Particle In Cell (PIC) simulation

BY YOUJUN HU

Email: yjhu@ipp.cas.cn
Institute of Plasma Physics, Chinese Academy of Sciences

Abstract

This note reviews the basic theory of Particle-In-Cell (PIC) simulation of plasmas.

1 Particle methods

There are two perspectives of interpreting the particle-in-cell (PIC) method. The first one is simple and intuitive but applies only in restricted cases (such as full-f and full-orbits simulations). In this interpretation, a marker in PIC simulation is considered as a super-particle which contains many physical particles of (nearly) the same velocities and locations. The source terms in the Maxell equations are obtained by gathering all the super-particles in a cell to get the various (cell-averaged) moments such as the charge density and current density. The cell-averaging has the effect of smoothing the self-consistent electromagnetic field, and thus reducing or removing the collision effects between physical particles.

The second perspective of interpreting PIC method is more mathematical and applies to more general cases (such as delta-f and gyrokinetic simulations). In this interpretation, the PIC methods make use of the characteristic lines of hyperbolic Partial Differential Equations (PDEs), e.g. the Vlasov equation and gyrokinetic equation. These methods reduce a hyperbolic PDE to a family of ordinary differential equations (ODEs) that can be easily integrated from initial values. Assembling the solutions from all the ODEs then give the solution to the original PDE.

By using the integration along the characteristic lines, we avoid directly discretising the partial derivatives of the unknown distribution function with respect to phase-space coordinates, and thus we do not need a phase-space mesh to construct numerical approximation to the

phase-space partial differential operators. This enables us to adopt random sampling of the phase-space, which has the advantage of reducing the error in evaluating high-dimensional phase-space integration. In this sense, this method can be considered as a kind of "mesh-free" method.

Since the characteristic lines are usually identical to the orbits of particles/elements in the phase-space, these methods are generally called "particle methods". In the core of these methods are the method of characteristics and Monte-Carlo sampling and integration.

The computational nodes (often called markers/particles/super-particles) in particle methods follow the particle orbits and thus their positions in phase-space evolve with time, which is one of the differences from the usual Euler-grid-based methods that usually uses fixed grid-points. The evolving computational nodes in particle method are often called Lagrangian markers while the fixed grids are often called Euler grid-points. Therefore, particle method is a kind of **mesh-free Lagrangian** method.

The so-called Particle-In-Cell (PIC) method, however, is a kind of **incomplete** particle method, in which both evolving nodes (mesh-free) and fixed grid-points (mesh) are used. Specificly, the kinetic equation for the particle distribution function in the phase space is solved by using mesh-free particle methods, whereas Maxwell's eqaution for the electromagnetic field are solved by using spatial mesh. To obtain the source terms in Maxwell's equation at grid-points, we need to calculate moments of the distribution function represented by the mesh-free computational nodes. In PIC simulations, the values of a moment at a spatial grid-point is approximated by the averaged value of the moment in the corresponding spatial cell. [This averaging procedure, along with the use of spatial shape function of markers (discussed later), has the effect of reducing (ideally removing) collisions between particles. Collisions, if to be modeled, should be modeled by other means.] Therefore, the PIC method is a kind of **hybrid particle-mesh method**.

This note discusses the basic theory of the Particle-In-Cell (PIC) simulation, along with practical implementation in curvilinear coordinate system. In the process of learning the PIC simulation method, I have developed several toy Fortran codes to test what I have learned. The numerical results given in these notes are obtained by using these codes. A copy of these codes can be found at the following URL:

http://theory.ipp.ac.cn/~yj/codes/pic_1D_src.tar

1.1 Brief history of particle-mesh methods

Particle-mesh (PM) method was invented in the 1950's at LANL for simulation of compressible fluid flows. The first wide-spread application was for collisionless plasma simulation, for which particle-mesh method was reinvented in the 1960's and called particle in cell methods. These methods later also obtained popularity in cosmology simulations.

2 Phase-space sampling

Particle simulations in space plasma physics community usually use the Cartesian coordinate system, which is suitable for the simple configuration considered there. In tokamak physics community, curvlinear coordinates are usually adopted because of the toroidal configuration. I will primarily use general curvilinear coordinate system in presenting the PIC method (since my research interests are in tokamak plasma). Specificly, the Jacobian of the coordinate system will be explicitly shown in the formulas. (Explicitly showing the Jcobian is helpful, considering that many bugs/confusions in practice come from the Jacobian being not correctly handled.)

2.1 Phase sapce coordinates

Suppose that the 6D phase space $\mathbf{Z} = (\mathbf{r},\ \mathbf{v})$ is described by general curvlinear coordinates $(\alpha,\ \beta,\ \gamma,\ v_\alpha,\ v_\beta,\ v_\gamma)$. For notational convenience, define

$$d\mathbf{Z} = d\mathbf{r}d\mathbf{v}, \tag{1}$$

$$d\mathbf{r} = d\alpha d\beta d\gamma, \tag{2}$$

$$d\mathbf{v} = dv_\alpha v_\beta v_\gamma, \tag{3}$$

then the differential volume element in the phase space is written as

$$\begin{aligned} d\Gamma &\equiv |\mathcal{J}_r\mathcal{J}_v| d\alpha d\beta d\gamma dv_\alpha dv_\beta dv_\gamma \qquad (4)\\ &= |\mathcal{J}_r\mathcal{J}_v| d\mathbf{r}d\mathbf{v} \\ &= |\mathcal{J}_r\mathcal{J}_v| d\mathbf{Z} \qquad (5) \end{aligned}$$

where $\mathcal{J}_r$ is the Jacobian of the transformation from Cartesian space coordinates (x, y, z) to curvlinear coordinates (α, β, γ), and $\mathcal{J}_v$ is the Jacobian of transformation from Cartesian velocity coordinates (v_x, v_y, v_z) to curvlinear coordinates (v_α, v_β, v_γ). (Here I assume that the spatial Jacobaian and velocity Jacobian are separable. So the total Jacobian is written as the product of them) Most authors use the notations that $d\mathbf{r} = |\mathcal{J}_r| d\alpha d\beta d\gamma$ and $d\mathbf{v} = |\mathcal{J}_v| dv_\alpha dv_\beta d\gamma$, i.e., $d\mathbf{r}$ and $d\mathbf{v}$ refer to volume elements rather than $d\alpha d\beta d\gamma$ and $dv_\alpha v_\beta v_\gamma$. However, $d\mathbf{r}$ and $d\mathbf{v}$ defined by Eqs. (2) and (3) refer to $d\alpha d\beta d\gamma$ and $dv_\alpha v_\beta v_\gamma$, respectively. These seem to be more uniform in syntax. I will stick to this notation in this note.

2.2 Phase-space sampling

Given a probability density function $P = P(\alpha, \beta, \gamma, v_\alpha, v_\beta, v_\gamma)$ that satisfies the following normalization condition

$$\int_\Omega P d\Gamma = 1, \tag{6}$$

where Ω is the phase space region of interest. Using expression (4), the normalization condition (6) is written as

$$\int_\Omega P \, |\mathcal{J}_r \mathcal{J}_v| d\alpha d\beta d\gamma dv_\alpha dv_\beta dv_\gamma = 1. \tag{7}$$

We use the probability density function P to sample the phase space region Ω with N markers (It is $P' = P \, |\mathcal{J}_r \mathcal{J}_v|$ that is directly used in the rejection method, which is discussed in Sec. 7.). Then the marker distribution function $\tilde{g}$ is given by

$$\tilde{g}(\mathbf{Z}) = \frac{1}{|\mathcal{J}_r \mathcal{J}_v|} \sum_{j=1}^{N} \delta(\alpha - \alpha_j)\delta(\beta - \beta_j)\delta(\gamma - \gamma_j)\delta(v_\alpha - v_{\alpha j})\delta(v_\beta - v_{\beta j})\delta(v_\gamma - v_{\gamma j}), \tag{8}$$

where δ is the Dirac delta function, and (α_j, β_j, γ_j, $v_{\alpha j}$, $v_{\beta j}$, $v_{\gamma j}$) are random numbers generated by the rejection method.

Denote the continuous limit (i.e., $N \to \infty$) of $\tilde{g}$ by g. Then g is given by

$$g = N \times P. \tag{9}$$

2.3 Monte-Carlo estimation of an integral

Next, consider an integral in a subspace of Ω (denoted by Ω'):

$$\int_{\Omega'} A(\mathbf{Z}) f(\mathbf{Z}) d\Gamma, \tag{10}$$

where $f(\mathbf{Z})$ is a distribution function and $A(\mathbf{Z})$ is a known function of $\mathbf{Z}$. Using the marker distribution g and $\tilde{g}$, the above integral can be written as

$$\begin{aligned}\int_{\Omega'} A(\mathbf{Z}) f(\mathbf{Z}) d\Gamma &= \int_{\Omega'} A(\mathbf{Z}) \frac{f(\mathbf{Z})}{g(\mathbf{Z})} g(\mathbf{Z})\, d\Gamma \\ &\approx \int_{\Omega'} A(\mathbf{Z}) \frac{f(\mathbf{Z})}{g(\mathbf{Z})} \tilde{g}(\mathbf{Z}) d\Gamma \\ &= \int_{\Omega'} A(\mathbf{Z}) \frac{f(\mathbf{Z})}{g(\mathbf{Z})} \frac{1}{|\mathcal{J}_r \mathcal{J}_v|} \sum_{j=1}^{N} \delta(\alpha-\alpha_j)\delta(\beta-\beta_j)\gamma(\gamma-\gamma_j) \\ &\times \ \delta(v_\alpha - v_{\alpha j})\delta(v_\beta - v_{\beta j})\delta(v_\gamma - v_{\gamma j})|\mathcal{J}_r \mathcal{J}_v| d\alpha d\beta d\gamma dv_\alpha dv_\beta dv_\gamma \\ &= \sum_{j=1}^{N'} A(\mathbf{Z}_j) \frac{f(\mathbf{Z}_j)}{g(\mathbf{Z}_j)}. \end{aligned} \tag{11}$$

where N' is the number of markers that are located within the subspace Ω'. Equation (11) is the Monte-Carlo estimation of the integral (10).

2.4 Phase space volume sampled by a marker

The Monte-Carlo estimation (11) can be understood in terms of phase space volume sampled by a marker. The definition of g implies that the number of markers within a small volume of phase-space $d\Gamma$ is given by

$$dN = g d\Gamma. \tag{12}$$

Therefore the average phase space volume occupied (or sampled) by a marker is given by

$$V_p \equiv \frac{d\Gamma}{dN} = \frac{1}{g}. \tag{13}$$

Then, using the definition of the volume integral, we obtain

$$\int_{\Omega'} A(\mathbf{Z}) f(\mathbf{Z}) d\Gamma \approx \sum_{j=1}^{N'} A(\mathbf{Z}_j) f(\mathbf{Z}_j) V_{pj} = \sum_{j=1}^{N'} A(\mathbf{Z}_j) \frac{f(\mathbf{Z}_j)}{g(\mathbf{Z}_j)}, \tag{14}$$

which agrees with Eq. (11).

2.5 Effective probability density used in loading markers

In numerical implementation, markers are loaded in terms of coordinates $(\alpha, \beta, \gamma, v_\alpha, v_\beta, v_\gamma)$, i.e., these variables are directly sampled. Since the probability of markes being in coordinate interval $d\alpha d\beta d\gamma dv_\alpha dv_\beta dv_\gamma$

is given by $P|\mathcal{J}_r\mathcal{J}_v|d\alpha d\beta d\gamma dv_\alpha dv_\beta dv_\gamma$, the effective probability density function in terms of these coordinates is given by

$$P' = P|\mathcal{J}_r\mathcal{J}_v|. \tag{15}$$

It is P' (rather than P) that is directly used in loading markers. (The methods of generating random numbers satisfying a given probability density function are discussed in Sec. 7.) Note that Jacobian is not uniform in space for a general curvilinear coordinate system. Therefore equal intervals in coordinates do not corresond to equal volume elements at different locations. For a uniform probability density P and in a fixed interval $d\alpha d\beta d\gamma dv_\alpha dv_\beta dv_\gamma$, more markers are loaded for locations where $|\mathcal{J}_r\mathcal{J}_v|$ is larger.

2.6 Time evolution of phase-space volume occupied by a marker

Evaluate V_p at the initial location of each marker and then assign this value, V_{pj0}, to the corresponding marker, i.e.,

$$V_{pj0} = \frac{1}{g(\mathbf{r}_{j0}, \mathbf{v}_{j0})}, \tag{16}$$

where $\mathbf{r}_{j0}$ and $\mathbf{v}_{j0}$ are the initial coordinates of the jth markers.

Further note that any particle distribution function g in any non-relativistic continuous electromagnetic field evolves as $dg/dt = 0$, where d/dt is the convective derivative in the phase space, i.e., the distribution function is constant along a characteristic line. Since $V_p = 1/g$, it follows that,

$$\frac{d}{dt}(V_{pj}) = 0, \tag{17}$$

Therefore, V_{pj} associated with a marker at later time is always equal to the volume initially assigned to it, i.e., $V_{pj} = V_{pj0}$. This is assumed in most collisionless PIC simulation codes and related to the issue of discrete particle noise (in time). In practical simulations, we use finite number (rather than infinite number) of markers to represent the marker distribution function g. Due to this reason, $d\tilde{g}/dt$ is never exactly zero. This is the so-called discrete particle effect. We can examine how accurate $d\tilde{g}/dt \approx 0$ is by numerically count the number of markers ΔN in a small phase volume ΔV around a marker and examining the evolution of $\Delta N/\Delta V$. Some choices of the initial state $g(\mathbf{x}, \mathbf{v}, t = 0)$ can make $d(\Delta N/\Delta V)/dt$ smaller and thus reduce the noise introduced by the discrete particle effect.

2.7 Weights of markers

Denote the physical particle distribution function in question by f (f can be the full-f or δf, depending on the context). In this note, I define the weight of a marker as the number of physical particles carried by f in the corresponding phase-space volume sampled by the marker, i.e.,

$$w_j = f(\mathbf{Z}_j) V_{pj}, \tag{18}$$

which, by using Eq. (13), is further written as

$$w_j = \frac{f(\mathbf{Z}_j)}{g(\mathbf{Z}_j)}. \tag{19}$$

In most PIC codes I wrote, I implement several kinds of marker distribution functions, which can be chosen via an option. In practice, I often use the marker distribution g that is uniform in real space and Maxwellian in velocity space with a constant temperature:

$$g = \frac{N_p}{V}\left(\frac{m}{2\pi T_g}\right)^{3/2} \exp\left(-\frac{m v^2}{2T_g}\right), \tag{20}$$

where T_g is a constant temperature, N_p is the number of markers, and V is the spatial volume of the computational box. [This marker distribution satisfies the normalization:

$$\int g d\Gamma = N_p. \tag{21}$$

]

2.7.1 Various definitions of weights

In the literature on δf PIC simulations, the weight defined by different authors may differ by a factor and this factor is taken into account when calculating moments using the weight.

2.7.2 In Ben's thesis

For example, in Ben's thesis[?], the weight is defined as $w_\star = \delta f \,/\, f_0$, where f_0 is a Maxwellian distribution with constant temperature T_g and constant density n_0. Then the marker distribution g given in Eq. (20) is related to f_0 as $g = N_p f_0 / (n_0 V)$. Then $w_{j\star}$ and the weight w_j defined above is related by

$$w_j = \frac{\delta f}{g} = \frac{\delta f}{f_0 N_p} n_0 V = w_{j\star} \frac{V}{N_p} n_0. \tag{22}$$

2.7.3 In GEM

In the GEM code, the weight is defined by $w_\star = \delta f \,/\, g_\star$ and $g_\star$ is chosen as

$$g_\star = \left(\frac{m}{2\pi T_g}\right)^{3/2} \exp\left(-\frac{m v^2}{2T_g}\right), \tag{23}$$

which is related to g defined in Eq. (20) by $g = g_\star N_p / V$. Here $g_\star$ satisfies the normalization $\int g_\star \, d\Gamma = V$. Then the relation of $w_{\star j}$ in GEM and w_j in this note is given by

$$w_j = \frac{\delta f}{g} = \frac{\delta f}{g_\star N_p / V} = w_{j\star} \frac{V}{N_p}, \tag{24}$$

Note that $w_{j\star}$ defined here is of dimension of number density. Actually used in the code is the normalized weight defined by $\overline{w}_{j\star} = w_{j\star} \,/\, n_{a0}$, where n_{a0} is the volume averaged number density of a species (denoted by cn0 in GEM). Then the relation between w_j and $\overline{w}_{j\star}$ is given by

$$\overline{w}_{j\star} = w_j \frac{N_p}{n_{a0} V}, = w_j \frac{N_p}{(n_{a0} x_u^3)\bar{V}}, \tag{25}$$

where $\bar{V} = V \,/\, x_u^3$ and x_u is the length unit used in GEM. Assume that the equilibrium density and temperature are approximately equal to n_{a0} and T_g, respectively, then $\overline{w}_{j\star} \approx \delta f \,/\, f_0$, where f_0 is the equilibrium distribution function. The benefit of defining the weight this way is that we can easily appreciate the magnitude of $\overline{w}_{j\star}$, which should be much smaller than 1 for a healthy δf simulation.

3 Shape function of markers: basis functions used in expanding distribution function

3.1 Physical particles represented by a marker

A marker in PIC simulations represents a group of physical particles, which has their own distribution function given by

$$f_p(\mathbf{v}, \mathbf{r}) = w_p S_v(\mathbf{v} - \mathbf{v}_p) S(\mathbf{r} - \mathbf{r}_p), \tag{26}$$

where the subscript p denotes quantities on a marker, w_p is the weight of the marker defined above, $S_v(\mathbf{v} - \mathbf{v}_p)$ and $S(\mathbf{r} - \mathbf{r}_p)$ are the shape functions in the velocity space and real space, respectively. Here S_v and S have physical diemsion of $1 \,/\, \text{length}^3$ and $1 \,/\, \text{velocity}^3$ so that the physical dimension of f_p given in expression (26) is consistent with the physical dimension of a phase-space distribution function.

(A marker in PIC simulations is also called a "super-particle" or simply "particle", the latter can be confused with the physical particle and I will try to avoid using it.)

3.1.1 Shape in velocity space

In standard PIC simulations, the shape of markers in the velocity space is always a Dirac delta function:

$$S_v(\mathbf{v}-\mathbf{v}_p)=\delta^{(3)}(\mathbf{v}-\mathbf{v}_p), \tag{27}$$

where $\delta^{(3)}(\mathbf{v}-\mathbf{v}_p)$ is the three-dimensional Dirac delta function. In this case, all the physical particles represnted by a marker have the same veloicty $\mathbf{v}_p$.

[In terms of general velocity coordinates $(v_\alpha,\ v_\beta,\ v_\gamma)$, the three-dimensional Dirac delta function is defined via the 1D Dirac delta function as follows:

$$\delta^{(3)}(\mathbf{v}-\mathbf{v}_p)=\frac{1}{|\mathcal{J}_v|}\delta(v_\alpha-v_{\alpha p})\delta(v_\beta-v_{\beta p})\delta(v_\gamma-v_{\gamma p}), \tag{28}$$

where $\mathcal{J}_v$ is the the Jacobian of the general velocity coordinate system $(v_\alpha,\ v_\beta,\ v_\gamma)$. The Jacobian is included in order to make the 3D shape function satisfy the following normalization condition:

$$\int\delta^{(3)}(\mathbf{v}-\mathbf{v}_p)d\Gamma=\int\delta^{(3)}(\mathbf{v}-\mathbf{v}_p)|\mathcal{J}|dv_\alpha dv_\beta dv_\gamma=1. \tag{29}$$

]

3.1.2 Shape in real space

The shape function in the real space, $S(\mathbf{r}-\mathbf{r}_p)$, can also be chosen as the Dirac delta function (this is the choice for ions in `GEM` code). In conventional PIC, finite-size shape in spatial space is often adopted, i.e., the physical particles represented by a marker have different spatial locations. Along with the cell-averaging (discussed later), finite-size shape function has the effect of making the resulting self-consistent field more smooth, reducing (ideally completely removing) the artificial collision effects (the interaction among close particles) associated with using very few markers to approximate a plasma that has many physical particles in a Debye sphere[7].

We assume the shape function to be separable and to have the same functional dependence in all directions. Then a 3D shape function can be constructed by combining three 1D shape functions and the Jacobian of the coordinate system. For example, in (α, β, γ) coordinate system, a 3D shape function is written as

$$S(\mathbf{r}-\mathbf{r}_p)=\frac{1}{\mathcal{J}_r}\Big[\frac{1}{\Delta\alpha_p}S_{1D}\Big(\frac{\alpha-\alpha_p}{\Delta\alpha_p}\Big)\Big]\Big[\frac{1}{\Delta\beta_p}S_{1D}\Big(\frac{\beta-\beta_p}{\Delta\beta_p}\Big)\Big]\Big[\frac{1}{\Delta\gamma_p}S_{1D}\Big(\frac{\gamma-\gamma_p}{\Delta\gamma_p}\Big)\Big], \tag{30}$$

where $\mathcal{J}_r = \mathcal{J}_r(\alpha, \beta, \gamma)$ is the Jacobian of (α, β, γ) coordinate system, $\Delta\alpha_p$, $\Delta\beta_p$, and $\Delta\gamma_p$ are the scale-lengths of the support of the 1D shape function S_{1D} along α, β, and γ directions, respectively. The 1D shape function S_{1D} should satisfy the following normalization:

$$\int_{-\infty}^{\infty} S_{1D}(\xi-\xi_p)d\xi = 1, \tag{31}$$

where ξ is one of the spatial coordinates. [The reason to include the Jacobian in the definition of the 3D shape function is that this makes f_p be consistent with the definition of a distribution function, i.e., satisfy the following normalization:

$$\int f_p d\Gamma = w_p, \tag{32}$$

i.e.,

$$\int\Big(\int f_p \mathcal{J}_v dv_\alpha dv_\beta dv_\gamma\Big)\mathcal{J}_r\, d\alpha d\beta d\gamma = w_p. \tag{33}$$

]

Symmetric shapes are usually chosen, i.e. S_{1D} has the property: $S_{1D}(\xi-\xi_p) = S_{1D}(\xi_p-\xi)$. The support of S_{1D} should be compact (i.e. it is zero outside a small region) so that physical particles represnted by the marker are localized to a small portion of the space. Compact support of S_{1D} can make the deposition and gathering operations (defined later) computationally efficient.

Modern PIC codes usually use the so-called b-splines as the spatial shape functions. The b-spline functions are a series of consecutively higher order functions obtained from each other by integration:

$$b_l(\xi) = \int_{-\infty}^{\infty} b_{l-1}(\xi')b_0(\xi-\xi')d\xi' \tag{34}$$

with the 0th b-spline being the flat-top function defined as

$$b_0(\xi) = \begin{cases} 1 & \text{for } |\xi| < 1/2 \\ 0 & \text{other} \end{cases}. \tag{35}$$

Then, by using Eqs. (34) and (35), it is ready to derive the expression of $b_1(\xi)$, which is given by

$$b_1(\xi) = \begin{cases} 1 - |\xi| & \text{for } |\xi| < 1 \\ 0 & \text{other} \end{cases}. \tag{36}$$

The graphics of the first two b-splines, $b_0(\xi)$ and $b_1(\xi)$, are shown in Fig. 1.

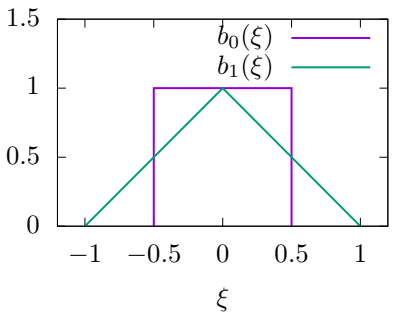

Figure 1. Graphics of the first two b-splines, $b_0(\xi)$ and $b_1(\xi)$.

The 1D shape function based on the b-spline function is then given by

$$S_{1D}(\xi) = b_l(\xi), \tag{37}$$

Most PIC codes use b_0, i.e, the flat-top function, as the shape function and choose the support of the shape function Δ_p equal to the grid spacing. (k-spectra of particle shape functions, to be continued)

3.2 Physical particle distribution function

The physical particle distribution is the sum of all the particle elements given in expression (26), i.e.,

$$f = \sum_p f_p = \sum_p w_p \delta^3(\mathbf{v} - \mathbf{v}_p) S(\mathbf{r} - \mathbf{r}_p), \tag{38}$$

which shows that the shape functions are essentially similar to the basis functions used in finite element methods.

The spatial shape of markers determines how physical particles represented by a marker are distributed to spatial cells (called deposition) and also how the force on a marker is related to the nearby electromagnetic field. [Note that the phase-space volume occupied (or sampled) by a marker is different from the concept of the spatial-shape of a marker.]

Update: later I realise that the finite shape function is not essential in making PIC method work, considering the fact that the Dirac delta function can be used as the shape function and gives reasonable results. It is the cell-averaing that is the most important thing that makes PIC simulation work.

3.3 Integration in velocity space

Consider a general velocity moment of the distribution function f:

$$I(\mathbf{r}) = \int_{-\infty}^{\infty} A(\mathbf{v}) f(\mathbf{r}, \mathbf{v})\, d\mathbf{v}, \tag{39}$$

where $A(\mathbf{v})$ is a known function. Using expression (38), the above moment is written as

$$I(\mathbf{r}) = \sum_p S(\mathbf{r} - \mathbf{r}_p) w_p \int_{-\infty}^{\infty} A(\mathbf{v}) \delta^3(\mathbf{v} - \mathbf{v}_p)\, d\mathbf{v}. \tag{40}$$

By using the property of the Dirac delta function, the above equation is written as

$$I(\mathbf{r}) = \sum_p w_p A(\mathbf{v}_p) S(\mathbf{r} - \mathbf{r}_p). \tag{41}$$

3.4 Cell-averaged velocity moment

One of the most important methods of reducing collisions between markers when using very few markers to approximate a system consisting of much more physical particles is to use the cell-averaged moments obtained from markers as the source term when solving Maxwell's equation on discrete gridpoints.

To be clear, grid points and the corresponding cells are defined as illustrated in Fig. 2 for the 1D case.

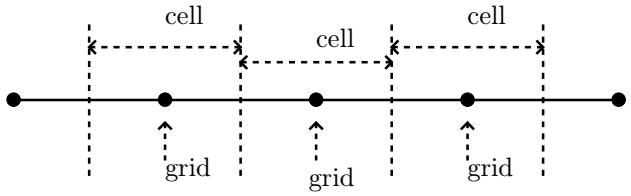

Figure 2. Definition of spatial grid points and cells in PIC simulations. A grid point is the center of the corresponding cell.

Field solvers in PIC code need the values of I at the grid-points. This value at the grid point is defined as the average of I over the corresponding cells (similar to that in the finite element method), i.e.,

$$I_{i,j,k} \equiv I(\alpha_i, \beta_j, \gamma_k) \equiv \frac{1}{\Delta V} \int_{\alpha_{i-1/2}}^{\alpha_{i+1/2}} \int_{\beta_{j-1/2}}^{\beta_{j+1/2}} \int_{\gamma_{k-1/2}}^{\gamma_{k+1/2}} I(\alpha, \beta, \gamma) \mathcal{J} d\alpha d\beta d\gamma,$$

where $\Delta V = \int_{\alpha_{i-1/2}}^{\alpha_{i+1/2}} \int_{\beta_{j-1/2}}^{\beta_{j+1/2}} \int_{\gamma_{k-1/2}}^{\gamma_{k+1/2}} \mathcal{J} d\alpha d\beta d\gamma$ is the cell volume, which can be approximated as $\Delta V \approx \mathcal{J}_{i,j,k} \Delta\alpha \Delta\beta \Delta\gamma$. Using Eq. (41), the above expression is written as

$$I_{i,j,k} = \frac{1}{\Delta V} \int_{\alpha_{i-1/2}}^{\alpha_{i+1/2}} \int_{\beta_{j-1/2}}^{\beta_{j+1/2}} \int_{\gamma_{j-1/2}}^{\gamma_{j+1/2}} \sum_p w_p A(\mathbf{v}_p) S(\mathbf{r} - \mathbf{r}_p) \mathcal{J} d\alpha d\beta d\gamma. \tag{42}$$

Using the shape function given in expression (30), the above expression is written as

$$\begin{aligned} I_{i,j,k} \;=\; & \frac{1}{\Delta V} \sum_p w_p A(\mathbf{v}_p) \int_{\alpha_{i-1/2}}^{\alpha_{i+1/2}} \int_{\beta_{j-1/2}}^{\beta_{j+1/2}} \int_{\gamma_{j-1/2}}^{\gamma_{j+1/2}} \\ & \left[\frac{1}{\Delta\alpha_p} S_{1D}\left(\frac{\alpha - \alpha_p}{\Delta\alpha_p} \right) \right] \left[\frac{1}{\Delta\beta_p} S_{1D}\left(\frac{\beta - \beta_p}{\Delta\beta_p} \right) \right] \left[\frac{1}{\Delta\gamma_p} S_{1D}\left(\frac{\gamma - \gamma_p}{\Delta\gamma_p} \right) \right] d\alpha d\beta d\gamma \end{aligned} \tag{43}$$

where the Jacobian in the integrand is cancelled out. The 3D integral in expression (43) consists of three identical 1D integrations. Consider one of them:

$$W_\alpha = \frac{1}{\Delta\alpha_p} \int_{\alpha_{i-1/2}}^{\alpha_{i+1/2}} S_{1D}\left(\frac{\alpha - \alpha_p}{\Delta\alpha_p} \right) d\alpha, \tag{44}$$

Then expression (43) is written as

$$I_{i,j,k} = \frac{1}{\Delta V}\sum_p w_p A(\mathbf{v}_p) W_\alpha W_\beta W_\gamma. \tag{45}$$

3.4.1 b-spline shape function

Consider the case $\Delta\alpha_p = \Delta\alpha$, and choose S_{1D} to be the lth order b-spline function, b_l, then expression (44) is written as

$$W_\alpha = \frac{1}{\Delta\alpha}\int_{\alpha_{i-1/2}}^{\alpha_{i+1/2}} b_l\Big(\frac{\alpha-\alpha_p}{\Delta\alpha}\Big)d\alpha \tag{46}$$

$$= b_{l+1}\Big(\frac{\alpha_i-\alpha_p}{\Delta\alpha}\Big), \tag{47}$$

where b_{l+1} is the $(l+1)$th order b-spline function.

[Proof of Eq. (47): Using the property of the zeroth order b-spline function b_0 (a flat top function), the integration (46) can be written as

$$W_\alpha = \frac{1}{\Delta\alpha}\int_{-\infty}^{\infty} b_l\Big(\frac{\alpha-\alpha_p}{\Delta\alpha}\Big) b_0\Big(\frac{\alpha-\alpha_i}{\Delta\alpha}\Big)d\alpha. \tag{48}$$

By using the definition of the b-splines, we find the above expression is a b-spline function that is one order higher than the corresponding b-spline shape function, i.e.,

$$W_\alpha = b_{l+1}\Big(\frac{\alpha_i-\alpha_p}{\Delta\alpha}\Big). \tag{49}$$

]

Therefore expression (43) is written as

$$I_{i,j,k} = \frac{1}{\Delta V}\sum_p w_p A(\mathbf{v}_p) b_{l+1}\Big(\frac{\alpha_i-\alpha_p}{\Delta\alpha}\Big) b_{l+1}\Big(\frac{\beta_j-\beta_p}{\Delta\beta}\Big) b_{l+1}\Big(\frac{\gamma_k-\gamma_p}{\Delta\gamma}\Big). \tag{50}$$

Recall that, in terms of the b-spline functions, the local value $I(\alpha,\beta,\gamma)$ is given by expression (41), i.e.,

$$I(\alpha,\beta,\gamma) = \frac{1}{\Delta V}\sum_p w_p A(\mathbf{v}_p) b_l\Big(\frac{\alpha-\alpha_p}{\Delta\alpha}\Big) b_l\Big(\frac{\beta-\beta_p}{\Delta\beta}\Big) b_l\Big(\frac{\gamma-\gamma_p}{\Delta\gamma}\Big). \tag{51}$$

It is instructive to compare expression (50) with (51): they are similar except that the b-spline functions involved in the cell-averaged expression (50) is one order higher than that involved in the local expression (51).

3.4.2 Dirac delta function as shape function

If S_{1D} is chosen as the Dirac delta function and $\Delta\alpha_p = \Delta\alpha$, then expression (44) is written as

$$W_\alpha = \frac{1}{\Delta\alpha}\int_{\alpha_{i-1/2}}^{\alpha_{i+1/2}} \delta\Big(\frac{\alpha-\alpha_p}{\Delta\alpha}\Big)d\alpha = \begin{cases} 1 & \text{for } \left|\dfrac{\alpha_i-\alpha_p}{\Delta\alpha}\right| \leqslant \dfrac{1}{2} \\ 0 & \text{for } \left|\dfrac{\alpha_i-\alpha_p}{\Delta\alpha}\right| > \dfrac{1}{2} \end{cases}, \tag{52}$$

which is the zero order b-spline function (flat-top function) $b_0\Big(\frac{\alpha_i-\alpha_p}{\Delta\alpha}\Big)$. (This also indicates that the Dirac delta function can be considered as a b-spline function that is one order less than b_0. So the cell average promotes the delta function to b_0.) It is obvious that depositing particles to grids corresponds to the neareast neighbour interpolation when using the above interpolation function.

In the `GEM` gyrokinetic code, the particle shape is chosen to be Dirac delta function.

3.5 Effective field on a marker

The effective field on a marker is defined as the averaged field on the group of particles represented by the marker. To get the averaged field, we need to reconstruct a continuous field from the discrete field defined on the gridpoints and then average the field over all the physical particles contained in the marker.

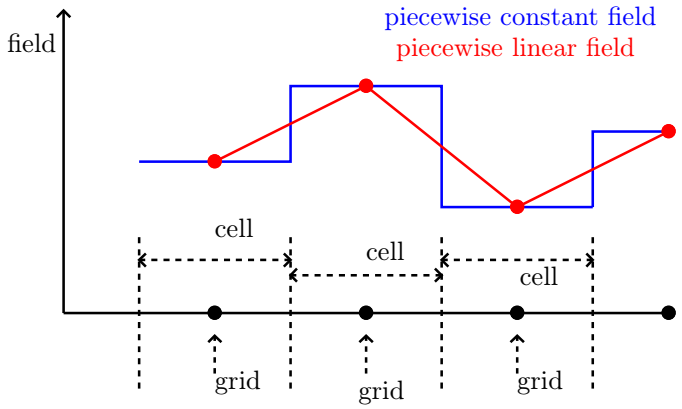

Figure 3. Reconstructing continuous field from discrete field defined on gridpoints by using piecewise constant function (blue) or piecewise linear function (red). The picewise constant field usually has jumps across cell boundaries.

3.5.1 Piecewise constant field

One method of reconstructing the continuous electric field is to assume that the field is constant in each cell, i.e., piecewise constant function. In terms of the zeroth order b-spline (flat-top function), the continous field is written

$$E(\alpha,\beta,\gamma)=\sum_{i,j,k} E_{i,j,k} b_0\Big(\frac{\alpha-\alpha_i}{\Delta\alpha}\Big) b_0\Big(\frac{\beta-\beta_i}{\Delta\beta}\Big) b_0\Big(\frac{\gamma-\gamma_i}{\Delta\gamma}\Big), \tag{53}$$

where $E_{i,j,k}$ is the field value at grid-points (centers of cells) obtained by solving the field equation. The field in Eq. (53) usually has jumps across cell boundaries, as is shown in Fig. 3.

The electric field on a computational marker is the average of the electric field over all the physical particles contained in the marker, i.e.,

$$E_p=\frac{1}{w_p}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} E(\alpha,\beta,\gamma)\Big(\int f_p d\mathbf{v}\Big)\mathcal{J}_r d\alpha d\beta d\gamma \tag{54}$$

Using f_p given in Eq. (26) in the above equation, we obtain

$$\begin{aligned} E_p \;=\; & \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} E(\alpha,\beta,\gamma)\times \\ & \Big[\frac{1}{\Delta\alpha_p} S_{1D}\Big(\frac{\alpha-\alpha_p}{\Delta\alpha_p}\Big)\Big]\Big[\frac{1}{\Delta\beta_p} S_{1D}\Big(\frac{\beta-\beta_p}{\Delta\beta_p}\Big)\Big]\Big[\frac{1}{\Delta\gamma_p} S_{1D}\Big(\frac{\gamma-\gamma_p}{\Delta\gamma_p}\Big)\Big] d\alpha d\beta d\gamma, \end{aligned} \tag{55}$$

where the Jacobian and w_p disappear. Use the reconstructed electric field [Eq. (53)] in the above equation, giving

$$\begin{aligned} E_p \;=\; & \sum_{i,j,k} E_{i,j,k}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} b_0\Big(\frac{\alpha-\alpha_i}{\Delta\alpha}\Big) b_0\Big(\frac{\beta-\beta_i}{\Delta\beta}\Big) b_0\Big(\frac{\gamma-\gamma_i}{\Delta\gamma}\Big)\times \\ & \frac{1}{\Delta\alpha_p} S_{1D}\Big(\frac{\alpha-\alpha_p}{\Delta\alpha_p}\Big)\frac{1}{\Delta\beta_p} S_{1D}\Big(\frac{\beta-\beta_p}{\Delta\beta_p}\Big)\frac{1}{\Delta\gamma_p} S_{1D}\Big(\frac{\gamma-\gamma_p}{\Delta\gamma_p}\Big) d\alpha d\beta d\gamma \end{aligned} \tag{56}$$

If we choose the shape function S_{1D} to be the Dirac delta function, then expression (56) will correspond to the neareast grid points interpolation. (In the `GEM` code, computing field on particles corresponds to the near-east grid point interpolation, which indicates that the particle shape is chosen to be Dirac delta function.)

If we choose the shape function S_{1D} as the b-spline function b_l with $\Delta\alpha_p = \Delta\alpha$, $\Delta\beta_p = \Delta\beta$, and $\Delta\gamma_p = \Delta\gamma$, then the above equation is written as

$$E_p = \sum_{i,j,k} E_{i,j,k} b_{l+1}\Big(\frac{\alpha_i - \alpha_p}{\Delta\alpha}\Big) b_{l+1}\Big(\frac{\beta_j - \beta_p}{\Delta\beta}\Big) b_{l+1}\Big(\frac{\gamma_k - \gamma_p}{\Delta\gamma}\Big), \tag{57}$$

which specifies how the effective field on a marker is related to the nearby field on the grid-points. If $l = 0$, equation (57) corresponds to a linear interpolation.

3.5.2 Piecewise linear field

Another way of reconstructing the continuum electric field is to assume that the field is linear function between grids, as is shown in Fig. 3. In this case, if the spatial shape function of markers is Dirac delta function, then it is obvious that the effective field on a marker can be obtained by linearly interpolating the fields on the nearyby grids. If the spatial shape function of markers is b_0, what is the interpolation scheme of obtaining the effective field on a marker? To be worked out. The result will be a little complicated than the linear interpolation and thus involves more computational overhead. The benefit can be that the noise can be further reduced, compared with the linear interpolation. But this seems to be seldom used in real world codes.

3.5.3 Effective force on a marker

The total charge of a group of particles represented by a maker, Q, is given by $Q = \int q f_p d^6v$, where q is the charge of a single particle. Then the effective force on a marker is then $F_p = Q E_p$ with E_p given by Eq. (57). The total mass of a group of particles represented by a marker, M, is given by $M = \int m f_p d^6v$, where m is the mass of a single particle. Then the ratio between Q and M is written as

$$\frac{Q}{M} = \frac{\int q f_p d^6v}{\int m f_p d^6v} = \frac{q}{m}, \tag{58}$$

which is identical to the single particle charge mass ratio. Note that the motion equation of a particle in an electromagnetic field is distinguished only by this ratio. Therefore motion of a marker in the phase space is identical with the motion of a real particle with the effective field given by Eq. (57).

3.6 Numerical implementation in codes

Most PIC codes use the $l = 0$ b-spline function (i.e., b_0, the flat-top function) as the shape function of markers. This model is often called Could in Cell (CIC) since a particle looks like a finite-sized cloud rather than a point. In this case, the cell average of I given in Eq. (50) is written as

$$I_{i,j,k} = \frac{1}{\Delta V} \sum_p w_p A(\mathbf{v}_p) b_1\Big(\frac{\alpha_i - \alpha_p}{\Delta\alpha}\Big) b_1\Big(\frac{\beta_j - \beta_p}{\Delta\beta}\Big) b_1\Big(\frac{\gamma_k - \gamma_p}{\Delta\gamma}\Big), \tag{59}$$

and the electric field on the marker given in Eq. (57) is written as

$$E_p = \sum_{i,j,k} E_{i,j,k} b_1\Big(\frac{\alpha_i - \alpha_p}{\Delta\alpha}\Big) b_1\Big(\frac{\beta_j - \beta_p}{\Delta\beta}\Big) b_1\Big(\frac{\gamma_k - \gamma_p}{\Delta\gamma}\Big), \tag{60}$$

Because the function b_1 has a narrow support, as shown in Fig. 1, in practice of calculating $I_{i,j,k}$ in expression (59), we loop over each particle for only once and assign the contribution of each one to their neighbouring cells (rather than looping over all particle for each cell as the straightforward reading of expression (59) would suggest). The operation of the latter would be $O(N \times N_p)$, where N is number of grids and N_p is the number of markers, wherea the former is only $O(nN_p)$, where n is the number of operation involved in assigning each particle to its neighbouring cells, which is usually much smaller than the grid number N.

Similarly, in calculating the force on a marker, the summation over all the grids (as the straightforward reading of expression (60) would suggest) is not needed. We only need to find which cell a marker is in and then sum the contribution from the nearby grids (rather than all the grids).

3.7 Monte-Carlo integration revisited

Consider a general integration of the distribution function f in the phase-space

$$I \equiv \int_{\Omega'} A(\mathbf{Z}) f(\mathbf{Z})\, dV, \tag{61}$$

where Ω' is a sub-region of the phase space Ω, $A(\mathbf{Z})$ is a general function of the phase-space coordinates $\mathbf{Z}$, dV is the phase space volume element. As is discussed above, in particle methods, $f(\mathbf{Z})$ is approximated by

$$f(\mathbf{Z}) \approx \sum_{j=1}^{N} w_j S_{\mathrm{ps}}(\mathbf{Z} - \mathbf{Z}_j), \tag{62}$$

where $S_{\mathrm{ps}}(\mathbf{Z} - \mathbf{Z}_j)$ is the phase space shape function of markers, N is the total number of marker loaded in the phase space Ω. Using this, expression (61) is written as

$$I = \sum_{j=1}^{N} \int_{\Omega'} A(\mathbf{Z})\, w_j S_{\mathrm{ps}}(\mathbf{Z} - \mathbf{Z}_j) dV, \tag{63}$$

If the shape function S_{ps} is chosen to be the Dirac delta function, then the above equation is written as

$$I = \sum_{j=1}^{N'} A(\mathbf{Z}_j) w_j, \tag{64}$$

where N' is the number of markers that are within the sub-region Ω'. Equation (64) is the Monte-Carlo approximation to the integration in Eq. (61)[4, 2].

In PIC simulations, we are usually interested in velocity moments of f averaged in a spatial cell, i.e., $\mathbf{A}$ in Eq. (63) is only a function of only $\mathbf{v}$ and the velocity integration is over the whole velocity space, whereas the spatial integration is over a small spatial cell. Furthermore, in PIC simulations, the markers can have a finite shape in the real space (the shape in velocity space is still the Dirac delta function). In this case, Eq. (63) is written as

$$\begin{aligned} I &= \sum_{j=1}^{N} \int_{\mathbf{v}} A(\mathbf{v})\, w_j \delta^{(3)}(\mathbf{v} - \mathbf{v}_j) \left(\int_{\Delta V_r} S_r(\mathbf{r} - \mathbf{r}_j) d\Gamma_r \right) d\Gamma_v, \\ &= \sum_{j=1}^{N} A(\mathbf{v}_j) w_j \left(\int_{\Delta V_r} S_r(\mathbf{r} - \mathbf{r}_j) d\Gamma_r \right), \end{aligned} \tag{65}$$

where S_r is the shape function. Then the cell-averaged value is written as

$$\frac{I}{\Delta V_r} = \sum_{j=1}^{N} A(\mathbf{v}_j) w_j \left(\frac{1}{\Delta V_r} \int_{\Delta V_r} S_r(\mathbf{r} - \mathbf{r}_j) d\Gamma_r \right). \tag{66}$$

3.8 On noise and accuracy of particle methods

particle methods vs. Euler-grid-based methods

The overall error of the Monte-Carlo approximation given in Eq. (64) always scales like $1/\sqrt{N'}$, which is independent of the dimensionality of the phase-space. It can be proved that the overall error of the usual regular-grids approximation to the integration scales like $1/N'^{1/d}$, where d is the dimension of the phase space[4]. This fact implies that Monte-Carlo approximation is more accurate than the regular-grids methods for high-dimension ($d \geqslant 3$) integration. Due to this reason, particle methods can be considered more accurate than the Euler-grid-based methods for the same number of sampling points in a high-dimensional ($\geqslant 3$) phase space.

On the other hand, PIC simulations obviously contain unphysical noise. The noise is due to the discrete marker effects, which can be further categorized into two types: sampling noise (fluctuation of the sampling error, statistical noise) and remaining unwanted collisions in a collisionless simulation.

Due to the limited small number of markers used in PIC code, there are considerable time and spatial fluctuation over the the number of markers in a spatial-cell. This fluctuation in the number of sampling points (i.e., fluctuation of the sampling error) gives rise to the sampling noise.

Inaccuracy in a PIC simulation is also related to the fact that the phase-space volume sampled by a marker is assumed to be constant in a PIC code but this assumption is not strictly satisfied in practice due to (1) the number of markers being not large enough and (2) the resulting self-consistent field being not smooth enough, which introduce effective collisional effects, making the conservation of the phase-space volume less accurate. How well the phase-space volume is conserved depends on the smoothness of the field: smooth field means less collisions and thus phase-space volume are better conserved. PIC simulation codes seek to reduce the collision through using finite-size particles (discussed in Sec. 3) and averaging in a spatial cell in solving for the electromagnetic fields, which effectively smooth the fields. This kind of PIC simulations are thus designed for collisionless plasmas. And the remaining collisional effect should be small enough to not affect the process of interest. And this remaining collisional effects should be viewed as numerical

artifacts rather than a modeling of any real collisional effect in plasmas. If we want to model the real collisional effect in PIC simulation, we need to use other techniques rather than relying on the remaining collisions mentioned above because the latter is not easy to control and ideally should be completely removed.

Various noise reduction techniques in PIC codes (e.g., finite-size particles, grids, and perturbative δf method) can be used when the marker number is fixed. When exhausting all these methods, the final brute-force method of reducing noise (reducing collisions) is to increase the marker number. Therefore the noise issue is finally a convergence issue about the marker number.

Also note that low noise is not synonymous with high accuracy. After reducing the noise level to an accepted level (e.g. by using δf method), we also need to verify that the low-noise result converges to the physical result by increasing the marker number.

From the view of particle simulations, the gyrokinetic model can also be considered as a noise reduction technique, where (1) 6D distribution function is infered from a 5D numerical distribution function, and thus is less noisy compared with computing directly 6D numerical distribution function; and (2) the gyro-averaging process makes the field on a marker more smooth.

One thing to note is that the noisy results obtained in particle method are not necessarily less accurate than the smooth results obtained in Euler-grid-based methods because bigger errors may be hidden in smooth results when one uses coarse grids. I.e., low or no noise is not synonymous with high accuracy.

My understanding and experience on this issue is being improved. The conclusions here seems reasonable but need direct numerical verification.

(Noise in PIC code is equivalent to the remaining artificial collision effect? We can test this by doing a test particle simulation, in which we loaded a group of markers to sample a distribution function in the phase space and then compute the density evolution of the sampled distribution under a given smooth electromagnetic field (i.e., eliminating the collisional effect). If there is still significant noise in the time evolution, then this indicates there are factors other than collisions that contribute to the noise. I did this when I studied Landau damping, the results indicate there is still significant noise in the solution, indicating the discrete phase space sampling is the root of the noise.)

noise-reduction is equivalent to the variance reduction?

Most improvements to Monte Carlo methods are variance-reduction techniques [2].

3.9 Practical comparison between particle methods and Euler-grid-based methods

Kinetic plasma simulations in a 6D phase space is usually difficult for grid-based methods because the six-dimensional phase space seems to be too high for Euler-gird-based methods to handle. As a result, there seems to be no 6D grid-based kinetic simulation code. With the invention of the gyrokinetic model, the dimenionality of the phase space is reduced to five, which makes it possible for Euler-grid-based method to handle. As a result, there appear several widely used gird-based gyrokinetic codes, such as `GYRO` and `GENE`.

Another reason why particle methods are more popular than the Eulerian-grid-based methods in plasma community is the algorithmic compactness and subsequent ease of coding in comparison with the Eulerian-grid-based methods. The popularity of PIC methods may also be related to that PIC method seems more intuitive at first glance (although this first impression may be only partially correct and even misleading). This is not about accuracy or even science. Historical or psychological reasons can partially account for the popularity of one particular method.

Parallization of marker pushing part of PIC codes is usually easy. All markers used in a simulation can be evenly distributed among all the MPI processes used. The loading balance can be good among all the processes in terms of marker pushing if proper domain decomposition and loading scheme are used to make the marker number in each MPI process apprxoimately equal to eah other. The deposition part of a PIC code can be slow because it involves random access to a large grid array when looping over all the markers in a MPI process. The parallization of the field solving part is the same as Euler based codes.

For gyrokinetic simulation of tokamak plasmas, it is not easy to judge which method, particle-based method or grid-based mthod, is better than the other. When dealing with full Landau collision oprators, some PIC codes, e.g., `XGC`, uses velocity space grids. This may indicate that velocity grids are generally needed, suggesting that grids-methods may have advantages. On the other hand, hybrid methods that use both velocity grids and markers may be a practical way since "hybrid" provides us freedoms to use whatever methods available that can get work done.

3.10 Modeling collisions in PIC simulations

(check**However, the grid-less approach in particle method makes it difficult to handle general Landau collision operators that involve the velocity gradients of the distribution function and evaluating these gradients usually needs velocity grids. For some simple collision operators, a corresponding Langevin equation can be constructed, which makes the collisional effect be able to be modeled by stochastic change in marker trajectories. However, this correspondence can not be found for all collisional operators.)

4 Evolution of distribution functions

The marker weight consists of two parts: the physical distribution function f and the marker distribution function g. The time evolution of the weight w is thus determined by the time evolution of f and g, which are discussed in Sec. 4.1 and 4.2, respectively.

4.1 Time evolution of the physical distribution function

4.1.1 Full-f formula

The collisionless kinetic equation (Vlasov equation) is given by

$$\frac{\partial f}{\partial t}+\mathbf{v}\cdot\frac{\partial f}{\partial \mathbf{x}}+\frac{q}{m}(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\frac{\partial f}{\partial \mathbf{v}}=0, \tag{67}$$

It is ready to find that the characteristic lines of the equation is given by

$$\frac{d\mathbf{x}}{dt}=\mathbf{v}, \tag{68}$$

and

$$\frac{d\mathbf{v}}{dt}=\frac{q}{m}(\mathbf{E}+\mathbf{v}\times\mathbf{B}). \tag{69}$$

Along a characteristic line, the kinetic equation is written

$$\frac{df}{dt}=0, \tag{70}$$

where f is the total particle distribution function (full-f). Particles methods that use Monte-Carlo sampling to approximate the total f are called total-f or full-f method.

4.1.2 Delta-f formula

Since the phenomena we consider in tokamak plasmas are usually developing from an equilibrium or near-equilibrium state, it is desirable to calculate the the evolution of the only the perturbation, instead of the total f. Therefore, we write f as the sum of a known time-independent distribution function f_0 ($\partial f_0 / \partial t = 0$) and a unknown time-dependent perturbation δf ($\partial \delta f / \partial t \neq 0$):

$$f = f_0 + \delta f, \tag{71}$$

Then the kinetic equation (70) is written

$$\frac{d\delta f}{dt} = -\frac{df_0}{dt}. \tag{72}$$

Particle methods that use Monte-Carlo sampling to approximate the δf are usually called δf method. When the right-hand side of Eq. (72) is known, Eq. (72) can be integrated to obtain the time evolution of δf.

The time evolution of δf can also be obtained by using

$$\delta f(t, \mathbf{Z}_j(t)) = f(t=0, \mathbf{Z}_j(t=0)) - f_0(\mathbf{Z}_j(t)), \tag{73}$$

where the total distribution f at $t = 0$ are known (f at $t = 0$ can be obtained by summing $f_0(\mathbf{Z}_j(t = 0))$ and $\delta f(t = 0, \mathbf{Z}_j(t = 0))$, both of which are known when we start the simulation). Using Eq. (73) to update δf avoids the time integration of $d\delta f / dt$, which may reduce the computational overhead and improve the accuracy of δf. This seemingly trivial method was emphasized in a CPC paper[1], which introduces an adaptive f_0 method based on this idea. I have tested this method in my toy code about the 1D Landau damping, which gave the same results as the usual method. However, Yang Chen pointed out that this method is never used in simulations of tokamak plasmas due to the following reasons. The chosen equilibrium distribution function f_0 in tokamak plasmas simulation is usually not a solution the following time-independent Vlasov equation:

$$\mathbf{v} \cdot \frac{\partial f_0}{\partial \mathbf{x}} + \frac{q}{m}(\mathbf{E}_0 + \mathbf{v} \times \mathbf{B}_0) \cdot \frac{\partial f_0}{\partial \mathbf{v}} = 0, \tag{74}$$

Instead, the chosen f_0 is a solution of the following equation

$$\mathbf{v} \cdot \frac{\partial f_0}{\partial \mathbf{x}} + \frac{q}{m}(\mathbf{E}_0 + \mathbf{v} \times \mathbf{B}_0) \cdot \frac{\partial f_0}{\partial \mathbf{v}} = S, \tag{75}$$

where S is a nonzero term. On the other hand, the perturbation part is governed by the following equation

$$\frac{d\delta f}{dt} = -\frac{q}{m}(\delta \mathbf{E} + \mathbf{v} \times \delta \mathbf{B}) \cdot \frac{\partial f_0}{\partial \mathbf{v}}. \tag{76}$$

Then it is ready to prove that the actual kinetic equation for the full f in the usual tokamak simulation is actually given by

$$\frac{df}{dt} = S, \tag{77}$$

which indicates that f is not a constant along the characteristic curves and thus is not consistent with the algorithm given in Eq. (73). (**may be wrong, check**The case given in Eq. (77) is the more relevant case to tokamak experiments where a particular form of f_0 is maintained by external sources and we are asked to calculate the perturbation evolution around f_0.**wrong!)

4.2 Time evolution of marker distribution function

For Hamiltonian motion in particular and non-diffusive motion in general, any particle distribution function $g(\mathbf{x}, \mathbf{v}, t)$ satisfies $dg/dt = 0$. In other words, the phase-space particle flow is volume preserving (this is Louisville's theorem), i.e.,

$$\frac{dV_j}{dt} = 0. \tag{78}$$

i.e, the phase-space volume sampled by a marker does not change along the characteristic curves. This is good news for PIC simulation because this means that the marker distribution is known exactly along the characteristic line at every time-step and we do not need to numerically evaluate it (numerically evaluating the marker distribution by directly counting markers would be noisy due to the small number of markers loaded and should be avoided in practice whenever possible).

For diffusive motion, the phase space particle flow is usually not volume preserving, i.e.,

$$\frac{dg(\mathbf{Z}_j)}{dt} \neq 0. \tag{79}$$

For this case, the values of g at markers need to be evaluated numerically every time step, which is usually noisy and time-consuming in terms of CPU time. Therefore, this kind of evaluation should be avoided in practice whenever possible. The usual way of achieving this is by choosing a suitable initial distribution for the markers, so that g remains approximately constant along the trajectory of markers[3].

4.3 Time evolution of marker's weight

As mentioned in Sec. (), marker's weight, $w = f/g$, is composed of two factors, namely the physical distribution function f and the markers' distribution function g. The time evolution of the weight w is thus

determined by the time evolution of f and g.. In some cases, the formula for the time evolution of both f and g can be obtained analytically, as discussed in Sec. 4.1 and 4.2, respectively. [*check**In other cases, the formula for the time evolution of f and/or g can not be easily obtained but the formula for the time evolution of w can still be obtained analytically. A typical example of this kind is the full-f simulation including the collision effects in the orbits. In this case, the phase-space flow is not volume-conserving, i.e., $dg/dt \neq 0$, and it is difficult to find an analytic formula for the time evolution of g. However, the conservation of the particle number along the orbits in the phase space is still valid (check this!, wrong), i.e.,

$$\frac{dw}{dt} = 0. \tag{80}$$

Does this algorithm correctly describe the "collision"? This algorithm (i.e., including the collision effects via randomly changing the orbit variables) is obviously correct when w is 1, i.e., a marker only represents one physical particle. The correctness for $w_j > 1$ case should be verified. ***check]. Collisions in δf methods seem to be implemented by including a source term in the evolution equation for the weight, rather than kicking the orbit (I will check the kicking orbit method later).

5 An example: One-dimensional electrostatic simulation

5.1 Vlasov equation

Consider the electrostatic case. The Vlasov equation () for electrons is written

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + \frac{e}{m} \nabla \phi \cdot \nabla_v f = 0, \tag{81}$$

where ϕ is the electric potential. Consider the one-dimensional case where f and ϕ are independent of y and z coordinates. In this case, the above equation is written

$$\frac{\partial f}{\partial t} + v_x \frac{\partial f}{\partial x} + \frac{e}{m} \frac{d\phi}{dx} \frac{\partial f}{\partial v_x} = 0 \tag{82}$$

Integrating both sides of the above equation over v_y and v_z, we obtain

$$\frac{\partial F}{\partial t} + v_x \frac{\partial F}{\partial x} + \frac{e}{m} \frac{d\phi}{dx} \frac{\partial F}{\partial v_x} = 0, \tag{83}$$

where $F(x,\, v_x,\, t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f dv_y dv_z$ is the reduced distribution function. Define characteristic lines by the following ordinary differential equations:

$$\frac{dx}{dt} = v_x, \tag{84}$$

and

$$\frac{dv_x}{dt} = \frac{e}{m_e}\frac{d\phi}{dx}. \tag{85}$$

Then along a characteristic line, we obtain

$$\frac{dF}{dt} = 0, \tag{86}$$

which indicates that the distribution function F remain unchanged along a characteristic line.

5.2 Poisson's equation

In terms of the reduced distribution function F, Poisson's equation is written

$$-\frac{d^2\phi}{dx^2} = \frac{e}{\varepsilon_0}\left(n_{\text{ion}} - \int_{-\infty}^{\infty} F dv_x\right), \tag{87}$$

where n_{ion} is the number density of ions, which are assumed to be uniform in x and not evolving with time.

5.3 Equilibrium state

Consider a spatially uniform distribution function given by

$$F(x, v_x, t) = F_0(v_x), \tag{88}$$

where $F_0(v_x)$ is a known velocity distribution function with number density being equal to those of ions, i.e., $\int_{-\infty}^{+\infty} F_0(v_x) dv_x = n_{\text{ion}}$. Consider a case with zero electric field, i.e.,

$$E(x, t) = 0. \tag{89}$$

Then it is ready to verify that expression specified by Eqs. (88) and (89) is a equilibrium solution to Vlasov-Poisson system (Eqs. (83) and (87)).

In this note, two kind of equilibrium distribution functions will be considered. The first one is the Maxwellian distribution given by

$$F_0(v_x) = \frac{n_{e0}}{\sqrt{\pi} v_t}\exp\left(-\frac{v^2}{v_t^2}\right). \tag{90}$$

In this system, small perturbation will be damped by a mechanism known as Landau damping. The second kind of distribution considered is the two-stream Maxwellian distribution given by

$$F_0(v_x) = \frac{n_{e0}}{\sqrt{\pi} v_t} \frac{1}{2} \left[\exp\left(-\frac{(v - v_b)^2}{v_t^2} \right) + \exp\left(-\frac{(v + v_b)^2}{v_t^2} \right) \right]. \tag{91}$$

In this system, small perturbation will give rise to an instability known as the two-stream instability.

5.4 δf evolution

Write the distribution function F as

$$F = F_0 + \delta F, \tag{92}$$

where F_0 is the equilibrium distribution function. Then Eq. (86) is written as

$$\frac{d\delta F}{dt} = -\frac{dF_0}{dt}. \tag{93}$$

Use the definition of the orbit propagator,

$$\frac{d}{dt} \equiv \frac{\partial}{\partial t} + v_x \frac{\partial}{\partial x} + \frac{e}{m} \frac{d\phi}{dx} \frac{\partial}{\partial v_x}, \tag{94}$$

to rewrite the right-hand side of Eq. (93), yielding

$$\frac{d\delta F}{dt} = -\left(\frac{e}{m} \frac{d\phi}{dx} \frac{\partial F_0}{\partial v_x} \right), \tag{95}$$

which can be integrated to obtain the time evolution of δF.

The time evolution of δF can also be obtained by using

$$\delta F(\mathbf{Z}_j(t)) = F(\mathbf{Z}_j(t=0)) - F_0(\mathbf{Z}_j(t)). \tag{96}$$

In this way, the time integration of $\delta F \,/\, dt$ is avoided, which may reduce the computational work load and improve the accuracy of δF. This seemingly trivial method was emphasized in a CPC paper[1], which introduces an adaptive F_0 method based on this idea. I have compared the results of Landau damping obtained by the two methods (i.e., using Eq. (95) and Eq. (96), respectively), which shows they agree with each other very well.

5.5 Normalization

Choose a typical number density n_0 and a typical velocity v_0, then

define the electron plasma frequency ω_{pe} and the Debye length λ_D as

$$\omega_{pe} = \sqrt{\frac{n_0 e^2}{m_e \varepsilon_0}}, \tag{97}$$

and

$$\lambda_D = \frac{v_0}{\omega_{pe}}, \tag{98}$$

respectively. Using ω_{pe} and λ_D, define the following normalized quantities:

$$\bar{t} = t\omega_{pe},\ \bar{x} = \frac{x}{\lambda_D},\ \bar{\phi} = \frac{e\phi}{m_e v_0^2},\ \bar{F} = \frac{v_0}{n_0} F, \bar{v}_x = \frac{v_x}{v_0}, \bar{n}_i = \frac{n_i}{n_0} \tag{99}$$

In terms of these normalized quantities, equation (86) is written

$$\frac{d\bar{F}}{d\bar{t}} = 0, \tag{100}$$

and the Poisson equation (87) is written

$$\frac{d^2\bar{\phi}}{d\,\bar{x}^2} = -\bar{\rho}. \tag{101}$$

where $\bar{\rho} = \bar{n}_{\text{ion}} - \int_{-\infty}^{\infty} \bar{F}\, d\,\bar{v}_x$.

The equations for the characteristic lines, Eq. (84) and (85), are written

$$\frac{d\,\bar{x}}{d\,\bar{t}} = \bar{v}_x, \tag{102}$$

$$\frac{d\,\bar{v}_x}{d\,\bar{t}} = \frac{d\bar{\phi}}{d\,\bar{x}}. \tag{103}$$

The electric field is given by $E = -d\phi\,/\,dx$, which, in terms of normalized quantities, is written

$$\bar{E} = -\frac{d\bar{\phi}}{d\,\bar{x}}, \tag{104}$$

where $\bar{E} = Ee\lambda_D/(mv_0^2)$.

In terms of the normalized quantities, the evolution equation (95) of δF is written as

$$\frac{d\delta\bar{F}}{d\,\bar{t}} = \bar{E}\frac{\partial\bar{F}_0}{\partial\bar{v}_x} \tag{105}$$

In terms of $\delta\bar{F}$, Poisson's equation is written as

$$\frac{d\bar{\phi}}{d\,\bar{x}^2} = \int_{-\infty}^{\infty} \delta\bar{F}\, d\,\bar{v}_x, \tag{106}$$

where $n_{\text{ion}} = n_{e0}$ has been assumed.

5.6 Boundary condition for field

The periodic boundary condition is used for the electrical field E, i.e., the electric field on the right boundary is set equal to that on the left boundary.

5.7 Boundary condition for particles

The periodic boundary condition is also used for markers, i.e., a marker that leaves from the right boundary will re-enter the computational region from the left boundary and vice versa.

5.8 Evaluation of particle number density

Set up uniform grid-points in x-direction: $x_j = j\Delta$ for $j = 0, 1, 2, ..., N$, as is shown in Fig. 4. Use the first b-spline (flat-top function) with a support $\Delta_p = \Delta$ as the shape function of markers. It is obvious that we can use the following procedures to obtain the number of physical particles in each cell. For a particle marker labeled by k whose position is $x = r$, we can find which two grids the particle lies between. Suppose that r is between x_j and x_{j+1}, then the particle number n_j and n_{j+1} is evaluated as follows

$$n_j \to n_j + w_k \cdot \frac{x_{j+1} - r}{\Delta}, \; n_{j+1} \to n_{j+1} + w_k \cdot \frac{r - x_j}{\Delta}, \tag{107}$$

where w_k is the weight of the marker. Performing the same procedures for each marker in turn allows us to build up n_j at all the grid points (all n_j are set to zero before these procedures).

The particle number at the boundary grid-points $j = 0$ and $j = N$ needs special treatment. The above treatment do not include the contribution to n_0 from the left cell of the $j = 0$. Since we use periodic boundary condition, the contribution to n_0 from the left cell of the $j = 0$ grid is identical to the contribution to n_N from the left cell of the $j = N$ grid. The latter has already been calculated in the above, which can be added to n_0 obtained above to get the final n_0. The same situation applies for n_N. After these treatment, we have $n_0 = n_N$.

Dividing the particle number n_j by the cell size Δ gives the number density.

5.9 FFT solver for Poisson equation

The normalized one-dimensional Poisson equation is given by Eq. (101). For notation simplicity, omit the over-bar on ϕ and x, then Eq. (101) is

written

$$\frac{d^2\phi}{dx^2} = -\rho. \tag{108}$$

This is a two-points boundary value problem. Two boundary conditions are needed to determine the solution. Assume the periodic boundary condition $\phi(0) = \phi(L)$ and note that ϕ can contain an arbitrary constant. Thus the periodic boundary condition alone is sufficient to specify the electrical field. We use Fourier transformation method to solve Eq. (108). The Fourier transformation of the left-hand side of the above equation is written

$$\begin{aligned} \int_{-\infty}^{\infty} \frac{d^2\phi}{dx^2} e^{ikx} dx &= -k^2 \int_{-\infty}^{\infty} \phi e^{ikx} dx \\ &= -k^2 \hat{\phi}(k), \end{aligned} \tag{109}$$

where $\hat{\phi}$ is the Fourier transformation of ϕ. Using this, the Fourier transformation of Eq. (108) is written

$$\hat{\phi}(k) = \frac{\hat{\rho}(k)}{k^2}, \tag{110}$$

where

$$\hat{\rho} = \int_{-\infty}^{\infty} \rho(x) e^{ikx} dx \tag{111}$$

is the Fourier transformation of ρ. After $\hat{\phi}$ is obtained, the electric potential ϕ is finally reconstructed via the inverse Fourier transformation

$$\phi = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{\phi}(k) e^{-ikx} dk. \tag{112}$$

In the numerical implementation, the Fourier transformation in Eq. (111) and the inverse transformation in Eq. (112) are discretized by the Discrete Fourier Transformation (DFT), which is further evaluated by using the FFT algorithm (I use the FFTW library). Set up uniform grid-points in x-direction: $x_j = j\Delta$ for $j = 0, 1, 2, ..., N$, as is shown in Fig. (4).

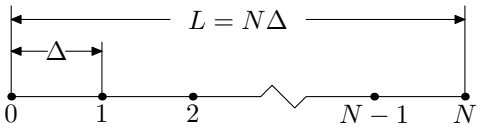

Figure 4. One-dimensional spatial computational box and grids.

Let $\rho_j = \rho(x_j)$ and $\phi_j = \phi(x_j)$. Let $\hat{\rho}_j$ and $\hat{\phi}_j$ denote the corresponding DFT. Using the sampled points ρ_j with $j = 0, 1, 2, ..., N-1$,

we can obtain the DFT $\hat{\rho}_j$. Note that the corresponding wave-number k of $\hat{\rho}_j$ (also for $\hat{\phi}_j$) is given by $k = j2\pi/(N\Delta)$ for $j = 0, 1, ..., N/2$ and $k = (j-N)2\pi/(N\Delta)$ for $j = N/2+1, ..., N-1$ (this corresponds to the negative wave-number part). Use Eq. (110) and the corresponding expression of the wave-number, the discrete form of Eq. (110) is written

$$\hat{\phi}_j = \frac{\hat{\rho}_j}{[j2\pi/(N\Delta)]^2}. \tag{113}$$

for $j = 1, 2, ..., N/2$, and

$$\hat{\phi}_j = \frac{\hat{\rho}_j}{[(j-N)2\pi/(N\Delta)]^2} \tag{114}$$

for $j = N/2+1, N/2+2, ..., N-1$. The $j = 0$ case is a special one because in this case $k = 0$ and k appears in the denominator of (110). Since the overall charge neutrality $\int_{-\infty}^{\infty} \rho dx = 0$ implies $\hat{\rho}_0 = 0$. we usually set $\hat{\phi}_0 = 0$. After obtaining $\hat{\phi}_j$ with $j = 0, 1, ..., N-1$, we can obtain ϕ_j through the inverse DFT.

Knowing the electron potential ϕ_j, the electric field is obtained through the following central difference scheme

$$E_j = -\left.\frac{d\phi}{dx}\right|_{x=x_j} = -\frac{\phi_{j+1} - \phi_{j-1}}{2\Delta}. \tag{115}$$

The electric field at the boundary points are obtained by using the periodic boundary conditions of ϕ.

In the above we use Fourier transformation method to get the electric potential and then use finite difference scheme to calculate the electric field. This is a mixed way to calculate the electric field. We can use only Fourier transformation method to solve for the electric field. In terms of the electric field, Poisson equation (101) is written

$$\frac{d\,\overline{E}}{d\,\overline{x}} = \overline{\rho}. \tag{116}$$

For notation simplicity, omit the over-bar on variables, the above equation is written

$$\frac{dE}{dx} = \rho. \tag{117}$$

The Fourier transformation of the left-hand side of the above equation is written

$$\int_{-\infty}^{\infty} \frac{dE}{dx} e^{ikx} dx \;=\; \int_{-\infty}^{\infty} e^{ikx} dE$$

$$\begin{aligned} &= Ee^{ikx}|_{-\infty}^{+\infty} - ik\int_{-\infty}^{\infty} Ee^{ikx}dx \\ &= -ik\int_{-\infty}^{\infty} Ee^{ikx}dx \\ &= -ik\hat{E}, \end{aligned} \tag{118}$$

where $\hat{E}$ is the Fourier transformation of E. Using this, the Fourier transformation of Eq. (117) is written

$$\hat{E} = \frac{\hat{\rho}}{-ik}, \tag{119}$$

The discrete form of Eq. (119) is similar to the form given in Eqs. (113) and (114), i.e.,

$$\hat{E}_j = \frac{\hat{\rho}_j}{-i[j2\pi/(N\Delta)]}. \tag{120}$$

for $j = 1, 2, ..., N/2$, and

$$\hat{E}_j = \frac{\hat{\rho}_j}{-i[(j-N)2\pi/(N\Delta)]} \tag{121}$$

for $j = N/2+1, N/2+2, ..., N-1$.

5.10 Finite difference solver for Poisson equation

Using the center difference scheme for the second order derivative, the discrete form of Eq. (108) is written

$$\phi_{i-1} - 2\phi_i + \phi_{i+1} = -\Delta^2\rho_i \tag{122}$$

Using the boundary condition $\phi_0 = \phi_{N-1} = 0$, equation (122) is written in the following tridiagonal matrix form:

$$A\phi = b \tag{123}$$

where

$$\mathbf{A} = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{pmatrix}, b = -\Delta^2 \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_{N-3} \\ \rho_{N-2} \end{pmatrix} \tag{124}$$

The results presented in this note are obtained by using the FFT solver, instead of the finite difference solver.

5.11 Interpolate the field to particle markers

As discussed in Sec. 3, for the spatial shape of markers given by the

zero-order b-spline function b_0, the corresponding interpolate function is b_1, which corresponds to a simple linear interpolation. Suppose the location of a marker, r_p, is between x_j and x_{j+1}, then the electric field on the marker is given by

$$E_p = E_j \frac{x_{j+1} - r_p}{\Delta} + E_{j+1} \frac{r_p - x_j}{\Delta}. \tag{125}$$

5.12 Integration of orbit and weight of markers

The evolution equation of orbit and weight can be generally written as

$$\frac{dv}{dt} = H_1(r, v, E), \tag{126}$$

$$\frac{dw}{dt} = H_2(r, v, E), \tag{127}$$

where H_1 and H_2 are known function. Note that E, as well as r and v, depends on time t. However E is not specified as an evolution equation. Instead, E is determined by a field equation, namely Poisson's equation.

The classical 4th Runge-Kutta time integrator requires four evaluations of the function appearing on the right-hand side of the equation of motion per time step. In PIC method, this corresponds to that the field equation needs to be solved for four times. For large-scale simulation (e.g. spatial three-dimension simulation), solving the field equation is usually time-consuming. Considering this, lower order Runge-Kutta (e.g. 2nd order), which requires fewer times of evaluation of functions (and thus fewer times of solving the field equation) may be preferred in practice.

We use the 2nd Runge-Kutta method to integrate orbit and weight of markers. In this method, r, v, and w are first integrated from t_n to the half time-step $t_{n+1/2}$ using (r_n, v_n, E_n) to evaluate the right-hand side of Eqs. (126) and (127). Then we solve the Poisson's equation at $t_{n+1/2}$ to obtain $E_{n+1/2}$ using the already obtained values of ($r_{n+1/2}$, $v_{n+1/2}$, $w_{n+1/2}$). After $E_{n+1/2}$ is obtained, r, v, and w are integrated from t_n to t_{n+1} by using the values at the half time-step, namely ($r_{n+1/2}$, $v_{n+1/2}$, $w_{n+1/2}$, $E_{n+1/2}$). Forgetting to solve the field equation at $t_{n+1/2}$ to get $E_{n+1/2}$ is one of the mistakes I made during writing the code. Forgetting to do this means that I am still using E_n, instead of $E_{n+1/2}$, in taking the full step from t_n to t_{n+1}, which amounts to the (inaccurate and thus may be unstable) Euler scheme.

[Besides, the leapfrog scheme (2nd order accuracy) is often adopted

to integrate the equations of motion. This scheme is given by

$$v^{(i+1/2)} = v^{(i-1/2)} + a^{(i)}dt, \tag{128}$$

$$x^{(i+1)} = x^{(i)} + v^{(i+1/2)}dt \tag{129}$$

where $a_i = a_i(x_i)$ is the acceleration of the particle at the time step i. The leapfrog scheme given by Eqs. (129) and (128) can be equivalently written as[6]

$$x^{(i+1)} = x^{(i)} + v^{(i)}dt + \frac{1}{2}a^{(i)}dt^2 \tag{130}$$

$$v^{(i+1)} = v^{(i)} + \frac{1}{2}(a^{(i)} + a^{(i+1)})dt, \tag{131}$$

The leapfrog scheme given by Eqs. (130) and (131) was implemented in my code, but was not tested seriously (I usually used the 2n Runge-Kutta method discussed above). Note that, for electrostatic case, $a^{(i+1)}$ is independent of $v^{(i+1)}$ and depends only on $x^{(i+1)}$, which is already obtained by Eq. (130). Thus the scheme given by Eqs. (130) and (131) is still an explicit scheme.]

5.13 Initial perturbations

In the electrostatic particle simulation, the electric field is determined by the particle distribution and thus no initial condition for the electric field is needed. The equilibrium distribution function can be chosen to various forms to investigate different physical problems. In this note, a Maxwellian distribution and a two-stream Maxwellian distribution will be chosen to demonstrate the Landau damping and the two-stream instability, respectively.

In the total-f simulation, the noise associated with the initial sampling of the phase space can provide initial perturbations for some physical instabilities (e.g. two-stream instability) to develop from the equilibrium state. In this case, we do not need to impose perturbation manually. However, for other cases where instability is weak or the perturbations are damping (e.g. Landau damping), manual perturbation to the particle weight is needed.

In the δf simulation, we do need to manually impose perturbation on the equilibrium state. i.e. set δf to nonzero value. Otherwise δf will be always zero.

5.14 Verification of the code by using analytic results of Landau damping

One advantage of δf simulation is that nonlinear effects can be readily

turned off by setting the particle orbits to the unperturbed orbits (orbits in the equilibrium field), so that the simulation results can be compared with analytic results obtained in the linear case. Choose Maxwellian distribution as the equilibrium velocity distribution function:

$$F_0(x,v_x) = F_0(v_x) = \frac{n_{e0}}{\sqrt{\pi}v_t}\exp\left(-\frac{v_x^2}{v_t^2}\right). \tag{132}$$

In order to impose a single-k (spatial wavenumber) density perturbation, we set the initial value of $\delta\overline{F}$ as

$$\delta\overline{F}(x,v_x) = 0.001\sin(kx)\frac{1}{\sqrt{\pi}}\exp\left(-\frac{v_x^2}{v_t^2}\right). \tag{133}$$

This perturbation will excite an electrostatic wave and this wave will be damped by a collisionless damping mechanism called Landau damping. Fig. 5 compares the analytic results of Landau damping with those of the linear δf simulation, which shows good agreement between each other in both the frequency and the damping rate.

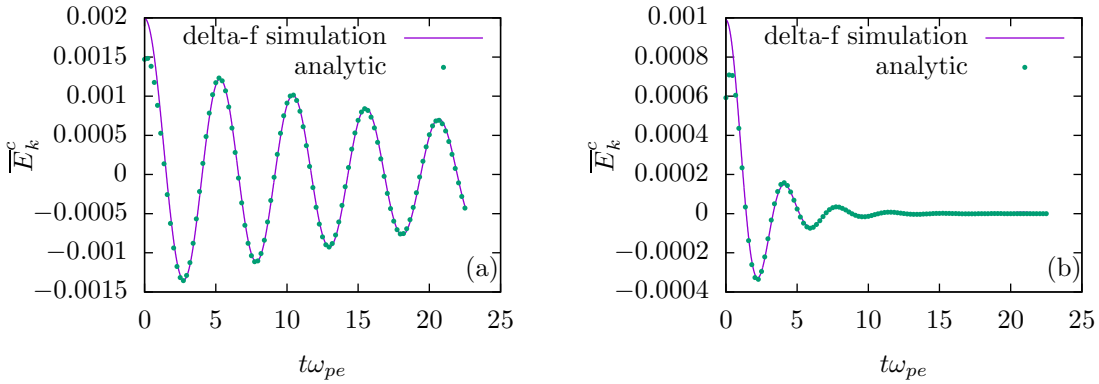

Figure 5. Time evolution of the Fourier cosine component of the electrical field $\overline{E}_k^{(c)}$ for (a) $k = 8\times 2\pi/L$ and (b) $k = 16\times 2\pi/L$. Initial value of the δf is set as $\delta f = 0.001\sin(kx)\exp(-v^2/v_t^2)/\sqrt{\pi}$. Parameters used in the δf simulations are $d\bar{t} = 0.0125$, $L/\lambda_D = 100$, $dx/\lambda_D = 0.25$, and $N = 2\times 10^5$. Uniform random sampling of the phase space is used. The difference between the linear and nonlinear δf simulations is negligible and only linear δf simulation results are plotted here. The analytic frequency and damping rate are obtained by numerically solving the electron plasma wave dispersion relation, which is given by $1 + 2\left(\frac{\omega_p}{kv_t}\right)^2[1 + \zeta Z(\zeta)] = 0$, where $\zeta = \omega/kv_t$, $v_t = \sqrt{2T_e/m_e}$, and $Z(\zeta) = \frac{1}{\sqrt{\pi}}\int_C \frac{e^{-z^2}}{z-\zeta}dz$ is the plasma dispersion function[5].

The initial perturbation $\delta\overline{F}$ given by Eq. (133) are carried equally by the right-going and left-going particles. As a result of this, the electron plasma wave excited in the simulation is always a standing-wave (a

standing wave is composed of two waves with the same frequency and wave-number but opposite propagation directions). Figure 6 plots the spatial structure of the electrical field $\overline{E}_x$ at four successive time, which clearly shows that the wave excited is a standing wave.

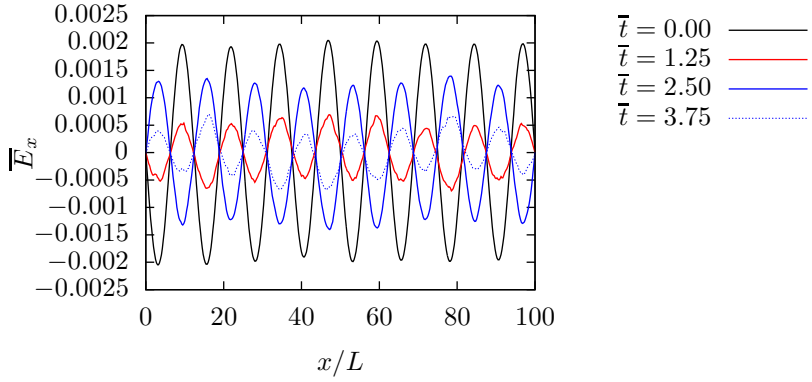

Figure 6. Spatial structure of the electric field at four successive time in the δf simulation with the initial perturbation given by $\delta f = 0.001\cos(kx)\exp(-v^2/v_t^2)/\sqrt{\pi}$ with $k = 8 \times 2\pi/L$. Other parameters are $d\,\bar{t} = 0.0125$, $L/\lambda_D = 100$, and $dx/\lambda_D = 0.25$; $N = 2 \times 10^5$ markers with Maxwellian distribution in velocity and uniform distribution in space are loaded.

To excite a right-going wave, instead of a standing-wave, we can set the initial value of $\delta\overline{F}$ as

$$\delta\overline{F} = \begin{cases} 2 \times 0.001\sin(kx)\dfrac{1}{\sqrt{\pi}}\exp\left(-\dfrac{v_x^2}{v_t^2}\right), & \text{for} \quad v_x > 0 \\ 0, & \text{for} \quad v_x < 0 \end{cases}. \tag{134}$$

This initial perturbation is not symmetric about v_x and thus will carry electric current. Unless otherwise specified, the remainder of this note will consider only the symmetrical perturbation of the form (133).

5.15 Methods of identifying resonant particles

Analytically, resonant particles are defined as those particles whose

velocity is close to the phase-velocity of the wave. These particles are expected to exchange more energy with the waves compared with the non-resonant particles. Next, we examine the phase-space structure of $\delta\overline{F}$ in order to find a general way of identifying the resonant region in the phase-space. The initial phase-space structure of $\delta\overline{F}$ is plotted in Fig. 7a, which shows the fluctuation in x direction and Maxwellian distribution in v_x direction. Figure 7b plots the phase-space structure of $\delta\overline{F}$ at $t = 20/\omega_{pe}$. It is not obvious what kind of useful information can be obtained from the figure. Note that lower velocity particles carry more perturbation than higher velocity particles because of the $\exp(-v^2/v_t^2)$ dependence in $\delta\overline{F}$. The dominant structure of $\delta\overline{F}$ in the lower velocity region may blur the change of $\delta\overline{F}$ in the higher velocity region. To make the change of $\delta\overline{F}$ obvious, define a new function $S(v, x) \equiv \delta\overline{F}/[\exp(-v^2/v_t^2)/\sqrt{\pi}]$, which eliminate the initial variation of $\delta\overline{F}$ in v_x direction. Figure 8 plots the contour of $S(v, x)$ in phase space (x, v) at $t = 0$ and $t = 20/\omega_{pe}$, which shows that there are peaks developing near $v \approx \pm 2.44$ at $t = 20/\omega_{pe}$. The location of the peaks of S in the phase-space, $v \approx \pm 2.44$, is very near the phase-velocity of the wave

excited in the simulation ($v_p/v_t = \omega/kv_t = \pm 2.44$). Therefore, the peaks of S prove to be a good indication of the resonant region.

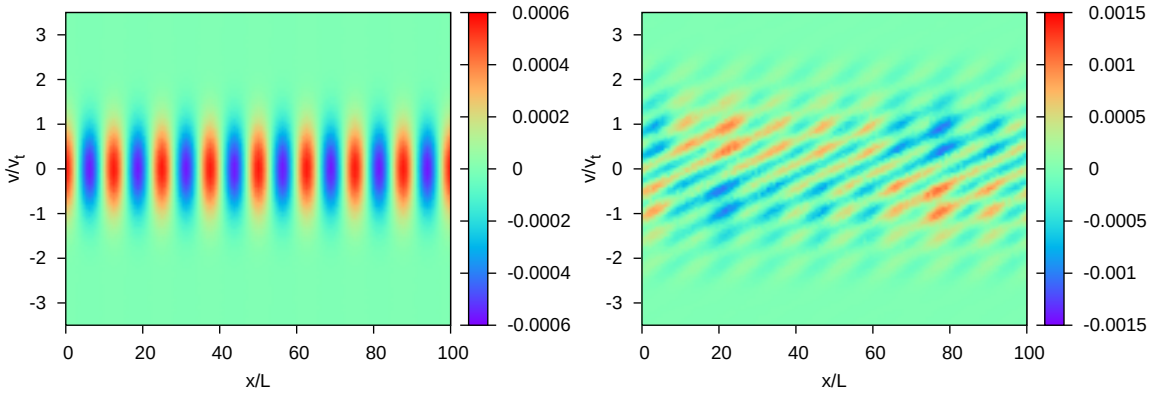

Figure 7. Contour of $\delta\bar{F}$ in phase-space (x, v) at $t = 0$ (left figure) and $t = 20/\omega_{pe}$ (right figure) in the δf simulation with uniform sampling of phase-space.. Initial value of $\delta\bar{F}$ is set as $\delta\bar{F} = 0.001\cos(kx)\exp(-v^2/v_t^2)/\sqrt{\pi}$ with $k = 8 \times 2\pi/L$.

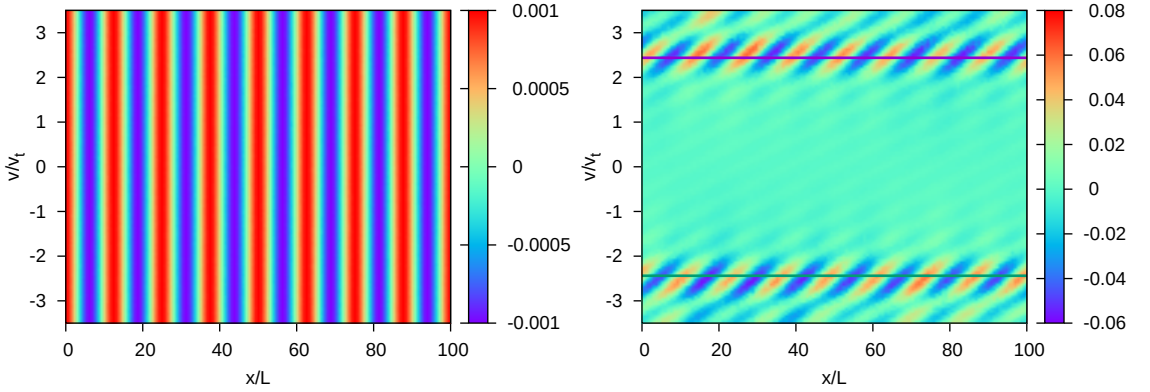

Figure 8. Contour of $S(x, v_x) \equiv \delta\bar{F}/[\exp(-v^2/v_t^2)/\sqrt{\pi}]$ at $t = 0$ (left figure) and $t = 20/\omega_{pe}$ (right figure) in the δf simulation with uniform sampling of phase-space. Initial value of $\delta\bar{F}$ is set as $\delta\bar{F} = 0.001\cos(kx)\exp(-v^2/v_t^2)/\sqrt{\pi}$ with $k = 8 \times 2\pi/L$. The solid lines on the figure indicate the phase-velocity of the wave excited in the simulation ($v_p/v_t = \omega/kv_t = \pm 2.44$). Other parameters used in the simulation are $d\bar{t} = 0.0125$, $L/\lambda_D = 100$, $dx/\lambda_D = 0.25$, $N = 2 \times 10^5$, and $[v_{\min}/v_t, v_{\max}/v_t] = [-3.5, +3.5]$.

We select the top 500 markers that have large variation in δf /

$[\exp(-v^2 \,/\, v_t^2) \,/\, \sqrt{\pi}]$ and then compare their velocity with the phase-velocity of the wave. The results are plotted in Fig. 9, which confirms that these velocities are close to the phase-velocity of the wave. Note that, since the wave excited in the simulation is a standing-wave, which has two opposite phase-velocities, the corresponding resonant velocity also have two opposite values.

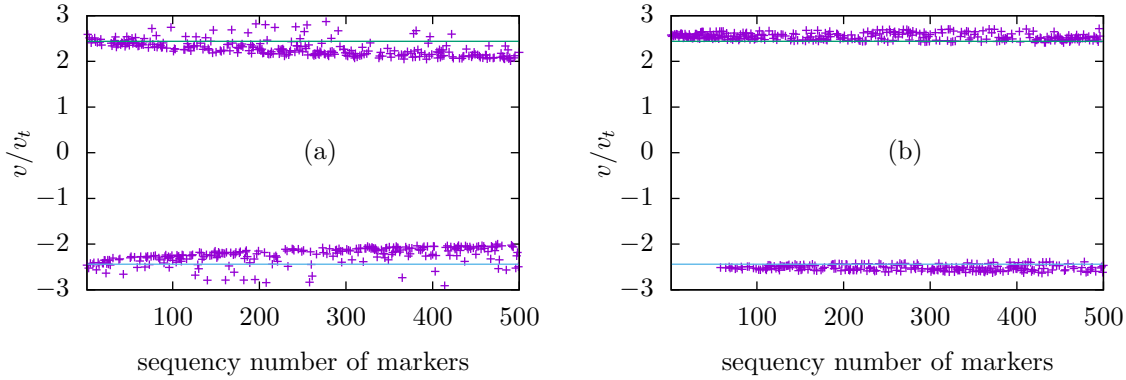

Figure 9. Velocity of the top 500 particles that have large variation in $\delta f \,/\, [\exp(-v^2 \,/\, v_t^2) \,/\, \sqrt{\pi}]$ between $t = 0$ and $t = 20 \,/\, \omega_{pe}$ in the δf simulation with Maxwellian velocity sampling (a) and uniform velocity sampling (b). Initial value of the δf is set as $\delta f = 0.001 \cos(kx) \exp(-v^2 / v_t^2) / \sqrt{\pi}$ with $k = 8 \times 2\pi \,/\, L$. The markers are ordered according to their magnitude of the variation in $\delta f \,/\, [\exp(-v^2 \,/\, v_t^2) \,/\, \sqrt{\pi}]$. The first marker has the largest variation. The solid lines on the figure indicate the phase-velocity of the wave excited in the simulation ($v_p / v_t = \omega / k v_t = \pm 2.44$). Other parameters used in the simulation are $d\bar{t} = 0.0125$, $L / \lambda_D = 100$, $dx / \lambda_D = 0.25$, $N = 2 \times 10^5$, and $[v_{\min} / v_t, v_{\max} / v_t] = [-3.5, +3.5]$; spatial sampling is uniform.

There is difference between Fig. 9a and Fig. 9b, which arises from the different sampling of the velocity space. In Fig. 9b, we note that the top 50 resonant particles all have positive velocity, which is non-physical because there is no preferred direction in the system with a standing wave and symmetric velocity distribution.

5.16 Energy conservation (check!)

Next we check how well the total energy of the system is conserved in a

total-f simulation. The total physical particles in the system is given by

$$N_s = \sum_{j=0}^{N_p} w_j, \tag{135}$$

The spatial volume occupied by these physical particles is given by $V = N_s / n_0$, where n_0 is the equilibrium electron number density. Since the length along the x direction of the system is L, the cross section S_{yz} of volume occupied by these physical particles is given by

$$S_{yz} = \frac{V}{L} = \frac{\sum_{j=0}^{N_p} w_j}{L n_0}. \tag{136}$$

Then the total electrical energy in the volume is given by

$$W_E = S_{yz} \int_0^{L_x} \frac{1}{2} \varepsilon_0 E^2 dx \approx S_{yz} \sum_{i=1}^{n} \frac{1}{2} \varepsilon_0 E^2(x_i)\, \Delta \tag{137}$$

Define $W_0 = (m v_0^2 / 2) \sum w_j$, and the normalized electric energy $\overline{W}_E = W_E / W_0$, which can be further written as

$$\overline{W}_E = \frac{\sum_{j=0}^{N} w_j}{n_{e0} L} \frac{\sum_{i=1}^{n} \frac{1}{2} \varepsilon_0 E^2(x_i) \Delta}{(m v_0^2 / 2) \sum w_j} = \frac{\sum_{i=1}^{n} \frac{1}{2} \varepsilon_0 E^2(x_i)\, dx}{n_{e0} L (m v_0^2 / 2)} = \frac{\sum_{i=1}^{n} \frac{1}{2} \varepsilon_0 \overline{E}_i^2\, dx}{n_{e0} L (m v_0^2 / 2)} \left(\frac{m v_0^2}{e \lambda_D} \right)^2 = \frac{\sum_{i=1}^{n} \overline{E}_i^2\, d\overline{x}}{\overline{L}}. \tag{138}$$

The total particle kinetic energy in the system is given by

$$W_k = \sum_{j=0}^{N_p} w_j \frac{1}{2} m v_j^2. \tag{139}$$

Define the normalized kinetic energy $\overline{W}_k = W_k / W_0$, which can be further written as

$$\overline{W}_k = \frac{\sum_{j=0}^{N} w_j \frac{1}{2} m v_j^2}{(m v_0^2 / 2) \sum w_j} = \frac{\sum_{j=0}^{N} w_j \overline{v}_j^2}{\sum w_j}. \tag{140}$$

Figure 10 plots the time evolution of $\overline{W}_E$, $\overline{W}_k - \overline{W}_k(t=0)$, and $\overline{W}_k + \overline{W}_E - \overline{W}_k(t=0)$, which indicates the total energy is approximately conserved.

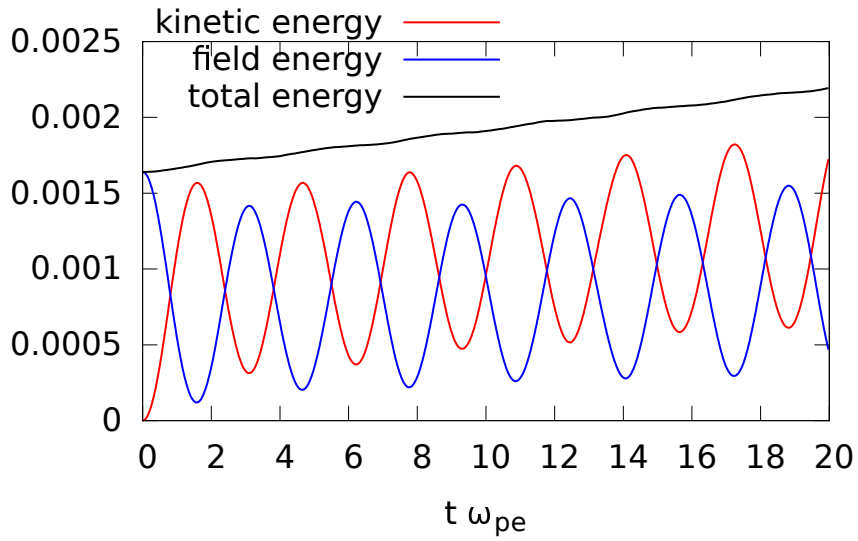

Figure 10. Time evolution of the electric energy $\overline{W}_E$, kinetic energy $\overline{W}_k$, and total energy $\overline{W}_k + \overline{W}_E - \overline{W}_k(t=0)$. $\overline{W}_k(t=0) = 0.9991$. Full-f simulation without imposing external perturbation.

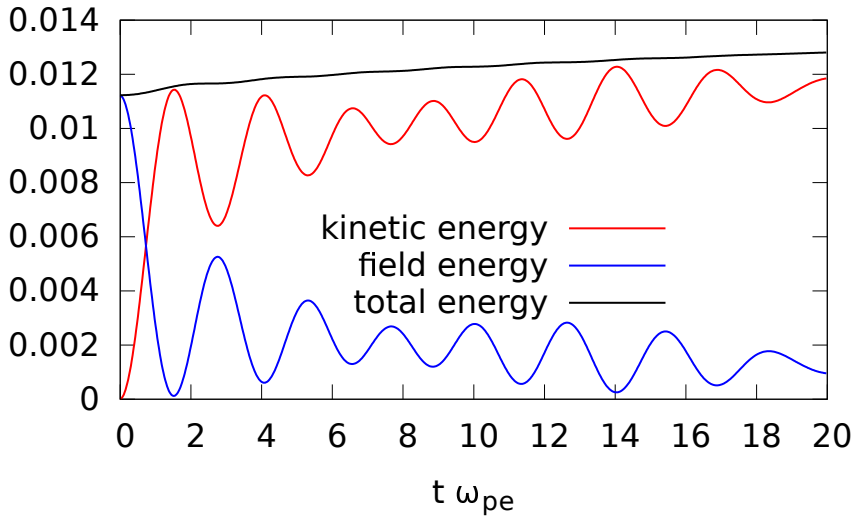

Figure 11. Time evolution of the electric energy $\overline{W}_E$, kinetic energy $\overline{W}_k$, and total energy $\overline{W}_k + \overline{W}_E - \overline{W}_k(t=0)$. Full-f simulation with perturbation $w_j \longrightarrow w_j + 0.05 w_j \sin(k r_j)$

5.17 Numerical results for two-stream instability

Choose an equilibrium distribution function $F_0(x,\, v_x)$ of the following form:

$$F_0(x,\quad v_x) \quad = \quad F_0(v_x) \quad = \quad \frac{n_{e0}}{2}\left\{\frac{1}{\sqrt{2\pi}\,v_t}\exp\left(-\frac{(v-v_b)^2}{2\,v_t^2}\right) \quad + \right.$$
$$\left.\frac{1}{\sqrt{2\pi}\,v_t}\exp\left(-\frac{(v+v_b)^2}{2\,v_t^2}\right)\right\}, \tag{141}$$

where n_{e0} is a constant which is chosen so that $n_{e0} = n_{\rm ion}$. It is obvious

that this initial condition corresponds to an equilibrium state. It is well-known that when $v_b > v_t$, this equilibrium is unstable to an instability called the two-stream instability, which destroys the equilibrium state. In practice, the numerical noise associated with the PIC method is usually large enough to provide the initial perturbation to make this instability grow up. Thus, to see the instability, we usually do not need to manually impose any perturbation to the equilibrium. Figure 12 plots the distribution of the electron makers in the phase space (x, v) at $\bar{t} = 0$ and $\bar{t} = 17.5$ in a full-f simulation. Every particle marker appears as a black dot on Figure 12. Note that, since this is a full-f simulation and the markers are loaded according to the initial distribution function, statistical weights of all the marker are equal to each other and remain constant during the time evolution. Therefore more markers means more real particles. And since every particle marker appears as a black dot on Figure 12, region with denser markers appears blacker. Thus the graphics in Figure 12 can be considered as contour plots of the distribution function with the brightness indicating the value (blacker meaning higher value).

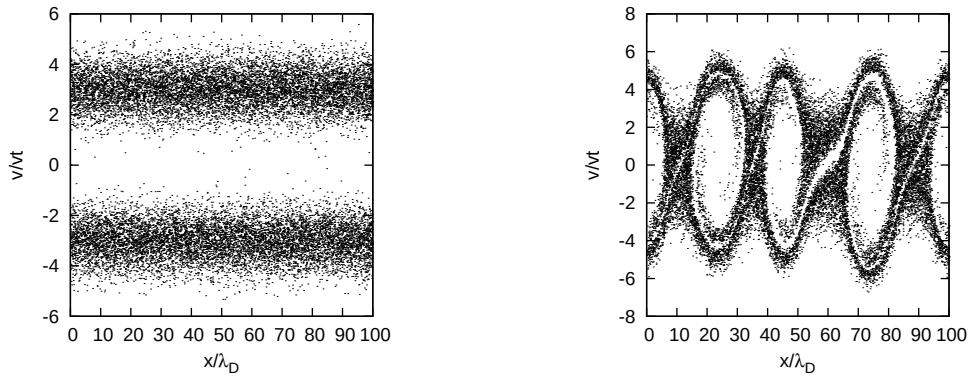

Figure 12. Electron distribution function in the phase-space evaluated at $t = 0$ and $\bar{t} = 17.5$ for a one-dimensional electrostatic simulation of the two-stream instability performed with $N_{\text{loaded}} = 2 \times 10^4$, $N = 1024$, $L/\lambda_D = 100$, $v_b/v_t = 3$, and $\delta t \omega_p = 0.1$.

to be continued.

My Fortran code solving the two-stream instability problem is in the directory /home/yj/project_new/pic_full-f/ of my computer.

6 Summary

In summary, PIC = random sample of phase space (Monte-Carlo integration) + particle spatial shape + field solver + characteristics (particle orbits) and/or marker's weight integrator.

7 Random number

7.1 Uniformly distributed random number

Generating random numbers that are uniform distributed in the range [0, 1] is the basis for generating non-uniform distribution. Because the same program with the same input always produces the same output, it is not possible to write a program that produces truly random numbers. However, for most purposes, a pseudo-random number sequence will work almost as well. By "pseudo-random number", we mean a repeatable sequence of numbers that has statistical properties similar to a random sequence. The most well-known algorithm for generating pseudo-random sequences of integers is the linear congruental method[4], in which the nth and $(n + 1)$th integers in the sequence is related by

$$I_{n+1} = \mathrm{Mod}(AI_n + C, M), \tag{142}$$

where Mod is the remainder function, A, C, and M are positive integer constants. The first number in the sequence, which is called the seed value, is selected by users. Equation (142) can generate pseudo-random number that is uniform distributed in the range $[0, M - 1]$. The obtained sequence can be scaled by a factor of $M - 1$ to lie in the range [0, 1]. Figure 13 plots the possibility density of 10^6 values returned by Eq. (142) with parameters $A = 16807$,$C = 0$, $M = 2147483647$ (this choice is called the Park and Miller method). In practice we need to use

Schrange's algorithm to avoid integer overflow[4].

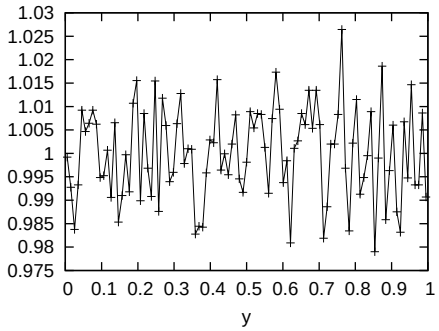

Figure 13. The distribution of the 10^6 values returned by the the random number generator Eq. (142). Plotted is the normalized possibility density, which is obtained by the following steps: (1) divide the range $[0, 1]$ into 100 sub-intervals; (2) then count the number of markers that are within each sub-interval; (3) the value in each sub-interval is then divided by the length of the sub-interval, $1/100$, to get the density. Further divide the density by the total marker number, 10^6, giving the normalized probability density, which satisfies the normalzation condition $\int_0^1 P(x)dx = 1$.

Another way to visualize whether the values generated by the random generator are random distributed in the region $[0, 1]$ is to view how the points (x_j, x_{j+1}) are distributed in the two-dimension plane, as is plotted in Fig. 14.

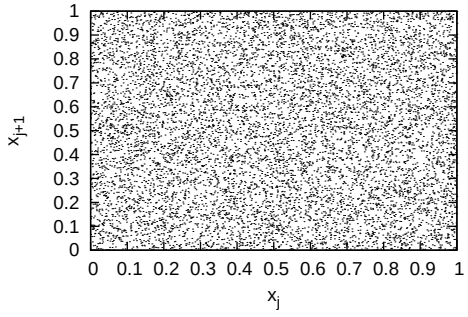

Figure 14. Plot of x_j verse x_{j+1} for $j = 1, 2, ..., 10^4$. Here x_j are random numbers generated by the random number generator.

7.2 Non-uniformly distributed random number

There are two methods of generating non-uniformly distributed random numbers satisfying a given distribution function, namely, the transformation method and the rejection method[4]. Let us examine these two methods in turn.

7.2.1 Transformation method

The probability density $P(x)$ is defined by that $P(x)dx$ represents the probability of finding the random variable x in the interval $[x, x + dx]$.

Suppose there are two random variables y and x that are related to each other by $y = f(x)$. If the probability density of x, $P_x(x)$, is known, how do we calculate the probability of the random variable y? Using the probability conservation, i.e,

$$P_y(y)|dy| = P_x(x)|dx|, \tag{143}$$

we obtain

$$P_y(y) = \frac{P_x(x)}{|f'(x)|}, \tag{144}$$

which gives the relation between $P_y(y)$ and $P_x(x)$. Next, consider the inverse problem of the above, i.e., if we want to generate non-uniform distribute random numbers y with probability density being $P_y(y)$ from a uniformly distributed random variables x in [0: 1], how do we choose the function $f(x)$? In this case, $P_x(x) = 1$ and Eq. (144) is written

$$P_y(f(x)) = \frac{1}{|f'(x)|}, \tag{145}$$

which can be solved to give $f(x)$. For a general function $P_y(y)$, Eq. (145) may not be analytically solvable. For the special case $P_y(y) = e^{-y}$ (Poisson distribution), we find that $f(x) = -\ln x$ solves Eq. (145). Therefore, we can generate Poisson distribution by the following Fortran codes:

```
call random_number(x) !generate uniform random numbers in
```

```
(0:1]
y=-log(x)
```

The transformation method requires explicit form of $f(x)$ be known, which is not always practical for a general probability density $P_y(y)$. In such cases, we can use the rejection method discussed next.

In PIC simulations, we often want to load markers as Maxwellian distribution in the velocity space, i.e., the normal or Gaussian distribution. It's not easy to find a tranform function that can transform a uniform distribution to a normal distribution. But some guys had found some ways to do this tranform. One of the methods is the Box-Muller algorithm: you generate two values from a uniform random $(0, 1]$ generator, and then combine them to get two values of a standard normal distribution:

```
call random_number(u1)
call random_number(u2)
z1 = sqrt(-2.0 * log(u1)) * cos(twopi * u2)
z2 = sqrt(-2.0 * log(u1)) * sin(twopi * u2)
```

Then you can scale and shift the values to get normal distribution of arbitrary mean μ and standard deviation σ:

$$Z = \sigma z_1 + \mu. \tag{146}$$

7.2.2 Rejection method

Rejection sampling is based on the observation that to sample a random variable in one dimension, one can perform a uniformly random sampling of the two-dimensional Cartesian graph, and keep the samples in the region under the graph of its density function.

Note that this property can be extended to N-dimension functions.

Suppose that we want to generate non-uniformly distributed random numbers between $x_{\min}$ and $x_{\max}$ with a given probability density $P(x)$. We first generate a uniform random number x_t between $x_{\min}$ and $x_{\max}$. Then we generate another uniform random number P_t between 0 and $P_{\max}$, where $P_{\max}$ is the maximal value of $P(x)$ for $x \in [x_{\min}, x_{\max}]$. If $P(x_t) \geqslant P_t$ then x_t is accepted as a sampling, otherwise x_t is rejected.

Repeat this process, then all the random numbers kept will satisfy the probability density $P(x)$ (need thinking why, to be proved). This method is called the rejection sampling. It is obvious how the rejection method generalizes to multiple-dimensional cases. Also note that the algorithm can be used to sample from a distribution whose normalizing constant is unknown.

Next is a numerical example. Consider the one-dimensional Gaussian distribution given by

$$P(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y-\bar{y})^2}{2\sigma^2} \right). \tag{147}$$

We use the rejection method to generate an assemble of values of y that satisfies the above probability distribution. Figure 15 compares the possibility density of the 10^6 numbers generated by the numerical code with that of the analytic form in Eq. (147), which indicates that the numerical result agrees well with the analytic one.

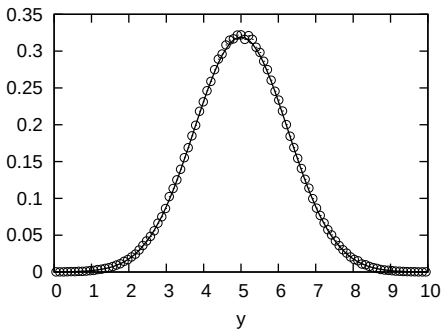

Figure 15. The distribution of the 10^6 values returned by the the Gaussian distribution generator (using the rejection method) with the parameters $\bar{y} = 5.0$ and $\sigma = 1.25$. The possibility density (value of the distribution function) is obtained as follows: (1) divide the range [0, 10] into 100 sub-regions (2) then counts respectively the number of the returned value whose values are in the sub-regions (3) the numbers of value in each sub-region obtained this way is further divided by the total number of values (10^6) to give the relativistic possibilities. (4) scale the relativistic possibilities by 10 times, which gives the exact possibility density (this scaling is needed because the sub-region is of length $1/10$, instead of unit length). The solid line in the figure is the value obtained by evaluating Eq. (147). The results indicates that the distribution returned by the Gaussian generator agrees well with the desired theoretic one.

Figure 16 is a plot of x_j verse x_{j+1} for $j = 1, 2, ..., 10^4$, which shows how the points (x_j, x_{j+1}) are distributed in the two-dimension plane.

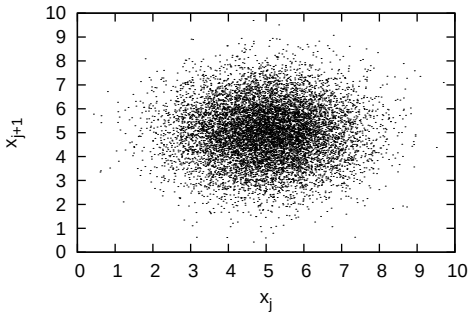

Figure 16. Plot of x_j verse x_{j+1} for $j = 1, 2, ..., 10^4$. Here x_j are random numbers generated by the Gaussian distribution generator.

random vs grid

Numerical integration using random samping is more accurate than grid-based integration method for dimensionality larger than two. One observation that can help one understand the reason is as follows. For a fixed number of sampling points, N, random sampling always has N samplings along each dimension, while for grid-based methods, the resolution along each dimension is given by $N^{1/d}$, which decreases with increase in dimensionality.

Figure 17 compares the grid sampling and the random sampling. Which one is better in terms of resolution? This depends on the function to be sampled. If the function is uniform in (x, y), then whatever sampling will give the same result. If the function depends on only x or y, then method 2 is beter than method 1 because it has more sampling points along each dimension. Method 2 chooses y according to the value of x, i.e., $y = x$. This makes y corelate with x and thus limits its adaptation to resovle general dependence of the function on x and y. For a general function, whose dependence on x and y is unknown, we expect that random sampling is more robust (i.e., give reasonable results in more cases than grid sampling.)

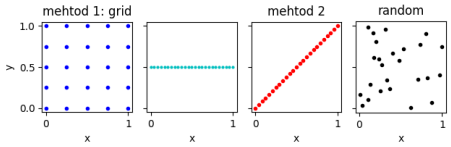

Figure 17. Samping x-y space using grid and random sampling. 5×5 grid (left) vs. 25 regular sampling (middle), vs. 25 uniformly random sampling (right).

—tmp–

$$f = f_0 + \delta f \tag{148}$$

$$\frac{d\delta f}{dt} = -\frac{df_0}{dt} \tag{149}$$

$$\frac{d\delta f/f}{dt} = -\frac{1}{f}\frac{df_0}{dt} \tag{150}$$

$$\frac{d\delta f/f}{dt} = -\frac{f_0}{f}\frac{1}{f_0}\frac{df_0}{dt} \tag{151}$$

$$\frac{dw}{dt} = -\frac{f-\delta f}{f}\frac{1}{f_0}\frac{df_0}{dt} \tag{152}$$

$$\frac{dw}{dt} = -(1-w)\frac{1}{f_0}\frac{df_0}{dt} \tag{153}$$

to be deleted–

A From discrete microscopic distribution function to statistic (continuum) distribution function

A plasma can be considered to be composed of a set of classical point particles, with motion subject to Newton's equations and with the Lorentz forces and electromagnetic field derived from Maxwell's equations. Because of the huge number of particles in a plasma, the above

microscopic representation is intractable, and simplifications must be sought[8].

A usual procedure is to approximate the set of particles by a continuum distribution function. This step involves some kind of averaging procedure to remove certain spatial and temporal frequencies that are associated with the graininess of the particle description (particle corelations, i.e., collisions). It is important to note that approximation is introduced in passing from the microscopic particle representation to the continuum distribution function.

The averaging procedure involves a small parameter, the so-called "plasma parameter," which is the inverse of the number of particles contained in a Debye sphere. The collisionless Boltzmann equation, also known as the Vlasov equation, emerges from the averaging procedure as the approximation at zero order in the plasma parameter. What is discarded in the the zeroth order approximation is the so-called collisional effects. In the first order approximation, there appear additional terms, which is a representation of the Coulomb collision effects.

The discrete microscopic distribution function (Klimontovich-Dupree distribution function) (Chapter 3 in Nicholson (1983)) is written as

$$F_\alpha^M(\mathbf{x},\mathbf{v},t) = \sum_{p_\alpha=1}^{N_\alpha} \delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})\delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha}). \tag{154}$$

Then the partial time derivative of F_α^M is written as

$$\begin{aligned}\frac{\partial F_\alpha^M(\mathbf{x},\mathbf{v},t)}{\partial t} &= \sum_{p_\alpha=1}^{N_\alpha} \delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha})\frac{\partial}{\partial t}\delta_x^3(\mathbf{x}-\mathbf{x}_{p_a}) + \sum_{p_\alpha=1}^{N_\alpha} \delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})\frac{\partial}{\partial t}\delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha}) \qquad (155)\\ &= -\sum_{p_\alpha=1}^{N_\alpha} \delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha})\frac{\partial \mathbf{x}_{p_\alpha}}{\partial t}\cdot\nabla_{\mathbf{x}}[\delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})] - \sum_{p_\alpha=1}^{N_\alpha} \delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})\frac{\partial \mathbf{v}_{p_\alpha}}{\partial t}\cdot\nabla_{\mathbf{v}}\delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha})\\ &= -\sum_{p_\alpha=1}^{N_\alpha} \delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha})\mathbf{v}_{p_\alpha}\cdot\nabla_{\mathbf{x}}[\delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})] - \sum_{p_\alpha=1}^{N_\alpha} \delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})\frac{q_\alpha}{m_\alpha}[\mathbf{E}^M(\mathbf{x}_{p_\alpha}) + \mathbf{v}_{p_\alpha}\times\mathbf{B}^M(\mathbf{x}_{p_\alpha})]\cdot\nabla_{\mathbf{v}}\delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha}) \qquad (156)\end{aligned}$$

Using the property of the Dirac delta function:

$$a\delta(a-b)=b\delta(a-b) \tag{157}$$

expression (156) is written as

$$\begin{aligned}\frac{\partial F_\alpha^M(\mathbf{x},\mathbf{v},t)}{\partial t} &= -\sum_{p_\alpha=1}^{N_\alpha}\delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha})\mathbf{v}\cdot\nabla_{\mathbf{x}}[\delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})]-\sum_{p_\alpha=1}^{N_\alpha}\delta_x^3(\mathbf{x}-\\ &\quad \mathbf{x}_{p_a})\frac{q_\alpha}{m_\alpha}[\mathbf{E}^M(\mathbf{x})+\mathbf{v}\times\mathbf{B}^M(\mathbf{x})]\cdot\nabla_{\mathbf{v}}\delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha})\\ &= -\mathbf{v}\cdot\nabla_{\mathbf{x}}\sum_{p_\alpha=1}^{N_\alpha}\delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha})[\delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})]-\frac{q_\alpha}{m_\alpha}[\mathbf{E}^M(\mathbf{x})+\\ &\quad \mathbf{v}\times\mathbf{B}^M(\mathbf{x})]\cdot\nabla_{\mathbf{v}}\sum_{p_\alpha=1}^{N_\alpha}\delta_x^3(\mathbf{x}-\mathbf{x}_{p_a})\delta_v^3(\mathbf{v}-\mathbf{v}_{p_\alpha})\\ &= -\mathbf{v}\cdot F_\alpha^M-\frac{q_\alpha}{m_\alpha}[\mathbf{E}^M(\mathbf{x})+\mathbf{v}\times\mathbf{B}^M(\mathbf{x})]\cdot\nabla_{\mathbf{v}}F_\alpha^M,\end{aligned} \tag{158}$$

i.e.,

$$\frac{\partial F_\alpha^M(\mathbf{x},\mathbf{v},t)}{\partial t}+\mathbf{v}\cdot F_\alpha^M+\frac{q_\alpha}{m_\alpha}[\mathbf{E}^M(\mathbf{x})+\mathbf{v}\times\mathbf{B}^M(\mathbf{x})]\cdot\nabla_{\mathbf{v}}F_\alpha^M=0 \tag{159}$$

This is the Klimontovich equation of the microscopic distribution function.

Note that Boltzmann equation provide a mesoscopic rather than a microscopic description because of the spatio-temporal averaging involved in deriving the Boltzmann equation

The particle-in-cell (PIC) method has obvious structural similarities to a direct simulation of the microscopic model of the plasma outlined above, but this is essentially misleading: Particle-in-cell simulation should be understood to be a Monte Carlo solution of the Boltzmann equation (or the Vlasov equation), i.e., the PIC method is solving a continuum mesoscopic kinetic equation rather than the primitive microscopic model of a plasma.

two things that reduce collision: (1) finite-size particle used in doing the deposition and force iterpolation and (2) finite grid size used in solving the field equation.

The nearest grid point deposition and force interpolation correspond zero sized particle, i.e., delta function

$$S(x)=\frac{1}{\Delta x}\delta\Big(\frac{x}{\Delta x}\Big). \tag{160}$$

Bibliography

[1] S.J. Allfrey and R. Hatzky. A revised delta-f algorithm for nonlinear pic simulation. *Computer Physics Communications*, 154(2):98 – 104, 2003.

[2] A. Y. Aydemir. A unified monte carlo interpretation of particle simulations and applications to non-neutral plasmas. *Physics of Plasmas*, 1(4):822–831, 1994.

[3] Yang Chen and Roscoe B. White. Collisional delta-f method. *Physics of Plasmas*, 4(10):3591–3598, 1997.

[4] Richard Fitzpatrick. *Computational Physics:An introductory course*. Richard Fitzpatrick, 2004.

[5] D. A. Gurnett and A. Bhattacharjee. *Introduction to plasma physics : with space and laboratory applications*. Cambridge University Press, Cambridge, UK, 2004.

[6] Piet Hut and Jun Makino. *http://www.artcompsci.org/vol$_1$/v1$_w$eb/node34.html*. Online, 2012.

[7] Giovanni Lapenta. *Particle In Cell Methods With Application to Simulations in Space Weather*. Online, 2012.

[8] M. M. Turner. Kinetic properties of particle-in-cell simulations compromised by monte carlo collisions. *Physics of Plasmas*, 13(3):033506, 2006.