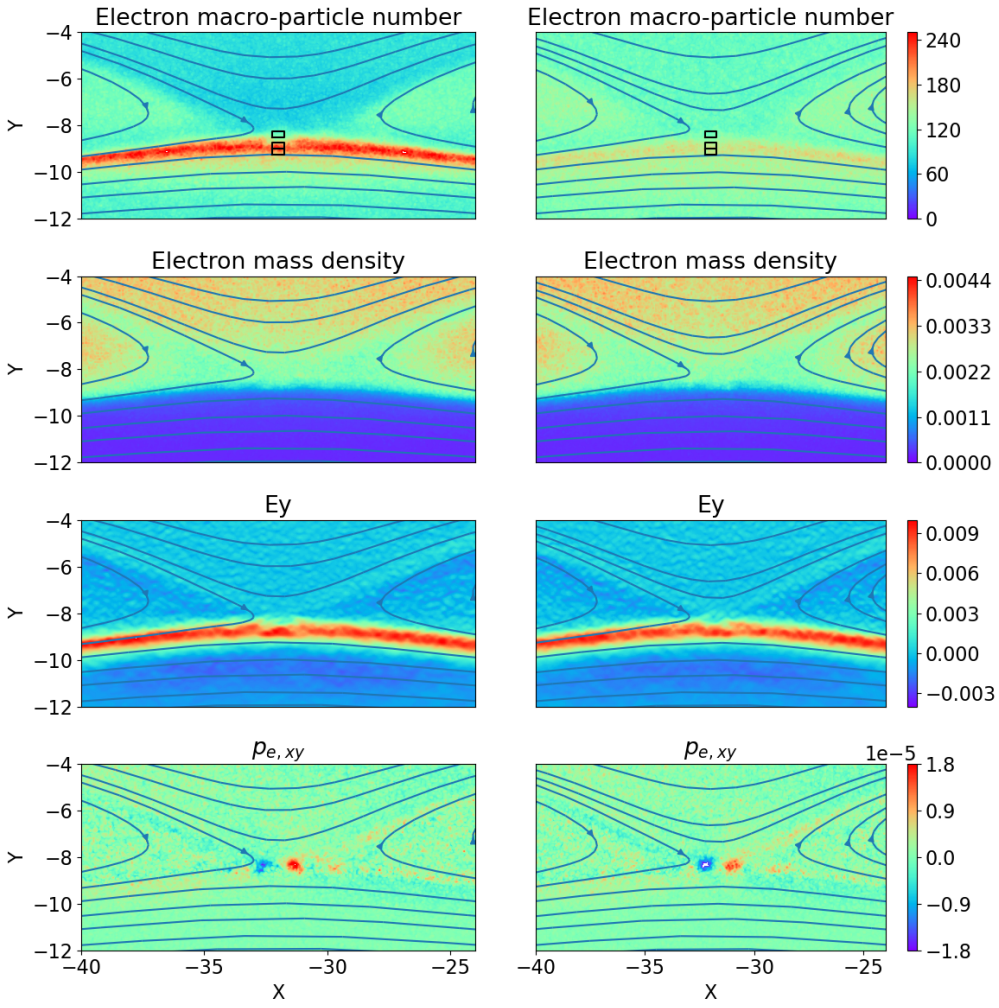

Figure 9: 2D magnetic reconnection results with (right column) or without (left column) particle resampling.

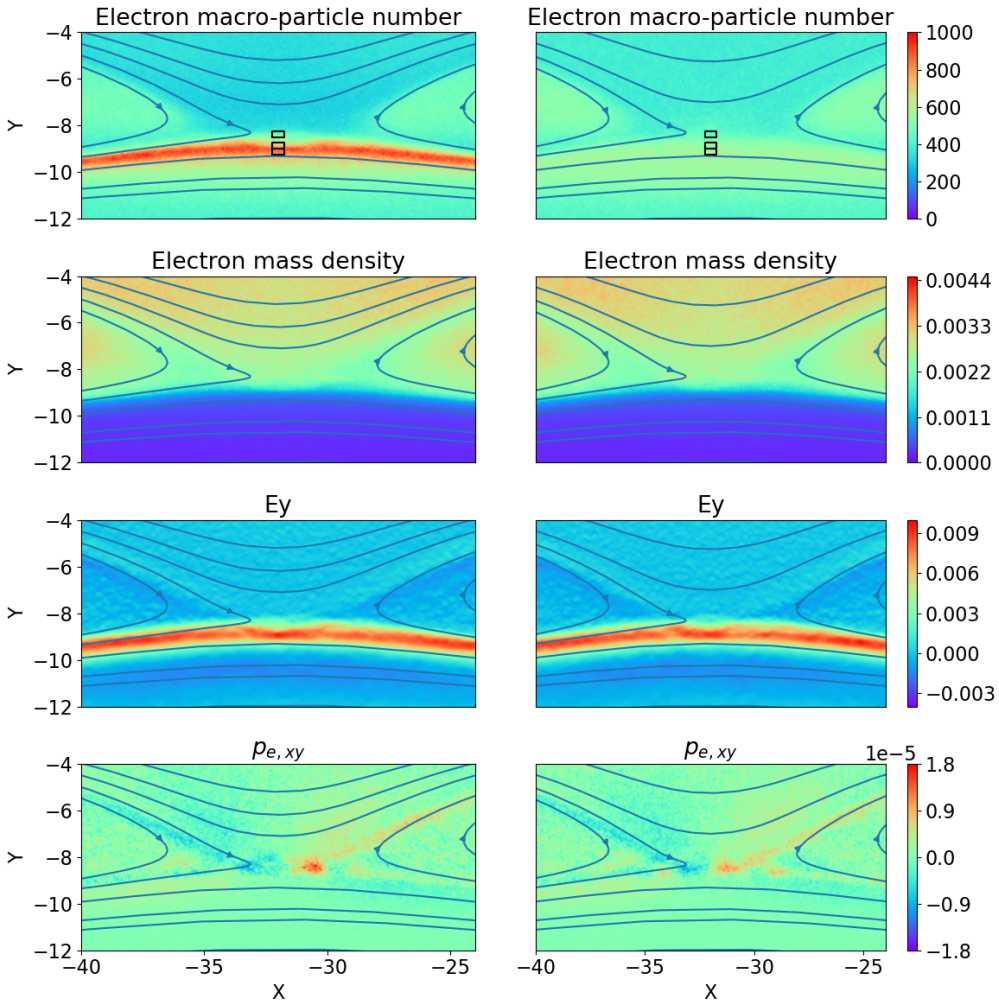

Figure 10: 2D magnetic reconnection results with (right column) or without (left column) particle resampling.

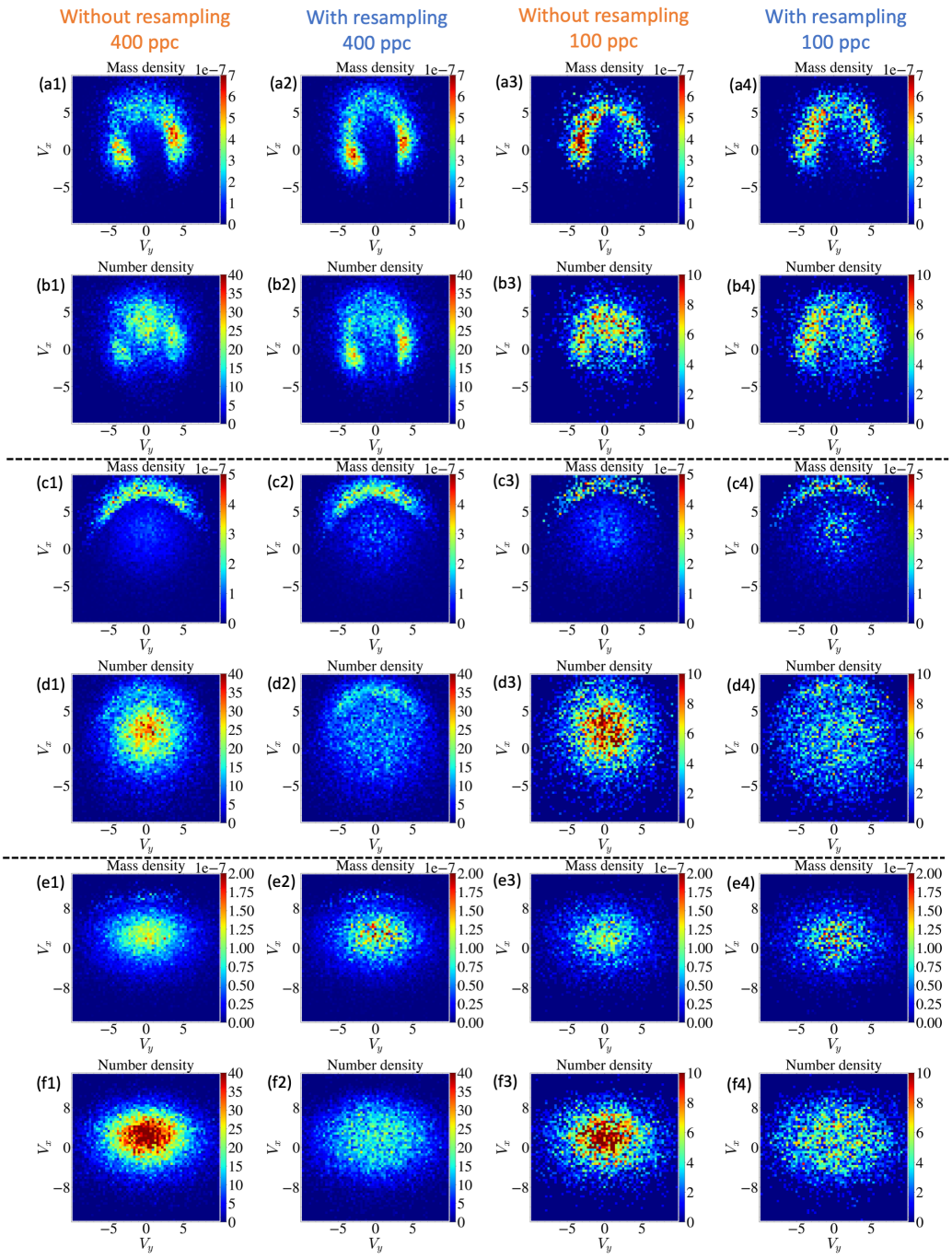

Figure 11: Phase space distributions.

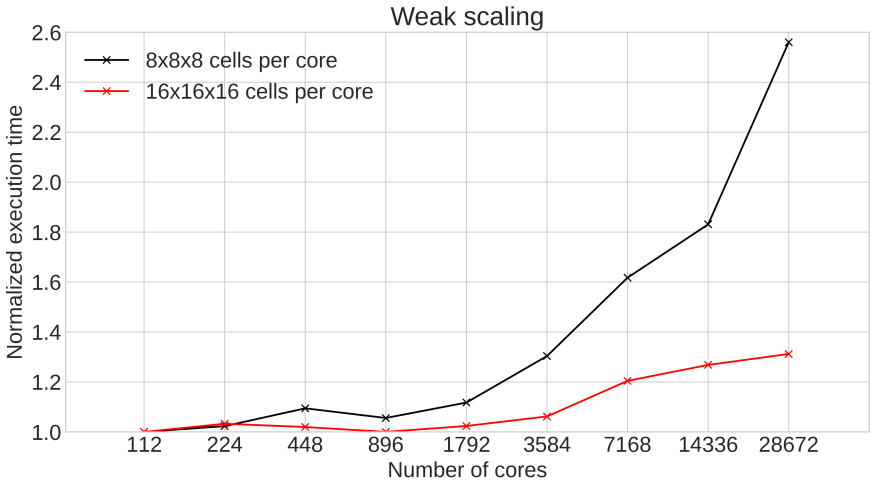

Figure 12: Weak scaling results.

7. Conclusion

In this paper, we present the algorithms and implementation of FLEKS. The adaptive grid allows FLEKS covering a region of any shape. The adaptive temporal discretization and particle resampling algorithms improves the accuracy and efficiency of simulations.

Acknowledgments: This work was supported by the INSPIRE NSF grant PHY-1513379 and the NSF PREEVENTS grant 1663800. Computational resources supporting this work were provided on the Blue Waters super computer by the NSF PRAC grant ACI-1640510, on the Pleiades computer by NASA High-End Computing (HEC) Program through the NASA Advanced Supercomputing (NAS) Division at Ames Research Center, and from Yellowstone (ark:/85065/d7wd3xhc) provided by NCAR's Computational and Information Systems Laboratory, sponsored by the National Science Foundation.

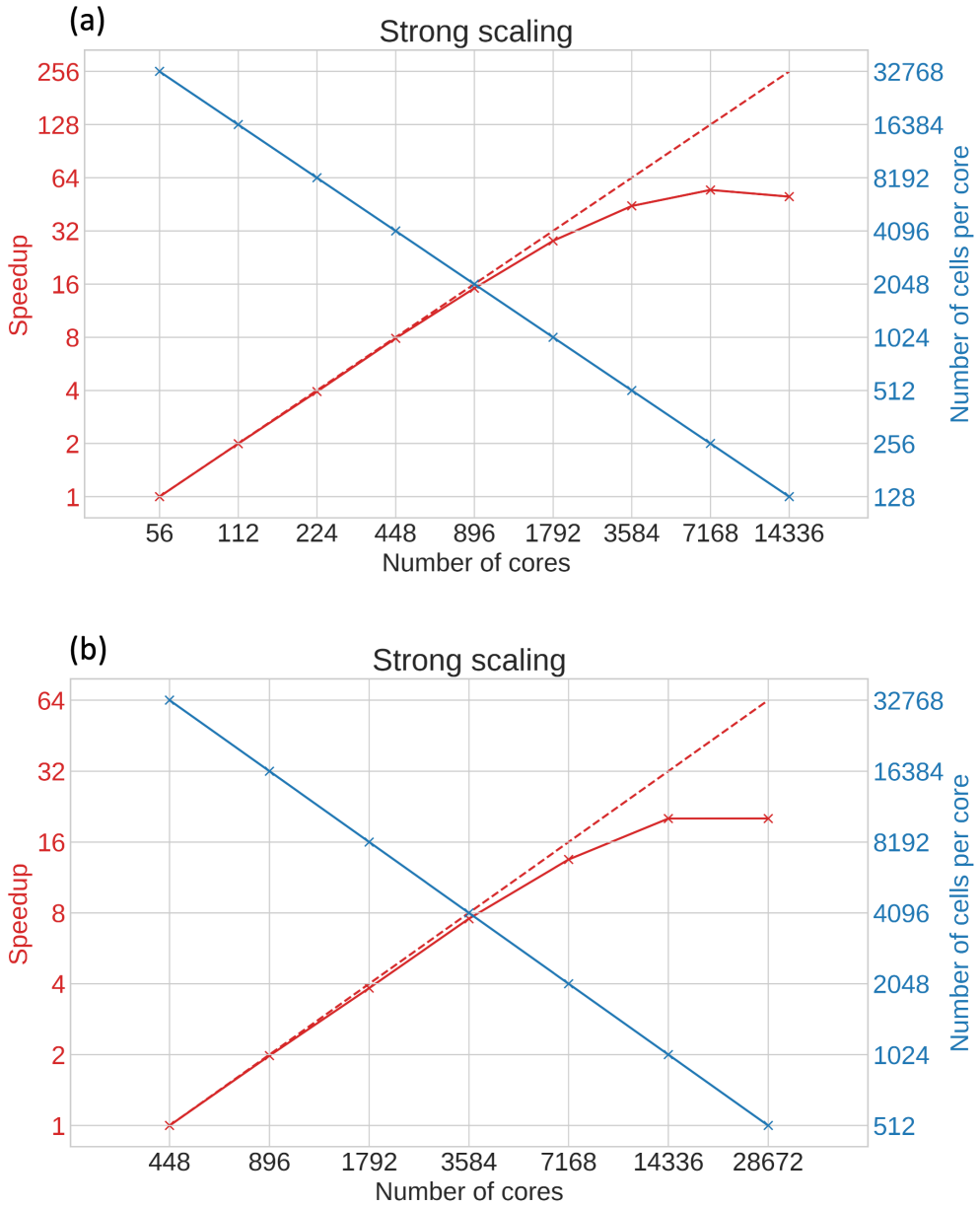

Figure 13: Strong scaling results.

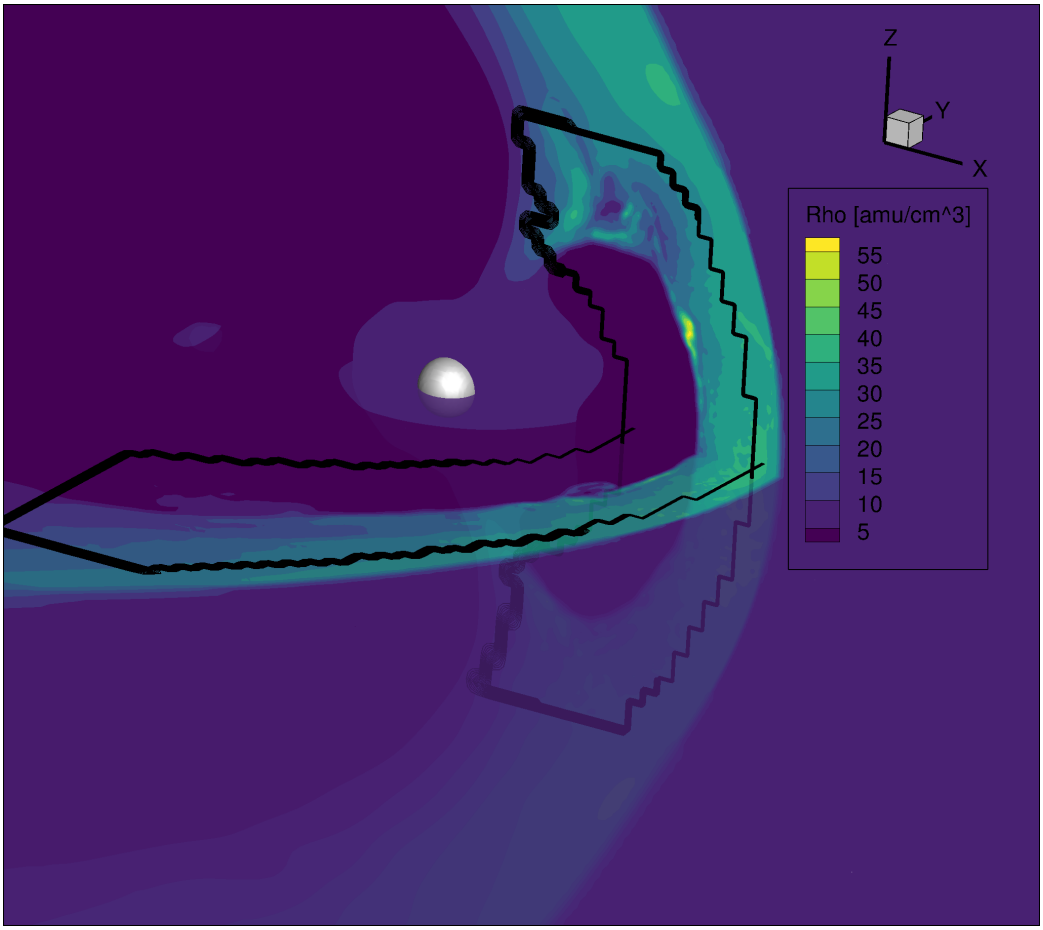

Figure 14: A MHD-AEPIC simulation of Earth's magnetosphere with the dayside magnetopaused covered by FLEKS.

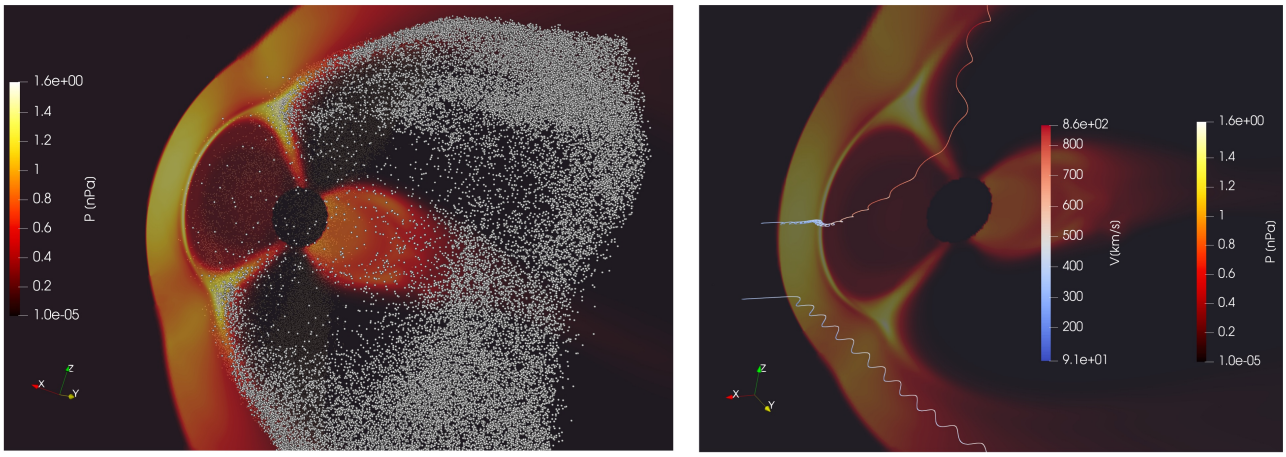

Figure 15: The locations (left) and the trajectories (right) of the test particles.

References

[1] L. K. S. Daldorff, G. Tóth, T. I. Gombosi, G. Lapenta, J. Amaya, S. Markidis, J. U. Brackbill, Two-way coupling of a global Hall magnetohydrodynamics model with a local implicit Particle-in-Cell model, J. Comput. Phys. 268 (2014) 236. doi:10.1016/j.jcp.2014.03.009.

[2] M. Rieke, T. Trost, R. Grauer, Coupled Vlasov and two-fluid codes on GPUs, Journal of Computational Physics 283 (2015) 436–452. doi:10.1016/j.jcp.2014.12.016.
URL http://dx.doi.org/10.1016/j.jcp.2014.12.016

[3] T. Sugiyama, K. Kusano, Multi-scale plasma simulation by the interlocking of magnetohydrodynamic model and particle-in-cell kinetic model, J. Comput. Phys. 227 (2007) 1340–1352. doi:10.1016/j.jcp.2007.09.011.

[4] G. Tóth, X. Jia, S. Markidis, B. Peng, Y. Chen, L. Daldorff, V. Tenishev, D. Borovikov, J. Haiducek, T. Gombosi, A. Glocer, J. Dorelli, Extended magnetohydrodynamics with embedded particle-in-cell sim-

ulation of ganymede's magnetosphere, J. Geophys. Res. 121. doi:10.1002/2015JA021997.

[5] Y. Chen, G. Tóth, P. Cassak, X. Jia, T. I. Gombosi, J. Slavin, S. Markidis, B. Peng, Global three-dimensional simulation of earth's dayside reconnection using a two-way coupled magnetohydrodynamics with embedded particle-in-cell model: initial results, J. Geophys. Res. 122 (2017) 10318. doi:10.1002/2017JA024186.

[6] Y. Chen, G. Tóth, X. Jia, J. A. Slavin, W. Sun, S. Markidis, T. I. Gombosi, J. M. Raines, Studying Dawn-Dusk Asymmetries of Mercury's Magnetotail Using MHD-EPIC Simulations, Journal of Geophysical Research: Space Physics 124 (11) (2019) 8954–8973. arXiv:1904.06753, doi:10.1029/2019JA026840.

[7] H. Zhou, G. Tóth, X. Jia, Y. Chen, S. Markidis, Embedded Kinetic Simulation of Ganymede's Magnetosphere: Improvements and Inferences, Journal of Geophysical Research: Space Physics 124 (7) (2019) 5441–5460. doi:10.1029/2019JA026643.

[8] Y. Shou, V. Tenishev, Y. Chen, G. Toth, N. Ganushkina, Magnetohydrodynamic with Adaptively Embedded Particle-in-Cell model: MHD-AEPIC, Journal of Computational Physics 446 (2021) 110656. doi:https://doi.org/10.1016/j.jcp.2021.110656.
URL https://www.sciencedirect.com/science/article/pii/S0021999121005519

[9] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, M. Katz, A. Myers, T. Nguyen, A. Nonaka, M. Rosso, S. Williams, M. Zingale, AMReX: a framework for block-structured adaptive mesh refinement, Journal of Open Source Software 4 (37) (2019) 1370. doi:10.21105/joss.01370.
URL https://doi.org/10.21105/joss.01370

[10] W. Zhang, A. Myers, K. Gott, A. Almgren, J. Bell, AMReX: Block-structured adaptive mesh refinement for multiphysics applications, International Journal of High Performance Computing Applications 0 (0) (2021) 1–19. doi:10.1177/10943420211022811.

[11] Y. Chen, G. Tóth, Gauss's law satisfying energy-conserving semi-implicit particle-in-cell method, J. Comput. Phys. 386 (2019) 632. doi:10.1016/j.jcp.2019.02.032.

[12] J. U. Brackbill, D. W. Forslund, An implicit method for electromagnetic plasma simulation in two dimensions, Journal of Computational Physics 46 (2) (1982) 271–308. doi:10.1016/0021-9991(82)90016-X.

[13] G. Lapenta, Exactly energy conserving semi-implicit particle in cell formulation, J. Comput. Phys. 334 (2017) 349. doi:10.1016/j.jcp.2017.01.002.

[14] K. Fujimoto, S. Machida, Electromagnetic full particle code with adaptive mesh refinement technique: Application to the current sheet evolution, Journal of Computational Physics 214 (2) (2006) 550–566. doi:10.1016/j.jcp.2005.10.003.

[15] K. Fujimoto, A new electromagnetic particle-in-cell model with adaptive mesh refinement for high-performance parallel computation, Journal of Computational Physics 230 (23) (2011) 8508–8526. doi:10.1016/j.jcp.2011.08.002.
URL http://dx.doi.org/10.1016/j.jcp.2011.08.002

[16] G. Lapenta, Particle rezoning for multidimensional kinetic particle-in-cell simulations, Journal of Computational Physics 181 (1) (2002) 317–337. doi:10.1006/jcph.2002.7126.

[17] M. Vranic, T. Grismayer, J. L. Martins, R. A. Fonseca, L. O. Silva, Particle merging algorithm for PIC codes, Computer Physics Communications 191 (1) (2015) 65–73. arXiv:1411.2248, doi:10.1016/j.cpc.2015.01.020.
URL http://dx.doi.org/10.1016/j.cpc.2015.01.020

[18] F. Assous, T. Pougeard Dulimbert, J. Segré, A new method for coalescing particles in PIC codes, Journal of Computational Physics 187 (2) (2003) 550–571. doi:10.1016/S0021-9991(03)00124-4.

[19] D. R. Welch, T. C. Genoni, R. E. Clark, D. V. Rose, Adaptive particle management in a particle-in-cell code, Journal of Computational Physics 227 (1) (2007) 143–155. doi:10.1016/j.jcp.2007.07.015.

[20] M. Pfeiffer, A. Mirza, C. D. Munz, S. Fasoulas, Two statistical particle split and merge methods for Particle-in-Cell codes, Computer Physics Communications 191 (1) (2015) 9–24. doi:10.1016/j.cpc.2015.01.010.
URL http://dx.doi.org/10.1016/j.cpc.2015.01.010

[21] D. Faghihi, V. Carey, C. Michoski, R. Hager, S. Janhunen, C. S. Chang, R. D. Moser, Moment preserving constrained resampling with applications to particle-in-cell methods, Journal of Computational Physics 409 (2020) 109317. doi:10.1016/j.jcp.2020.109317.
URL https://doi.org/10.1016/j.jcp.2020.109317

[22] J. Teunissen, U. Ebert, Controlling the weights of simulation particles: Adaptive particle management using k-d trees, Journal of Computational Physics 259 (2014) 318–330. arXiv:1301.1552, doi:10.1016/j.jcp.2013.12.005.
URL http://dx.doi.org/10.1016/j.jcp.2013.12.005

[23] P. T. Luu, T. Tückmantel, A. Pukhov, Voronoi particle merging algorithm for PIC codes, Computer Physics Communications 202 (2016) 165–174. arXiv:1504.00636, doi:10.1016/j.cpc.2016.01.009.
URL http://dx.doi.org/10.1016/j.cpc.2016.01.009

[24] Y. Chen, G. Tóth, H. Hietala, S. K. Vines, Y. Zou, Y. Nishimura, M. V. Silveira, Z. Guo, Y. Lin, S. Markidis, Magnetohydrodynamic with embedded particle-in-cell simulation of the Geospace Environment Modeling dayside kinetic processes challenge event, Earth and Space Sciencedoi:10.1029/2020ea001331.

[25] X. Wang, Y. Chen, G. Tóth, Global magnetohydrodynamic magnetosphere simulation with an adaptively embedded particle-in-cell model, Earth and Space Science Open Archive (2021) 21doi:10.1002/essoar.10508044.1.
URL https://doi.org/10.1002/essoar.10508044.1

[26] J. Birn, J. F. Drake, M. A. Shay, B. N. Rogers, R. E. Denton, M. Hesse, M. Kuznetsova, Z. W. Ma, A. Bhattacharjee, A. Otto, P. L. Pritchett, Geospace Environmental Modeling (GEM) magnetic reconnection challenge, J. Geophys. Res. 106 (A3) (2001) 3715–3720. doi:10.1029/1999JA900449.

[27] Z. Huang, G. Tóth, B. van der Holst, Y. Chen, T. Gombosi, A six-moment multi-fluid plasma model, J. Comput. Phys. 387 (2019) 134. doi:10.1016/j.jcp.2019.02.023.