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, which is not always feasible for covering the whole physical structure of interest. 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. It is usually used as the PIC component of the MHD with adaptively embedded particle-in-cell (MHD-AEPIC) model. FLEKS supports dynamically activating or deactivating cells to fit the regions of interest during a simulation. An adaptive time-stepping scheme is also introduced to improve the accuracy and efficiency of a long simulation. The particle number per cell may increase or decrease significantly and lead to load imbalance and large statistical noise in the cells with fewer particles. A particle splitting scheme and a particle merging algorithm are designed to limit the change of the particle number and hence improve the simulation load balancing. Both particle splitting and particle merging conserve the total mass, momentum, and energy. FLEKS also contains a test-particle module to enable tracking particle trajectories with the time-dependent electromagnetic that is obtained from a global simulation.

Keywords: particle-in-cell; particle merging; test particle; global kinetic simulation

---

[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. In recent years, 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 large-scale simulations. The MHD-EPIC model usually covers the dayside or/and the tail magnetic 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 an MHD-EPIC simulation, a PIC region is restricted to be a box, which is not always efficient or feasible to cover the whole physical structure of interest due to either the limitation of computational resources or geometric 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 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 plasma 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 numerical error source of typical PIC simulations. Dozens to hundreds of particles per cell are usually used to achieve a balance between the accuracy and computational cost. Since there are much more macro-particles than the 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 number 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 the 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 for 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.

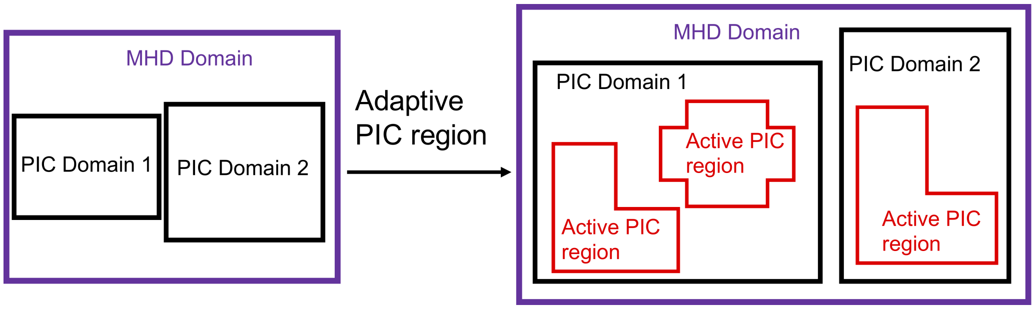


Figure 1: A schematic shows the improvement of the MHD-AEPIC (right) model from the MHD-EPIC (left) model.

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 particle trajectory 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-stepping scheme. Section 4 focuses on the particle splitting and particle merging algorithms. Section 5 discusses 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, the parallel efficiency of FLEKS, and examples of global simulations with 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 or 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 applying an active PIC, which is in the shape of a paraboloid, to fit the magnetopause would solve 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 [25]. The fluid model passes the bit-wise patch status 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 the word ‘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 coupling algorithm [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 neighboring blocks, and they should be filled in with new macro-particles as the particle boundary condition. The shared boundary ghost cells are overlapped with the boundary ghost cells of the neighboring blocks, and only one of these blocks should generate boundary particles. Here is the algorithm to choose the block for populating new particles. 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 neighboring cells (3D) in a fixed 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 neighboring cell that is either physical cell or internal ghost cell. If this neighboring 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 neighboring cells of C1 (C3) and find C2 (C4) is its first neighboring 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 a red-cross in Figure 2(c)), only one block should take care of the shared node when solving the linear equations of the electric fields. The aforementioned algorithm is also applied to choose the proper block for a shared node.

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 simulation efficiency and accuracy. 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 $\Delta t^{n+1} = \mathrm{CFL} \cdot \Delta x/v_{rms}$. 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 location $X^{n+1/2}$ is 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 location from $X^{n+1/2}$ to $X^{n+3/2}$. Since the velocity V^{n+1} is 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 small, so the loss of accuracy is negligible.

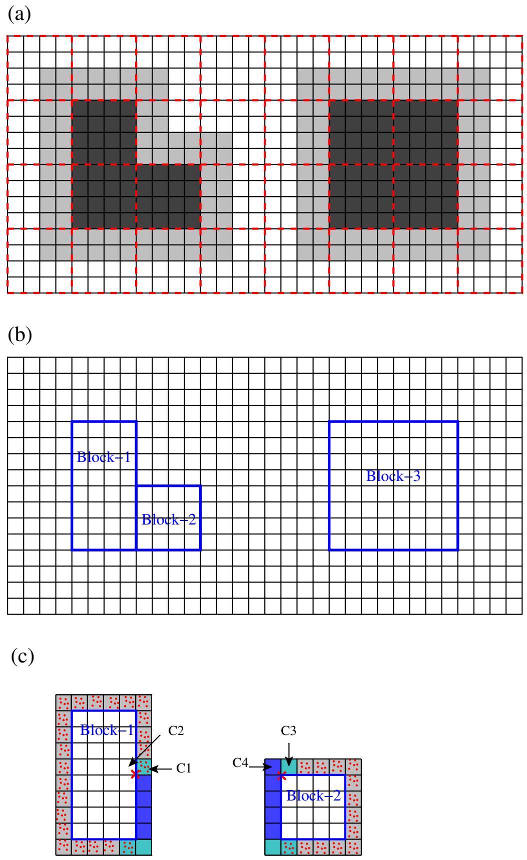


Figure 2: The black lines represent the cells of a PIC domain. The red dashed lines in (a) show the patches, and one patch contains 4 × 4 cells in this example. In (a), the active patches/cells are colored by dark gray, and light gray area represents the ghost cells of the active PIC regions. (b) shows 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 neighboring 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 neighboring blocks, and only one of the blocks should generate boundary particles.

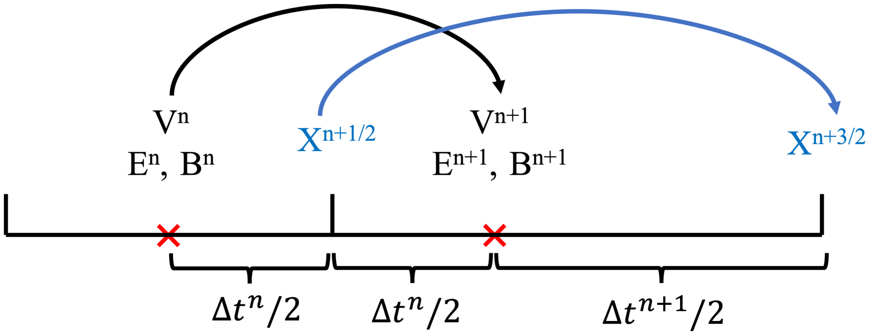


Figure 3: The adaptive temporal discretization.

4. Particle resampling

Particle resampling algorithms are used to control the macro-particle number of each cell. At the end of every computational cycle, a particle splitting (merging) algorithm is applied to generate (remove) macro-particles for the cells that contain fewer (more) particles than the splitting (merging) threshold. The goal of splitting and merging is to stop the number of particles per cell (ppc) from dropping or increasing continually. 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 preserve 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 locations are oppositely displaced slightly along the velocity direction. By displacing the new particles along the velocity direction, the orbits of the new particles are still close to the orbit of the old particle.

The particle splitting will be triggered for the cells with ppc less than the splitting threshold, which is 80% of the initial ppc by default, and we will use this number for all the simulations presented in section 6. 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 also the particle splitting and merging. For each cell, we choose to split the heaviest N particles to minimize the particle weight variance, where N is the difference between the current ppc and the splitting threshold.

4.2. Particle merging

The essence of particle merging is replacing a set of particles with a new set, which contains fewer particles than the old set. Particle merging reduces the particle number in some cells and improves simulation speed. In general, particle merging 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 statistical noise increasing is inevitable, but the errors caused by the replacement can be minimized with a proper merging algorithm.

Our particle merging algorithm consists of two steps: (1) selecting 6 particles that are close to each other in the phase space, and (2) merging these 6 particles into 5. In the following subsections, we will describe the merging step first, and what follows is the selecting step.

4.2.1. Merging particles

Once the 6 old particles for merging have been obtained from the selecting step, we use 5 new particles to replace them. The replacement should not alter the original phase space distribution significantly. However, it is not trivial to quantitatively measure the change of the phase space. The conservation of total mass, momentum, and energy can be used as a guidance and indicator of preserving phase space structure. However, satisfying the conservation property is not good enough, it is still possible that the new particle set occupies a very different velocity space volume than the old set. Previous methods [26, 17] usually generate new particles with velocities that are different from the velocities of the old particles, and extra actions are usually required to ensure the new particles are not too far away from the old ones in the velocity space. To avoid this difficulty, we choose to delete one of the 6 old particles and distribute its mass to the rest 5 particles under the constrain of conserving total mass, momentum, and energy. The weights of these 5 particles change, but their velocities inherit from the old ones, so the new particle set occupies almost the same phase space volume as the old set.

The total mass, momentum and energy of the old particle set are:

$$w_t = \sum_{i=1}^{N_{old}} w_i \tag{1}$$

$$\mathbf{p}_t = \sum_{i}^{N_{old}} w_i \mathbf{v}_i \tag{2}$$

$$e_t = \sum_{i=1}^{N_{old}} \frac{1}{2} w_i |\mathbf{v}_i|^2, \tag{3}$$

where $N_{old} = 6$. From these 6 particles, we find the pair that is closest to each other in the 6D phase space (Figure 4(d)), then remove the lighter one of this pair and adjust the weights of the rest 5 particles to satisfy the conservation requirement:

$$w_t = \sum_{i=1}^{N_{new}} w_{i,new} \tag{4}$$

$$\mathbf{p}_t = \sum_{i}^{N_{new}} w_{i,new} \mathbf{v}_i \tag{5}$$

$$e_t = \sum_{i=1}^{N_{new}} \frac{1}{2} w_{i,new} |\mathbf{v}_i|^2 \tag{6}$$

$$w_{i,new} > 0, \tag{7}$$

where we choose $N_{new} = 5$ since there are 5 quantities to conserve, the velocities $\mathbf{v}_i$ are known, and the weights $w_{i,new}$ are the unknowns of the linear equations under the constrain of $w_{i,new} > 0$. We note that if the solution does not satisfy the constrain, we skip this merging. The procedures described above need to calculate the distance in the 6D phase space, and its calculation method will be described in the next subsection.

By deleting the lighter particle from a pair that is closest to each other in the phase space, it is likely its heavier neighbor will gain most of the weight and the rest 4 particles adjust their weights relatively slightly. By inheriting the velocities and locations from the old particles, the new particles occupy almost the same phase space volume as the old particles (Figure 4(d) and (e)), so there is no room for the phase space structure to change dramatically. Compared to the schemes that allow choosing new particle velocity with fewer restrictions [17], our merging algorithm is less efficient to reduce the particle number because (1) the new particle set still contains 5 particles, and (2) the merging may fail when the constrain $w_{i,new} > 0$ can not be satisfied. If it is required, our algorithm can be easily modified to use a N_{old} that is larger than 6 by deleting $N_{old} - 5$ particles. However, as it can be seen from the following numerical test section, merging 6 particles into 5 is already efficient enough for our typical applications.

4.2.2. Selecting particles

To minimize the phase space change, the particles selected for merging should be close to each other in the 6D phase space. Several strategies have been proposed for selecting particles, including binning particles in the phase space [21], partitioning phase space with Voronoi diagram [23], and using k-d tree data structure [22]. For the sake of simplicity, we choose the binning strategy. The dimension of the 6D phase space is so high that even only splitting each direction into 3 pieces leads to $3^6 = 729$ bins in total. Our typical simulations use about 100 particles per cell initially, and it is likely few phase space bins contain enough particles for merging with 3^6 bins. To avoid this problem, we only bin particles in the 3D velocity space and skip the merging if the 6D volume occupied by the selected particles is too large. This approach takes into account the spatial distribution of the selected particles, but also implies reducing the variance in the velocity space is more crucial than controlling the spatial location variance. Because all the particles are already in the same spatial cell, i.e., they can not be too far away from each other in the spatial space. Inside each velocity space bin, we choose 6 particles that are closest to each other for merging.

The particle merging algorithm needs calculating the distance in the 6D phase space, and it is defined as:

$$l_{6D} = \Delta l_{3D}/\Delta x + c_1 * \Delta v/v_{th} \tag{8}$$

$$v_{th} = \frac{1}{N_p}\sum_{i=0}^{N_p} |\mathbf{v_i} - \mathbf{v}|^2 \tag{9}$$

where Δl_{3D} is the spatial distance, Δv is the distance in the velocity space, Δx is the simulation cell size, N_p is the particle number of the corresponding cell, $\mathbf{v}$ it the bulk velocity, and v_{th} is the thermal velocity. The constant c_1 determines the relative importance of the spatial distance Δl_{3D} and the velocity space distance Δv. We choose $c_1 = 2$ based on our numerical experiment experience.

At the end of each computational cycle, the following scheme will be executed to select particles for merging if the ppc of a cell is larger than the merging threshold, which is 1.5 times of the initial ppc by default.

- Step-1: binning particles in the velocity space. For each spatial cell (Figure 4(a)), we create a grid in the velocity space ranging from $(v_x - v_{th}, v_y - v_{th}, v_z - v_{th})$ to $(v_x + v_{th}, v_y + v_{th}, v_z + v_{th})$, and assign particles to velocity bins (Figure 4(b)). The velocity grid is divided into $n_{bin} =$

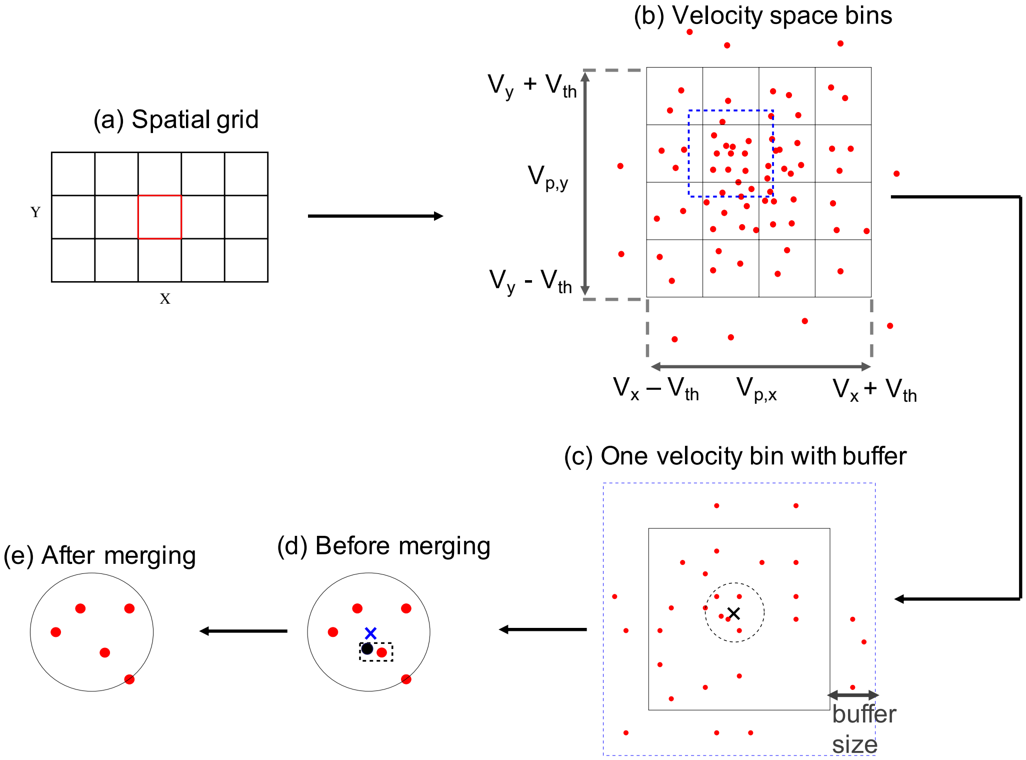
$\left\lceil 0.8 * N_p^{1/3} \right\rceil$ bins in each direction, where N_p is the current ppc of the cell and the constant 0.8 is chosen based on numerical experiments. We note that each bin contains a buffer region (Figure 4(c)), and the particles in the buffer region may also belong to other bins. We use 1/8 of the velocity space bin size as the width of the buffer region (Figure 4(c)). Due to the existence of the buffer region, one particle may belong to multiple bins, but it can only be selected for merging at most once during one cycle.

- Step-2: selecting particles from a bin. If there are more than 6 particles inside a bin, including the buffer region, we choose a cluster of 6 particles from them. For each velocity bin, we calculate the velocity center of the associated particles (black cross in Figure 4(c)), and find the 6 particles closest to the center in the 3D velocity space. If a particle in the buffer region has been selected for merging by a neighboring bin, this particle should not be selected again.

- Step-3: limiting the 6D distance. The previous step selects particles only based on the distance in the velocity space. This step ensures the selected particles are also close to each other in the 6D phase space. We find the 6D center (blue cross in Figure 4(d)) of these 6 particles, and the 6D distance l_{6D} of all the 6 particles to the center should be less than 0.6. Again, the constant 0.6 is chosen based on numerical experiments.

- Step-4: merging 6 particles into 5 with the algorithm described in section 4.2.1.

The particle selection method used in step-2 prefers selecting particles in the center of a bin. Without applying the buffer region in step-1, the particles near the edge of a bin are less likely chosen for merging. On the other hand, it is more likely that a bin contains more than 6 particles with a buffer region, and hence the merging efficiency is improved. Based on our numerical experiments, applying the buffer region does not improve the simulation results significantly, but it is still kept by default to avoid the aforementioned potential issues.



(a) Spatial grid
Y
X
(b) Velocity space bins
V_y + V_{th}
V_{p,y}
V_y - V_{th}
V_x – V_{th}
V_{p,x}
V_x + V_{th}
(c) One velocity bin with buffer
buffer size
(d) Before merging
(e) After merging

Figure 4: The steps of merging macro-particles.

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. The TP module 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, FLEKS becomes a pure test particle code, and the TP module can directly obtain the grid structure and electromagnetic fields from the MHD model. Compared to the embedded PIC simulations, the pure 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 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 reading the trajectory of a 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 particle for data analysis.

6. Numerical tests

6.1. Two-dimensional fast magnetosonic wave propagation with adaptive PIC region

We use a two-dimensional (2D) fast magnetosonic wave propagation test to demonstrate the capability of FLEKS's adaptive grids. The same initial

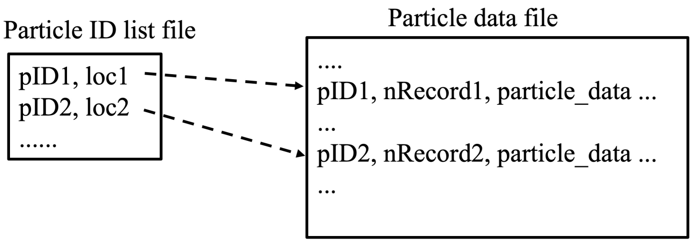


Figure 5: The file structure for storing test particles.

condition as what is described in [1] is applied here to produce a propagating fast magnetosonic wave. The simulation domain of the MHD code is $-160/3 < x < 160/3$ and $-40 < y < 40$. Two independent PIC domains are used. The left domain in Figure 6 covers the region of $-40 < x < 0$ and $-20 < y < 20$ with a grid resolution of $\Delta x = \Delta y = 1/16$. The right domain covers the region of $20 < x < 40$ and $-10 < y < 10$ with a grid resolution of $\Delta x = \Delta y = 1/8$. All cells of the right PIC domain are always switched on during the simulation. For the left domain, only the cells that satisfy the following conditions are switched on:

$$\begin{aligned} & r < \frac{L_x}{10} \\ & \text{or} \\ & r < \frac{L_x}{4} + \frac{L_x}{4} \cdot \frac{T \bmod 200}{200} \text{ and } r > \frac{L_x}{8} + \frac{L_x}{10} \cdot \frac{T \bmod 200}{200}, \end{aligned} \tag{10}$$

where r is the distance to the center of the PIC domain, L_x is the length of the PIC domain, which is 40 in this case, and T is the simulate time. The central PIC cells ($r < L_x/10$) are always switched on, and the outer shell of active PIC cells keep changing during the simulation. A movie that shows the adaptation of the active PIC region is provided as an online supplement. We note that the simulation parameters for these two PIC domains can be specified independently. For example, the cell size is different for these two PIC domains as it is described above, and the ion-electron mass ratio m_i/m_e

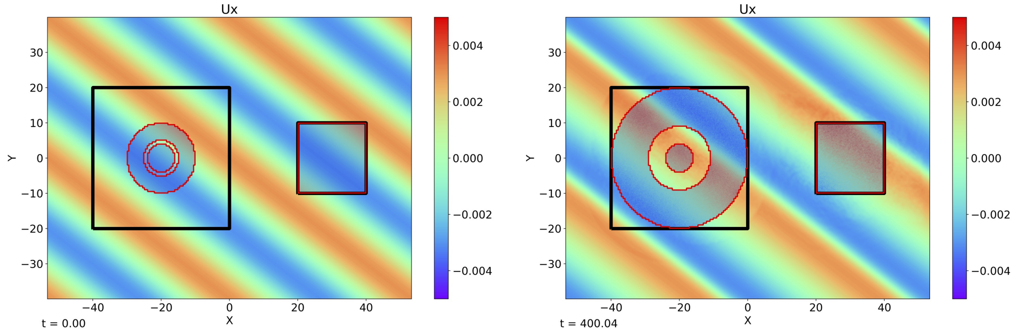


Figure 6: The velocity U_x of the 2D fast magnetosonic wave test at the beginning (left) and at t = 400 (right). The black rectangles show the area of the PIC grids. Inside each PIC grid, the semi-transparent area, which is enclosed by red lines, represents the active PIC region. Since all PIC cells are active for the right PIC grid, the black lines and the red lines are overlapped.

is 25 for the left domain and it is 100 for the right domain. Both PIC domains use CFL = 0.2, and 900 particles per cell (ppc) per species.

Figure 6 shows the plasma velocity U_x and the area of the active PIC cells at the beginning and at $t = 400$. The interface between the active PIC region and the MHD region is smooth, and there is not any significant artificial effect observed.

6.2. One-dimensional non-linear magnetosonic wave evolution

The evolution of the magnetosonic wave is non-linear. The wave may finally evolve into a shock, where the plasma phase space distributions may become non-Maxwellian. So the non-linear evolution of the magnetosonic wave simulation is suitable for testing the particle resampling algorithms.

In section 6.1, the wave vector is perpendicular to the background magnetic field direction. To make the particle phase space distribution further away from Maxwellian and hence more challenging for the particle resampling algorithms, we use a more general setting that the background magnetic field is neither perpendicular nor parallel to the wave vector in this 1D test. The initial condition of the 1D magnetosonic wave is:

$$\begin{aligned}
B_x(x) &= B_0\cos(\theta)\\
B_y(x) &= B_0\left[\sin(\theta)+\delta\sin(kx-\omega t)\right]\\
B_z(x) &= 0\\
u_x(x) &= \delta\sin(\theta)\frac{v_A^2 v_p}{v_p^2-v_s^2}\sin(kx-\omega t)\\
u_y(x) &= \delta\cos(\theta)\frac{v_A^2}{v_p}\sin(kx-\omega t)\\
u_z(x) &= 0\\
\rho(x) &= \rho_0\left[1+\delta\sin(\theta)\frac{v_A^2}{v_p^2-v_s^2}\sin(kx-\omega t)\right]\\
p(x) &= p_0\left[1+\gamma\delta\sin(\theta)\frac{v_A^2}{v_p^2-v_s^2}\sin(kx-\omega t)\right],
\end{aligned} \tag{11}$$

where γ is the specific heat ratio, $v_A = \frac{B_0}{\sqrt{\rho_0}}$ is the Alfven speed, $v_s = \sqrt{\frac{\gamma p_0}{\rho_0}}$ is the sound speed, and θ is the angle between the wave vector, which is the x-direction here, and the background magnetic field. The phase speed $v_p = \omega/k$ is:

$$v_p^2 = \frac{1}{2}\left(v_A^2 + v_s^2 + [(v_A^2+v_s^2)^2 - 4v_s^2 v_A^2 cos^2\theta]^{1/2}\right), \tag{12}$$

In this paper, we use $\gamma = 5/3$, $B_0 = 0.1$, $\theta = 30^\circ$, $\rho_0 = 1$, $p_0 = 1e-4$, $k = 2\pi/\lambda = 2\pi/64$, and $\delta = 0.5$. We note that the perturbation $\delta = 0.5$ is not small so that the initial conditions, which are obtained from linear theory, are not accurate. Since the goal of this test is to compare the simulation results with and without particle resampling, it is suitable and acceptable to use such a large perturbation.

The 1D simulation domain is $-32 < x < 32$ with a cell size $\Delta x = 0.05$. The initial number of particles per cell per species is 900, and CFL = 0.2. The simulation results at $t = 200$ are presented in Figure 7. To distinguish between the physical density and macro-particle number per cell, we use the word 'mass density' to represent the physical density, and 'number density' is the number of macro-particles per simulation cell or phase space bin. At $t = 200$, the wave already evolves into a non-linear state, and the velocity shows a sharp gradient near $x = 20$. The minimum and maximum number of ppc are about 500 and 3140, respectively, for the simulation without applying particle resampling. For the simulation with particle resampling, the minimum ppc is about 750 and the maximum ppc is about 1360, and these numbers are close to the splitting limit $0.8 * 900 = 720$ and the merging limit $1.5 * 900 = 1350$. It suggests the particle resampling algorithms are effective in controlling the particle number. Except for the particle number, the physical quantities of these two simulations are very similar to each other. The only noticeable different is that the electric field E_y of the simulation with particle resampling is more noisy near $x = 20$ due to the reduction of particle number. Figure 7(b) shows the ion phase space distribution for particles between $x = 21$ and $x = 21.2$. The two mass density distributions are comparable even though the particle number density is quite different.

Figure 8 shows the simulation speed, which represents the number of PIC cells that are updated per second per CPU core.For the first 700 cycles, both simulations become slower and slower due to the imbalance of the particle number per CPU core. Later, the minimum and maximum ppc reach the splitting and merging thresholds and the particle splitting algorithms start controlling the further change of the minimum and maximum ppc, so the simulation speed stops dropping for the simulation with particle resampling. At the end of the simulation, the simulation with particle resampling is almost twice faster than the one without particle resampling.

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

Magnetic reconnection is regarded as one of the most important physical processes for energy transfer between magnetic field and plasma in the space plasma environment, so it is also widely used to benchmark the performance of a kinetic plasma modeling code. Here, we use a two-dimensional (2D) asymmetric magnetic reconnection problem to test the particle resampling algorithms, because the particles distributions near the reconnection site can be non-Maxwellian. It is crucial to demonstrate the particle resampling algorithms preserve the non-Maxwellian distributions.

A double current sheet is used to initialize the simulation so that the whole system is symmetric and periodic boundary conditions can be applied in all directions. The simulation domain is $-64 < x < 64$ and $-16 < y < 16$. The background magnetic field is initialized as:

$$B_x(y) = \left(\frac{B_1 + B_2}{2}\right)\left[\tanh\left(\frac{y + 0.25L_y}{\delta}\right) - \tanh\left(\frac{y - 0.25L_y}{\delta}\right)\right] - B_2, \tag{13}$$

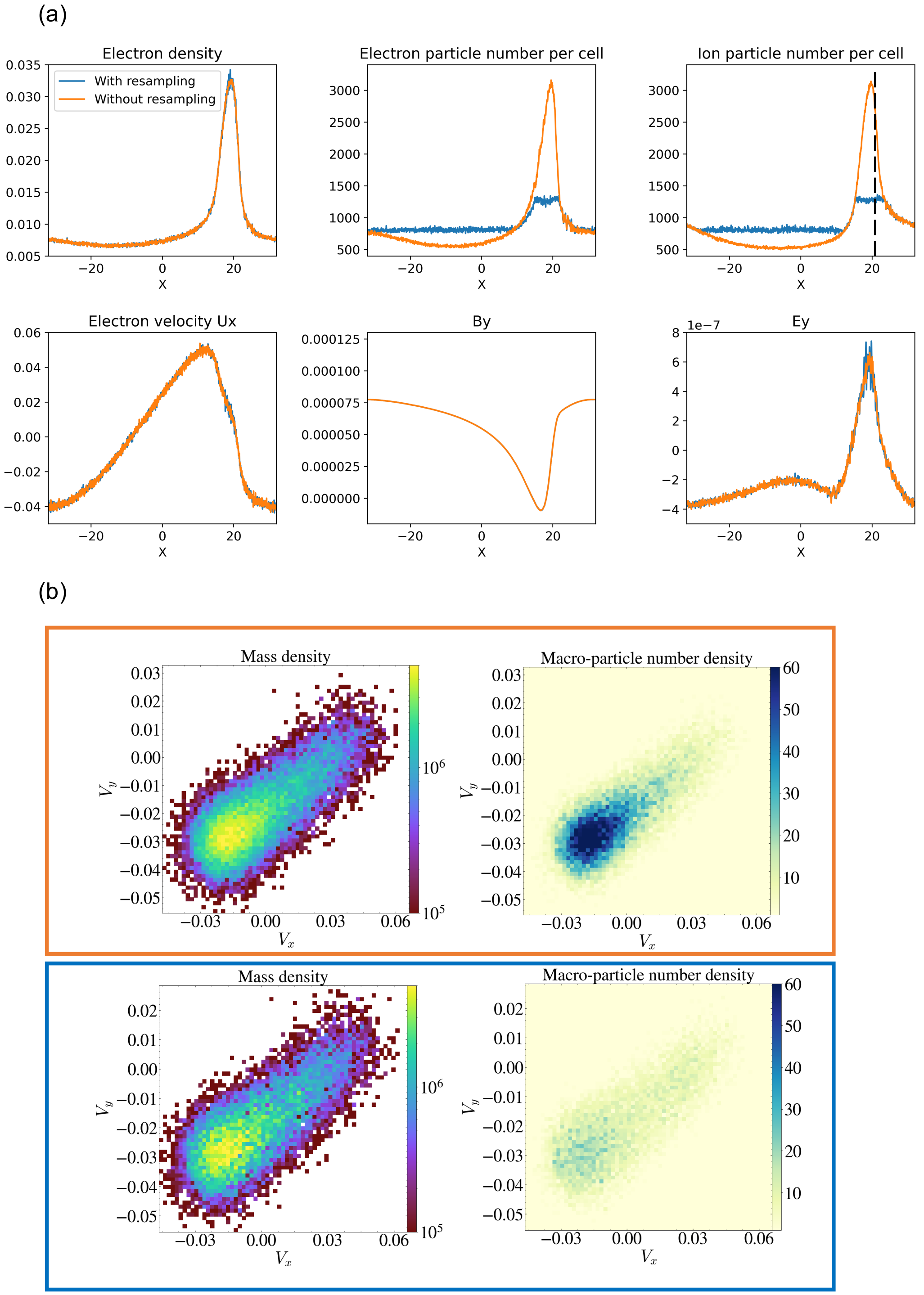


Figure 7: The 1D magnetosonic wave simulation results at $t = 200$. (b) shows phase space distributions at $x = 21$, where is marked with dashed black line in (a). The upper panel of (b) shows the results without particle resampling, and the lower panel shows the results with particle resampling.

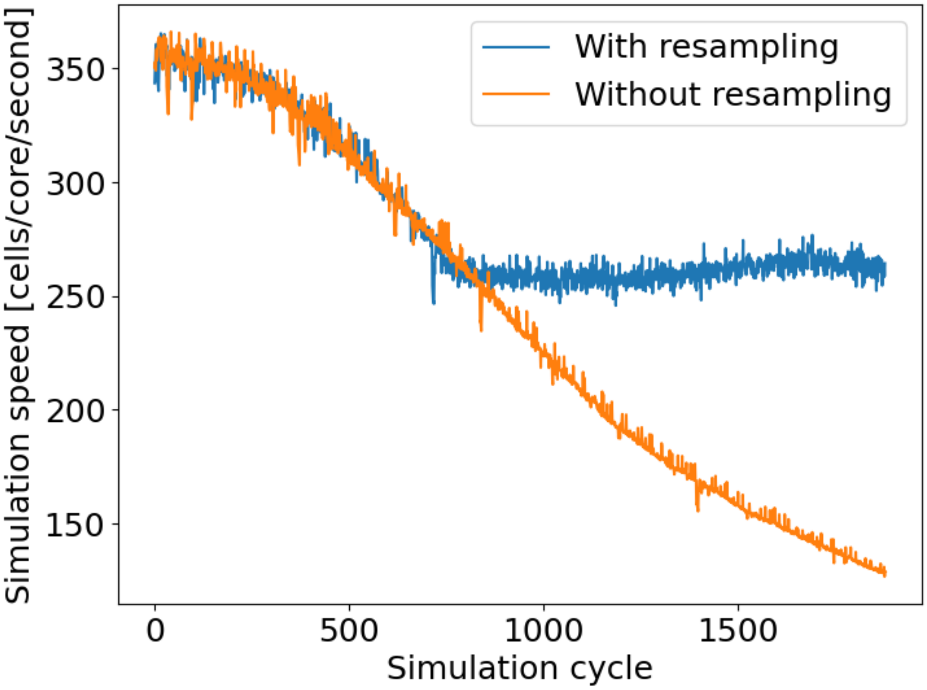


Figure 8: The simulation speed of the 1D magnetosonic wave simulations.

where $B_1 = 1$ and $B_2 = 2$ are the asymptotic magnetic field amplitudes. $L_y = 32$ is the width of the simulation domain, and the centers of the two current sheets are at $y = -8$ and $y = 8$, respectively. The plasma pressure is set to balance the magnetic field pressure p_B. To mimic the plasma environment of Earth's magnetopause, the asymptotic plasma beta $\beta = (p_i + p_e)/p_B$ are 3.6 and 0.15 on the "1" and "2" side, respectively. The initial pressure ratio between electrons and ions is $p_i/p_e = 5$ in the whole simulation domain. The ion temperature is:

$$T_i(y) = \left(\frac{T_{i,1} + T_{i,2}}{2}\right)\left[\tanh\left(\frac{y + 0.25L_y}{\delta}\right) - \tanh\left(\frac{y - 0.25L_y}{\delta}\right)\right] + T_{i,2}. \tag{14}$$

$T_{i,1} = 1.33$ and $T_{i,2} = 3.33$ are used in the simulation. With the pressure and temperature given above, the corresponding densities and ion inertial lengths are $n_1 = 1.127$, $n_2 = 0.0736$, $d_{i,1} = 0.942$ and $d_{i,2} = 3.69$. For all the simulations presented in this subsection, the grid resolution is $\Delta x = 1/16$, and the CFL is 0.4.

Figure 9 and Figure 10 show the fields near the reconnection site at $t = 20$ with 100 and 400 initial ppc, respectively. The left (right) columns of Figure 9 and 10 are the results without (with) applying particle resampling. Due to the magnetic reconnection plasma flow, the electron ppc around the current sheet increases to about 250 (950), and the minimum ppc in the inflow region reduces to less than 50 (200) in Figure 9 (Figure 10) without applying the particle resampling algorithms. After applying the particle resampling algorithms, the electron ppc becomes more uniform in the whole domain. With the threshold parameters described in section 4, the particle splitting (merging) threshold ppc is 80 (150) and 320 (600) for the simulations with the initial ppc of 100 and 400, respectively. The minimum electron ppc in the right column of Figure 9 (Figure 10) is about 83 (325), and the maximum ppc is about 190 (630). The minimum ppc in the simulate is just a few particles more than the splitting threshold since the splitting algorithm is effective in generating new particles. The difference between the maximum ppc and the merging threshold is larger, but the maximum ppc is still much smaller than that in the simulation without applying particle resampling.

Figure 9 and Figure 10 also compare the physical quantities of the simulations. All simulations show essentially the same structures, including the off-diagonal electron tensor. It demonstrates the particle resampling algorithms do not introduce any significant artificial effect.

Figure 11 shows the electron phase space distributions from three sampling locations near the reconnection site. These three sampling locations are marked with black rectangles in the first row of Figure 9 and Figure 10. From top to bottom, we label these three sampling boxes as box-A, box-B and box-C. In Figure 11, row (a) and (b) show distributions from box-A, row (c) and (d) show distributions from box-B, and row (e) and (f) show distributions from box-C. Each column shows the distributions from the same simulation, and the simulation parameters, i.e., the initial ppc and turning on/off the particle resampling algorithms, are described at the top of Figure 11. Row (a), (c) and (e) show the density distributions, and row (b), (d) and (f) show the macro-particle number distributions in phase space. Figure 11 demonstrates the particle resampling algorithms preserve the phase space distributions well. The particle resampling does not change the particle number too much at the sampling location box-A (row (b)), and the 'U' shape density distribution is well preserved (row (a)). From row (d) and row (f), it is clear that the particle resampling significantly reduces the particle number around the distribution centers, so the centers of the density distribution (row (c) and row (e)) with particle resampling are more noisy. But the density distribution structure, which consist of a core and a crescent distribution, is still clearly preserved in row (c) and also in (e1) and (e2). The distributions (e3) and (e4) are also very similar to each other.

6.4. Strong and weak scalings

3D asymmetric magnetic reconnection simulations are used to test the strong and weak scaling of FLEKS. The setup of the 3D test is similar to the 2D simulation in previous subsection, and it is uniform in the z-direction. Since FLEKS uses the parallel field and particle data structures provided by AMReX, The scaling results largely depend on the performance of AMReX [9, 10]. Figure 12 shows the weak scalings. With 8^3 cells per core, the performance is still good with up to about 10k cores. With 16^3 cells per CPU core, it reaches good performance even with 28672 cores. Figure 13 shows the strong scalings of two problems. The speedup is not too far away from the ideal scaling up to about 7k (Figure 13(a)) or 14k (Figure 13(b)) CPUs for these problems.

6.5. Magnetospheric simulations

Magnetospheric simulation is the most important application of MHD-AEPIC. Here, we show examples of how FLEKS benefits magnetosphere

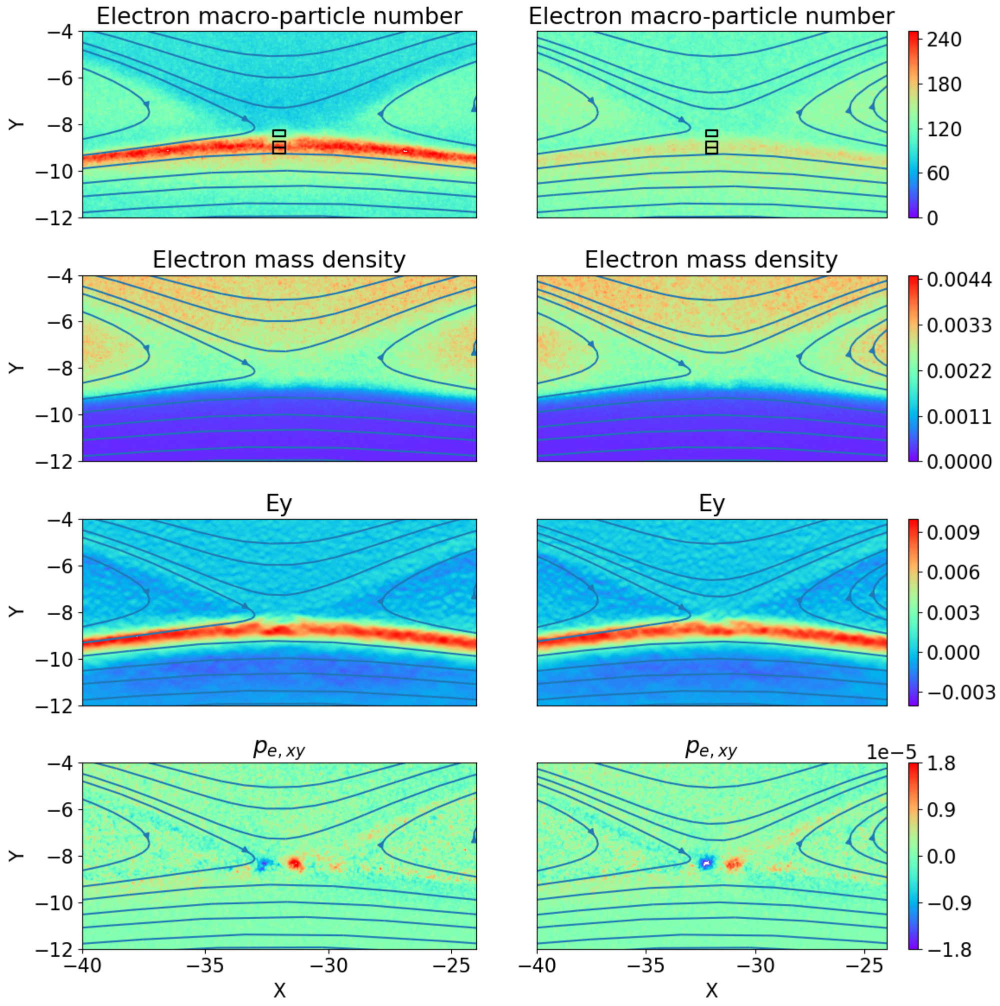


Figure 9: 2D magnetic reconnection results with (right column) or without (left column) particle resampling. The initial particle number per cell is 100.

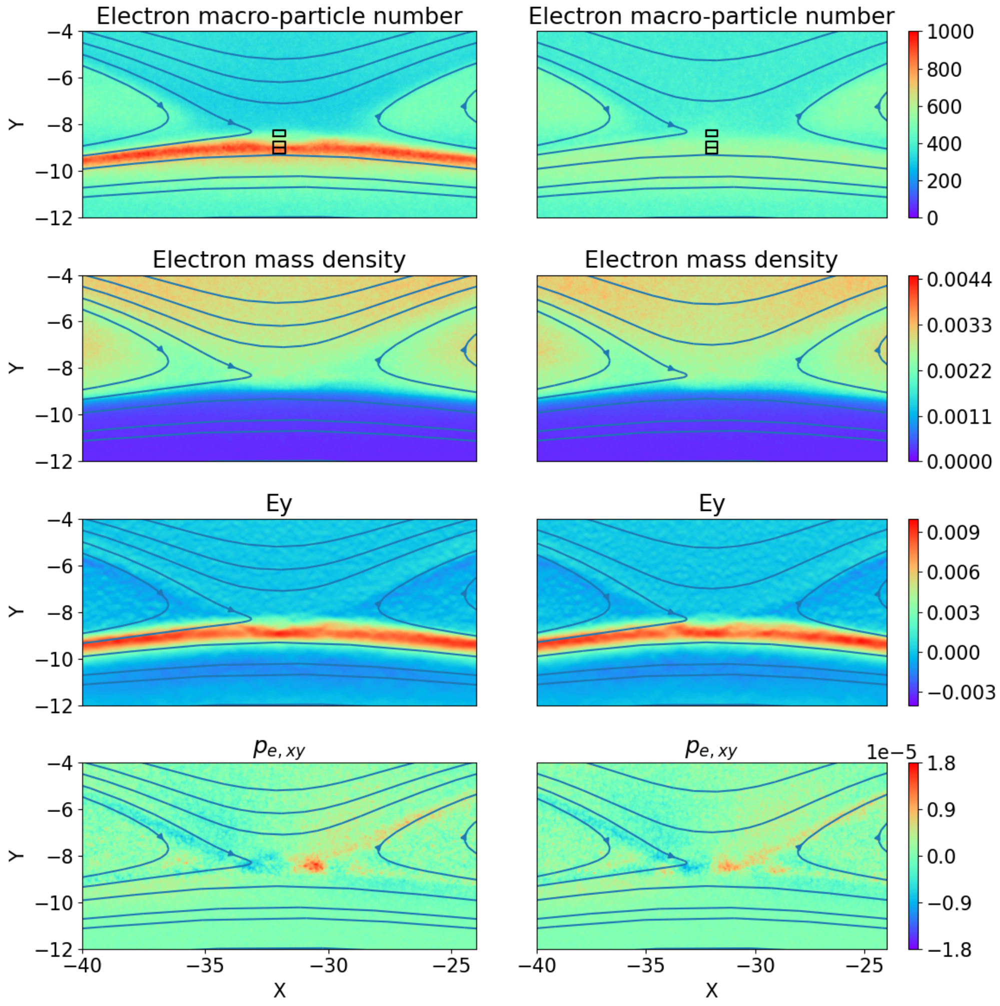


Figure 10: 2D magnetic reconnection results with (right column) or without (left column) particle resampling. The initial particle number per cell is 400.

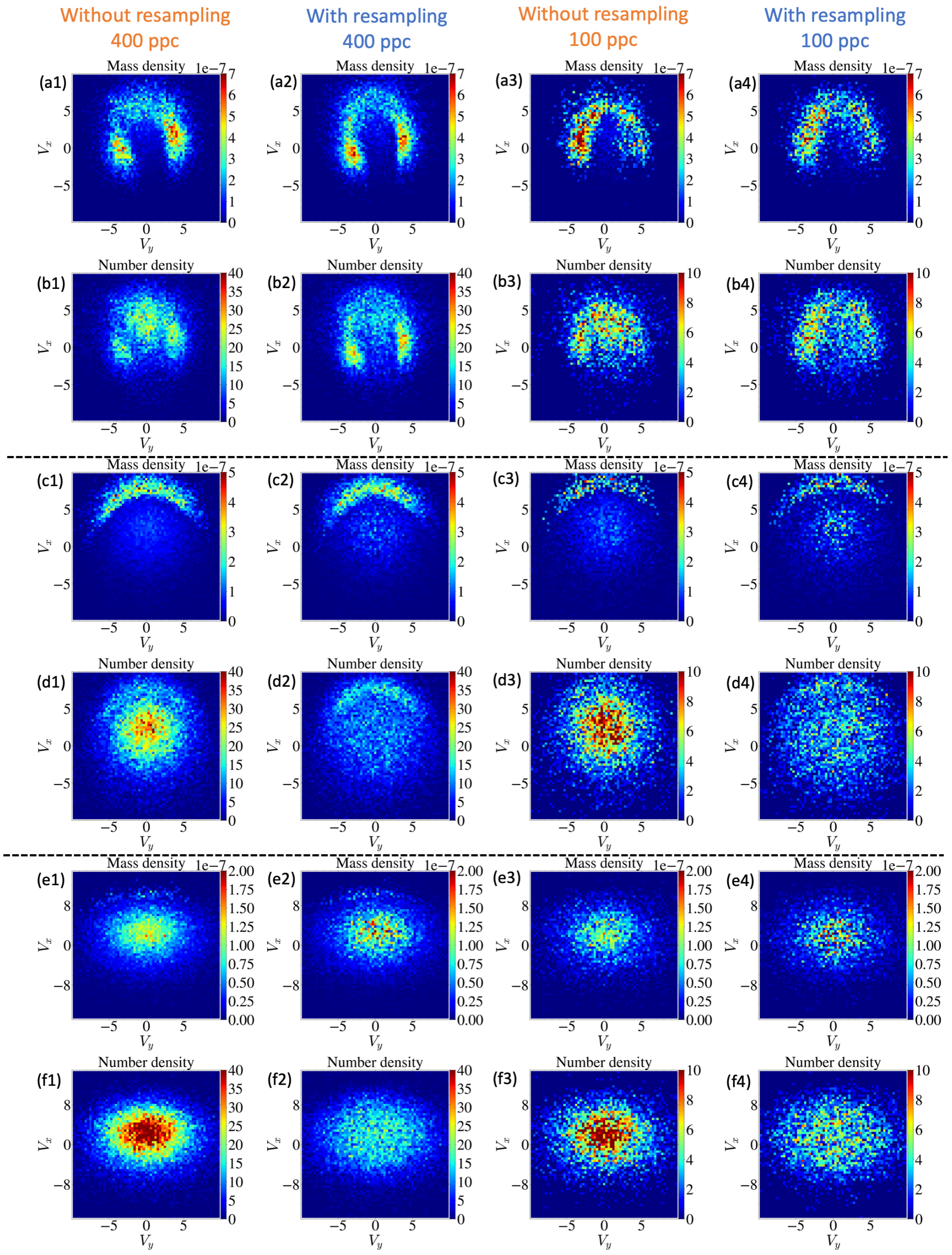


Figure 11: Each column shows the phase space distributions from the same simulations. The first two rows, middle two rows and the last two rows represent the distributions of box-A, box-B and box-C, respectively. From top to bottom, the black rectangles in Figure9 and Figure 10 show the locations of box-A, box-B and box-C. Row (a), (c) and (e) are physical density distributions. Row (b), (d) and (f) show particle number per phase space bin.

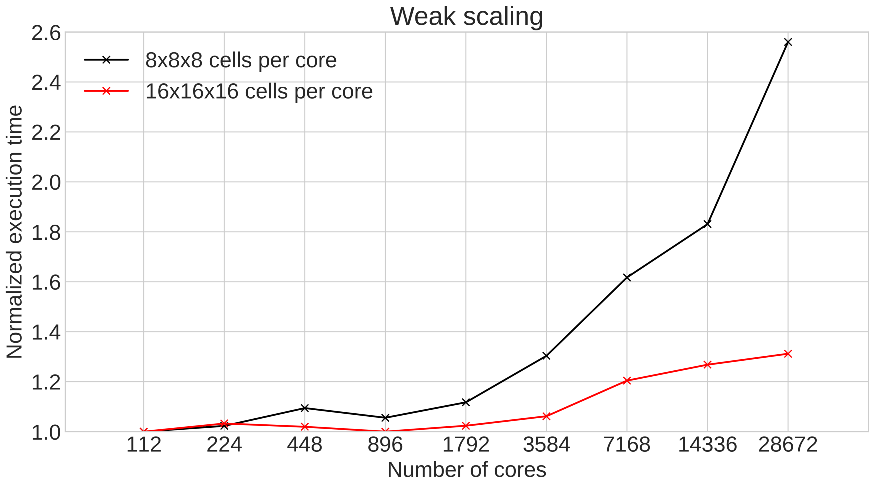


Figure 12: The weak scaling results of FLEKS. The execution time with 112 CPU cores is used as the reference. If the scaling is perfect, the normalized execution time should be 1.0 for all cases.

modeling. The global MHD magnetosphere model uses the same setup as the simulations presented in [5, 24], but the active PIC region is not limited to be a box anymore. In the example of Figure 14, the active PIC region can cover the dayside magnetopause, including the dawn-side and dusk-side flanks, and also the cusp region at the same time.

The test particle module enables us to follow the trajectories of particles in the magnetosphere. Figure 15 shows an example of test particles in Earth's magnetosphere.

7. Conclusion

In this paper, we introduce a new kinetic code FLEKS, which is designed as the kinetic component of the MHD-AEPIC model [8, 25]. To support long simulations with varying global configurations, FLEKS allows activating or deactivating cells dynamically during a simulation to fit the regions of interest. This feature is introduced by Shou et al. [8] first, but FLEKS is more flexible that the minimum activation unit is a patch contains N ($N \geq 2$) cells in each direction instead of a large block, and FLEKS supports multiple independent PIC domains in an MHD-AEPIC simulation. During

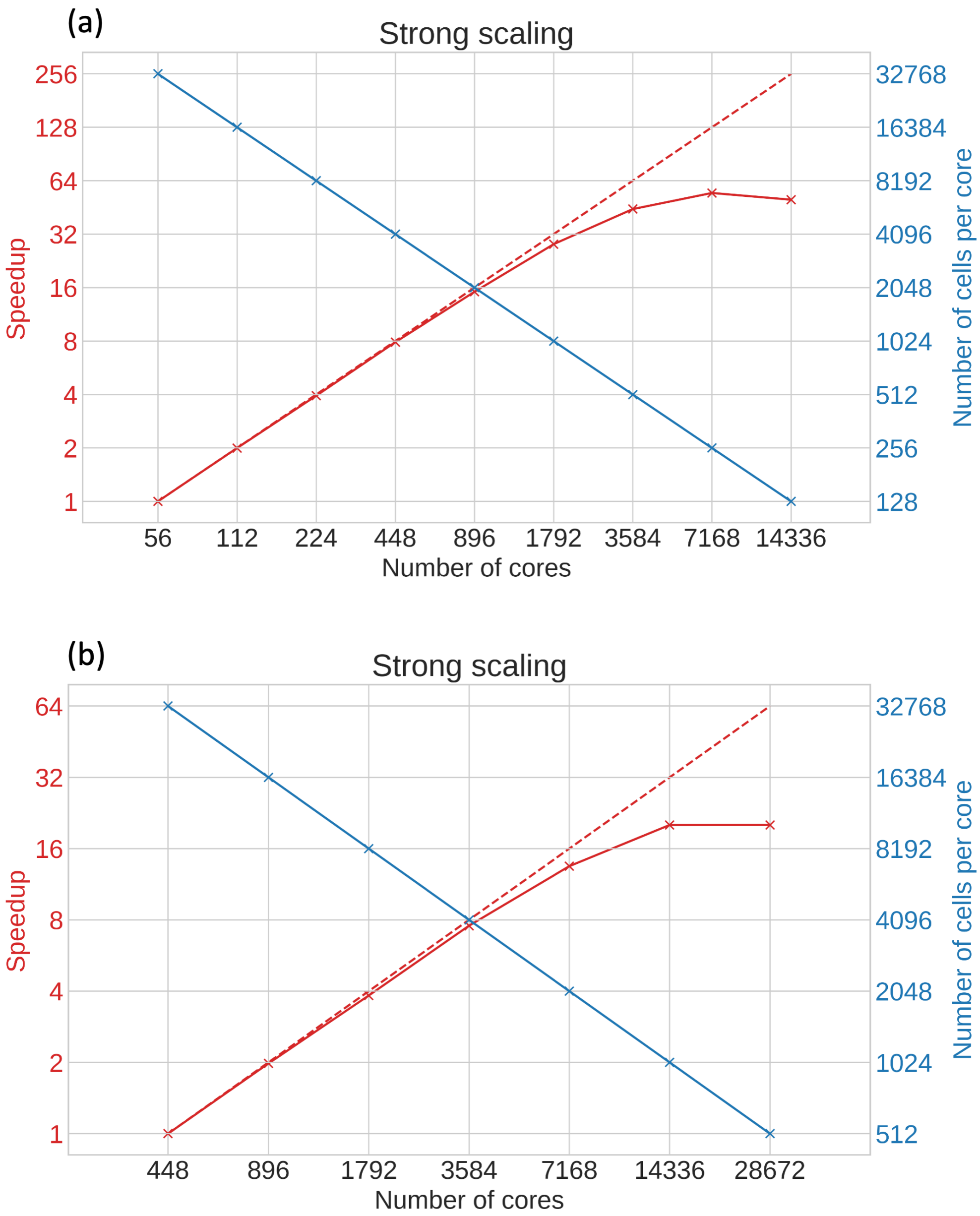


Figure 13: The strong scaling of FLEKS. (a) and (b) show the scalings of simulations with 1.835 million and 14.68 million cells, respectively. The red solid lines represent speedup, and the red dashed lines are perfect speedup. The blue lines show the number of PIC cells per CPU core.

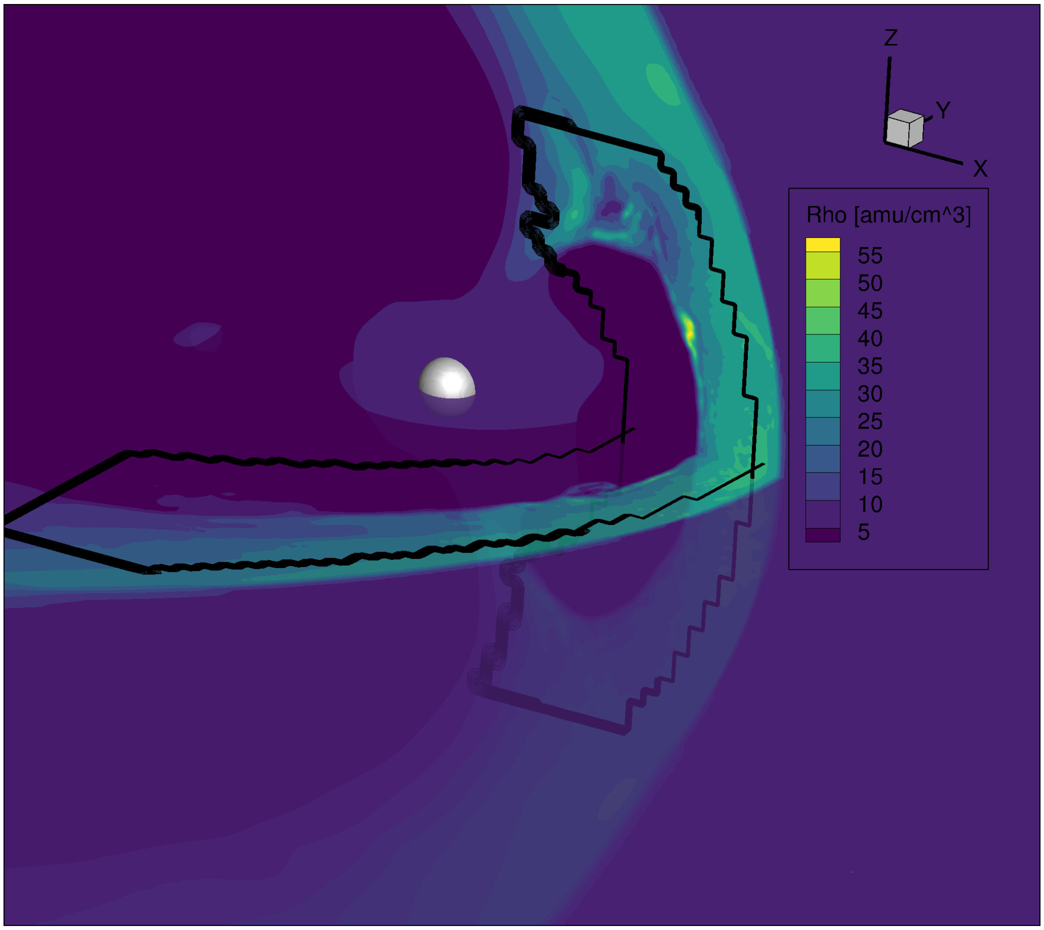


Figure 14: An MHD-AEPIC simulation of Earth's magnetosphere with the dayside magnetopause and the cusps covered by FLEKS. The black lines indicate the edge of the active PIC region.

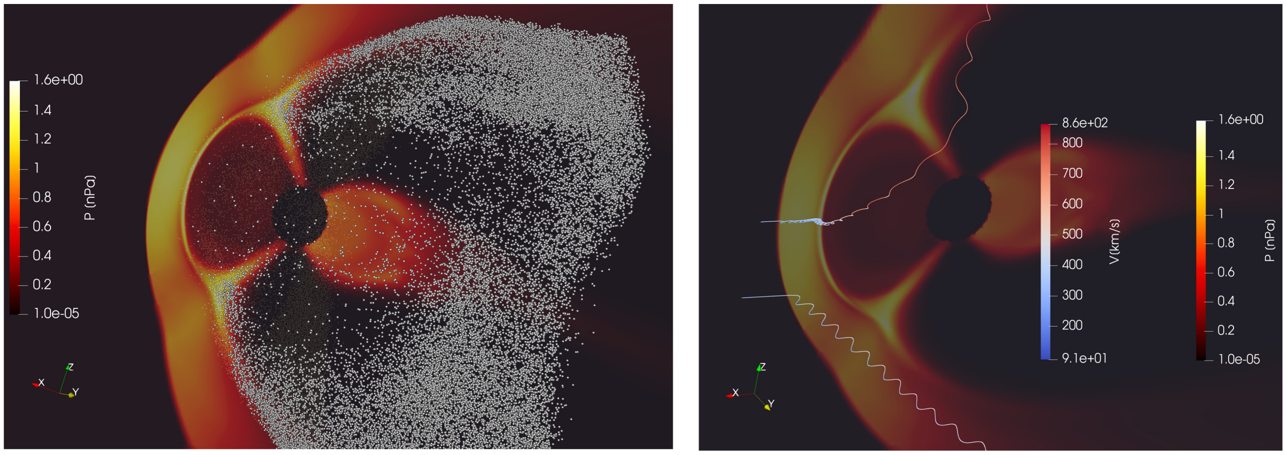


Figure 15: The locations (left) and the example trajectories (right) of test particles.

a long simulation, since the plasma properties inside the active PIC region may change greatly, we design an adaptive time-stepping algorithm to adjust PIC time step accordingly. The adaptive time-stepping scheme preserves the energy conservation property of the ECSIM algorithm.

Since the number of particle per cell may change dramatically during a long simulation and leads to load imbalance and loose of accuracy in the cells with fewer particles, a particle splitting and a particle merging algorithms are designed to control the change of ppc. The particle merging algorithm selects 6 particles that are close to each other in the phase space and combines them into 5 new particles. The merging conserves the total mass, momentum and energy, and it also preserves the phase space structure as much as possible by inheriting velocities from the old particles.

The test-particle module expands the capability of FLEKS, and provides a useful tool for investigating the transport and energization of particles in magnetospheres.

Acknowledgments: This work was primarily supported the NSF PREEVENTS grant 1663800. Computational resources supporting this work were provided by an NSF LRAC allocation at the Texas Advanced Computing Center

(TACC) at The University of Texas at Austin.

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 simulation of ganymede's magnetosphere, J. Geophys. Res. 121 (2016). `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 Science (2020). `doi: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) 21`doi:10.1002/essoar.10508044.1`.
URL `https://doi.org/10.1002/essoar.10508044.1`

[26] G. Lapenta, J. U. Brackbill, Nonlinear evolution of the lower hybrid drift instability: Current sheet thinning and kinking, Phys. Plasmas 9 (2002) 1544.