Particle in Cell (PIC) Method

Giovanni Lapenta

March 30, 2016

Contents

Chapter 1

Particle-based simulation of plasmas

1.1 Types of interacting systems

The first step to decide how to model a system of interacting particles is to distinguish between weakly and strongly interacting systems. The distinction has a profound meaning and a direct impact on how to model the system on a computer.

Figure 1.1 summarises visually the situation. Let us consider a system made of a collection of charged particles in a box with the side of the Debye length, λ_D (the box is 3D but is depicted as 2D for convenience). We choose the Debye length because a basic property of plasmas is to shield the effects of localised charges over distances exceeding the Debye length. Of course the shielding is exponential and the effect is not totally cancelled over one Debye length, but such a length provides a conventional reasonable choice for the interaction range. The electric field in each point of the box is computed by the superposition of the contributions of each particle.

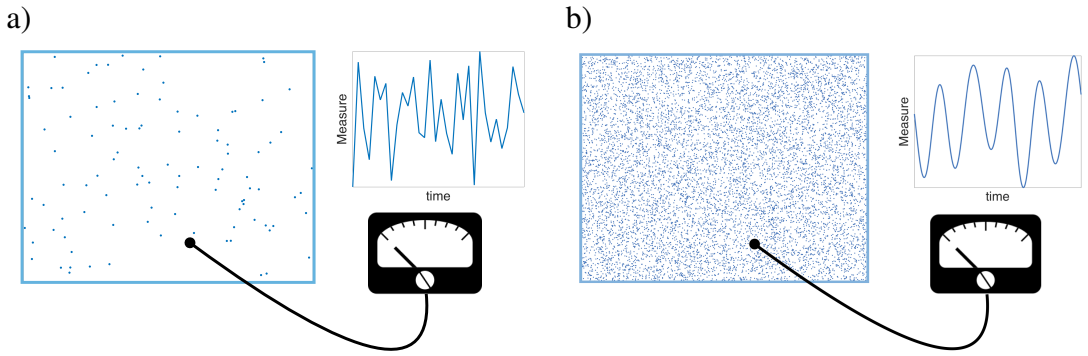

Figure 1.1: Thought experiment for strongly (a, left) versus weakly (b,right) coupled systems.

Let us conduct an ideal thought experiment based on using a device able to detect the local electric field in one spatial position. We identify in the figure such ideal measurement device with an icon connected to a point in the system. We try to conduct a thought experiment where in no step any law of physics is violated but where the difficulties of experimental work are eliminated.

If we consider the configuration in Fig. 1.1-a, we note that within the domain there are few particles and the measurement obtained by our fantastic electric field meter would be very jumpy. The particles in the box move constantly, interacting with each other and agitated by their thermal motion. As a particle passes by the detector, the measurement detects a jump up and when a particle moves away it detects a jump down. On average at any given time very few particles are near the detector and their specific positions are key in determining the value measured. The effect of a given particle on the electric field at the location of measurement decays very rapidly with the distance and only when the particle is nearby the effect is strong.

The same effect is detected by each of the particle in the system. The electric field each particle feels is a sum of the contributions of all others but only when another particle passes by the electric field would register a jump: in common term this event is called a *collision*. The particle trajectories would then be affected by a series of close encounters registered as jumps (collisions) in the trajectory.

The system described goes in the language of kinetic theory by the name of *strongly coupled system*, a system where the evolution is determined by the close encounters and by the relative configuration of any two pairs of particles. The condition just described is characterised by the presence of few particles in the Debye box.

The opposite situation is that of a weakly coupled system. The corresponding configuration is described in Fig. 1.1-b.

Now the system is characterised by being composed by an extremely large number of particles in the Debye box. At any given point, the number of particles contributing to the electric field is very large. Regardless of the particle motion, the field is given by the superposition of many contributions. As a consequence, by simple averaging of the effects of all the particles contributing to the measurement, the measurement is smooth and does not jump in time. Similarly the trajectory of a particle is at any time affected by a large number of other particles. The trajectory is smooth and without jumps. These systems are called *weakly coupled*. If in the strongly coupled system, the characteristic feature was the presence of a succession of collisions, in the weakly coupled system, the characteristic feature is the *mean field* produced by the superposition of contributions from a large number of particles.

1.1.1 Strength of interaction

The discriminant factor in the previous discussion was the number of particles present inside the box under consideration. If we choose the conventional box with side equal to the Debye length, the number of particles present is

$$N_D = n\lambda_D^3 \tag{1.1}$$

where n is the plasma density.

A system is considered weakly coupled when N_D is large and strongly coupled when N_D is small.

This concept can be further elaborated by considering the energies of the particles in the system. The particles in the box are distributed in a non-uniform, random way, but on average, the volume associated with each particle is simply the volume of the box, λ_D^3, divided by the number of particles in the box, N_D. This volume, $V_p = n^{-1}$, can be used to

determine the average interparticle distance, $a = V_p^{1/3} \equiv n^{-1/3}$. This relation provides an average statistical distance. The particles are distributed randomly and their distances are also random, but on average the interparticle distance is a.

The electrostatic potential energy between two particles with separation a is

$$E_{pot} = \frac{q^2}{4\pi\varepsilon_0 a} \tag{1.2}$$

where we have assumed equal charge q for the two particles. Conversely, from statistical physics, the kinetic energy of the particles is of the order of the thermal energy:

$$E_{th} = kT \tag{1.3}$$

where k is the Boltzmann constant.

A useful measure of the plasma coupling is given by the so-called *plasma coupling parameter*, Γ, defined as:

$$\Gamma = \frac{E_{pot}}{E_{th}} = \frac{q^2}{4\pi\varepsilon_0 akT} \tag{1.4}$$

Recalling the definition of Debye length ($\lambda_D = (\varepsilon_0 kT/nq^2)^{1/2}$) and the value of a obtained above, it follows that:

$$\Gamma = \frac{q^2 n^{1/3}}{4\pi\varepsilon_0 kT} \equiv \frac{1}{4\pi N_D^{2/3}} \tag{1.5}$$

The plasma coupling parameter gives a new physical meaning to the number of particles per Debye cube. When many particles are present in the Debye cube, the coupling parameter is small. In this case, the thermal energy far exceed the potential energy, making the trajectory of each particle little influenced by the interactions with the other particles: this is the condition outlined above for the weakly coupled systems. Conversely, when the number of particles per Debye cube is small, the coupling parameter is large, the potential energy dominates and the trajectories are strongly affected by the near neighbour interactions: this is the condition typical of strongly coupled systems.

1.2 Computer simulation of interacting systems

A computer simulation of a system of interacting particles can be done in principle by simply following each particle in the system. The so-called *particle-particle (PP)* approach describes the motion of N particles by evolving the equations of Newton for each of the N particles taking as a force acting on the particle the combined effect of all the other particles in the system.

The evolution is discretised in many temporal steps Δt, each chosen so that the particles move only a small distance. After each move, the force is recomputed and a new move is made for all the particles. The main cost of the effort is the computation of the force which requires to sum over all the particles in the system. Once the force is computed the new velocities can be computed. Then the new positions can be computed and the cycle can be repeated indefinitely.

For each particle, the number of terms to sum to compute the force is $N-1$, and considering that there are N particles, but that each pair needs to be computed only once, the total number of force computations is $N(N-1)/2$.

For strongly coupled systems, where the number of particles per Debye cube is small, the PP approach is feasible and forms the basis of the very successful *molecular dynamics method* used in condensed matter and in biomolecular studies. We refer the reader to a specific textbooks on molecular dynamics to investigate the approach more in depth [22].

The PP approach is also used in the study of gravitational interactions, for example in the cosmological studies of the formation and distribution of galaxies. In that case, specifically the dark matter is studied with a PP approach. The PP approach can be made more efficient by using the *Barnes-Hut* or *tree algorithm* [2] that can reduce the cost (but not without loss of information [60]) to $O(N \log N)$.

Even with the reduced cost of the tree algorithm, PP methods cannot be practical for weakly coupled systems where the number of particles is very large. As the number of particles increases, the cost scales quadratically (or as $N \log N$) and makes the computational effort unmanageable. In that case, one cannot simply describe every particles in the system and a method must be devised to reduce the description to just a statistical sample of the particles. This is the approach of the *particle in cell (PIC)* method (sometimes referred to as particle-mesh, PM).

The key idea behind the simulation of weakly coupled systems is to use as building block of the model not single particles but rather collective clouds of them: each *computational particle* (referred to sometimes as superparticle) represents a group of particles and can be visualised as a small piece of phase space. The concept is visualised in Fig. 1.2.

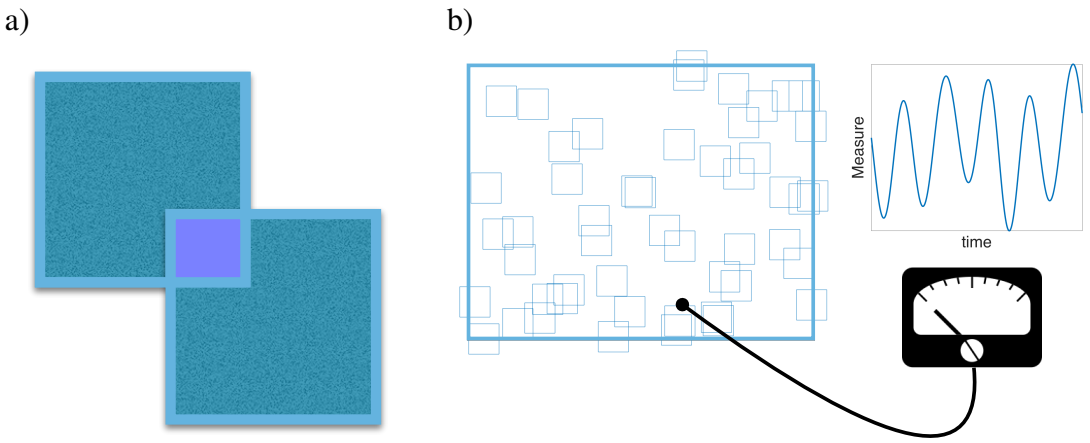

Figure 1.2: Finite size computational particles. Panel a shows the overlap of two finite size particles: the part of the particle clouds that overlap does not contribute to the force between the two clouds. As two clouds get closer, the force increases only till the overlap starts. After the overlap starts increasing, the force starts to decrease to become zero for complete overlap. Panel b shows a Debye cube filled by a set of computational particles to illustrate the idea of the PIC method.

The fundamental advantage of the finite-size particle approach is that the computational particles, being of finite size, interact more weakly than point particles. When two point

particles interact, for example via the Coulomb force, the repulsive or attractive force grows as the particles approach, reaching a singularity at zero separation. Finite size particles, instead, behave as point particles until their respective domains start to overlap. Once overlap occurs the overlap zone is neutralised, not contributing to the force between the particles. At zero distances when the particles fully overlap (assuming here that all particles have the same size) the force becomes zero. The figure presented is 2D for convenience but the reader is invited to try to visualise this also in 3D.

The use of finite-size computational particles allows us to reduce the interaction among particles. A system of finite size particles behaves as a weakly coupled system. Let us return to our fantastic electric field instrument. The passage of a computational particle is the passage of a cloud of finite size that is affecting the instrument more weakly at a point particle. At any given time multiple clouds are in the vicinity rendering the measurement smooth in time. In the same way, the motion of every cloud is affected by the other clouds in the vicinity. When two clouds begin to overlap their relative force weakens. As a result the motion of each cloud is smooth. Even though we have few computational particles in the Debye cube, the behaviour resembles that of weakly coupled systems.

Recalling the definition of plasma parameter, the use of finite-size particles in practice results in a reduction of the potential energy for the same kinetic energy. The beneficial consequence is that the correct plasma parameter can be achieved by using fewer particles than in the physical system. The conclusion is that the correct coupling parameter is achieved by fewer particles interacting more weakly. The realistic condition is recovered in essence making two mistakes that cancel each other: using fewer particles and using weaker forces. Obviously great mathematical insight has gone into making this rigorous [4, 28]. Let us now see how that can be achieved.

1.3 Particle in Cell Method

The kinetic description of a plasma aims at describing the system via electromagnetic fields and distribution functions. A distribution function expresses the probability of finding a particle with certain properties (e.g. position and velocity). In its nature it is a probabilistic description of Nature. It gives up on the hope of following every particle in the system and contents itself by describing the ensemble in statistical terms. A preliminary knowledge of the basic tenants of plasma kinetic theory is assumed in the remainder of this chapter. Classic textbooks provide the necessary preliminary knowledge (see e.g. [55]).

In extreme summary, the starting point of a kinetic simulation model of plasmas is the Boltzmann equation. The phase space distribution function $f_s(\mathbf{x},\mathbf{v},t)$ for a given species s (electrons or ions) is defined as the number of particles at time t in the infinitesimal element of phase space $d\mathbf{x}$ and $d\mathbf{v}$ around a certain phase space point $(\mathbf{x},\mathbf{v})$. The distribution of each species is governed by the Boltzmann equation:

$$\frac{\partial f_s}{\partial t} + \mathbf{v}\cdot\frac{\partial f_s}{\partial \mathbf{x}} + \frac{q_s}{m_s}(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\frac{\partial f_s}{\partial \mathbf{v}} = St(f_s) \tag{1.6}$$

where q_s and m_s are the charge and mass of the species, respectively. The last term is the collision operator that summarises collisions with a neutral medium (for partially ionised

plasmas or for plasmas with immersed dust) or among the charged particles of the plasma. By its definition it is obvious that the integral of f_s over all the phase space is the total number of particles in the system.

The derivatives are written in the standard vector notation in the 3-dimensional configuration space $\mathbf{x}$ and 3-dimensional velocity space $\mathbf{v}$, forming together the classical phase space.

The electric and magnetic fields are given by the two curl Maxwell's equations,

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{B} &= \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mu_0 \mathbf{j} \end{aligned} \tag{1.7}$$

supplemented by the two divergence equations:

$$\begin{aligned} \varepsilon_0 \nabla \cdot \mathbf{E} &= \rho \\ \nabla \cdot \mathbf{B} &= 0 \end{aligned} \tag{1.8}$$

where the net charge density is computed from the distribution functions as:

$$\rho(\mathbf{x},t) = \sum_s q_s \int_{\mathbb{V}} f_s(\mathbf{x},\mathbf{v},t) d\mathbf{v} \tag{1.9}$$

and the current density as:

$$\mathbf{j}(\mathbf{x},t) = \sum_s q_s \int_{\mathbb{V}} \mathbf{v} f_s(\mathbf{x},\mathbf{v},t) d\mathbf{v} \tag{1.10}$$

where $\mathbb{V}$ is the velocity space.

1.3.1 Mathematical Formulation of PIC

The PIC method can be regarded as a representation of the distribution function of each species by a superposition of moving elements, each representing a cloud of physical particles. We follow here the specific approach described in Ref. [40].

The mathematical formulation of the PIC method is obtained by assuming that the distribution function of each species is given by the superposition of several elements (called computational particles or superparticles):

$$f_s(\mathbf{x},\mathbf{v},t) = \sum_p f_p(\mathbf{x},\mathbf{v},t) \tag{1.11}$$

Each element represents a large number of physical particles that are near each other in phase space . For this reason, the choice of the elements is made in order to be at the same time physically meaningful (i.e. to represent a group of particles near each other) and mathematically convenient (i.e. to allow the derivation of a manageable set of equations) citedawson.

The PIC method is based upon assigning to each computational particle a specific functional form for its distribution, a functional form with a number of free parameters whose time evolution will determine the numerical solution of the Vlasov equation [14]. In the standard PIC methods [4, 28], the choice is made to have two free parameters in the functional shape for each spatial dimension. The free parameters will acquire the physical meaning of position and velocity of the computational particle. More advanced method take into account also the evolving shape of the physical cloud the super particle is supposed to represent [14].

In the standard PIC model, the functional dependence is assumed to be a tensor product of the shape in each direction of phase space [28]:

$$f_p(\mathbf{x},\mathbf{v},t) = N_p S_{\mathbf{x}}(\mathbf{x}-\mathbf{x}_p(t)) S_{\mathbf{v}}(\mathbf{v}-\mathbf{v}_p(t)) \tag{1.12}$$

where $S_{\mathbf{x}}$ and $S_{\mathbf{v}}$ are the *shape functions* for the computational particles and N_p is the number of physical particles that are present in the element of phase space represented by the computational particle.

A number of properties of the shape functions come from their definition:

1. The support of the shape functions is compact, to describe a small portion of phase space, (i.e. it is zero outside a small range).

2. Their integral over the domain is unitary:

$$\int_{V_\xi} S_\xi(\xi-\xi_p)d\xi = 1 \tag{1.13}$$

   where ξ stands for any coordinate or any velocity direction.

3. While not strictly necessary, Occam's razor suggests to choose symmetric shapes:

$$S_\xi(\xi-\xi_p) = S_\xi(\xi_p-\xi) \tag{1.14}$$

While these definitions still leave very broad freedom in choosing the shape functions, traditionally the choices actually used in practice are very few.

1.3.2 Selection of the particle shapes

A critical choice in the definition of a PIC algorithm is the choice of the shape functions.

For the velocity, $S_{\mathbf{v}}$, virtually all PIC methods assume a Dirac's delta in each direction:

$$S_{\mathbf{v}}(\mathbf{v}-\mathbf{v}_p) = \delta(v_x-v_{xp})\delta(v_y-v_{yp})\delta(v_z-v_{zp}) \tag{1.15}$$

This choice has the fundamental advantage that if all particles within the element of phase space described by one computational particle have the same speed, they remain closer in phase space during the subsequent evolution.

The original PIC methods developed in the 50's were based on using a Dirac's delta also as the shape function in space. But now for the spatial shape functions, all commonly used PIC methods are based on the use of the so-called b-splines [5]. The b-spline functions are a

series of consecutively higher order functions obtained from each other by integration. The first b-spline is the flat-top function $b_0(\xi)$ defined as:

$$b_0(\xi) = \begin{cases} 1 & if\ |\xi| < 1/2 \\ \\ 0 & \text{otherwise} \end{cases} \tag{1.16}$$

The subsequent b-splines, b_ℓ, are obtained by successive integration via the following generating formula:

$$b_\ell(\xi) = \int_{-\infty}^{\infty} d\xi' b_0(\xi - \xi') b_{\ell-1}(\xi') \tag{1.17}$$

Figure 1.3 shows the first three b-splines.

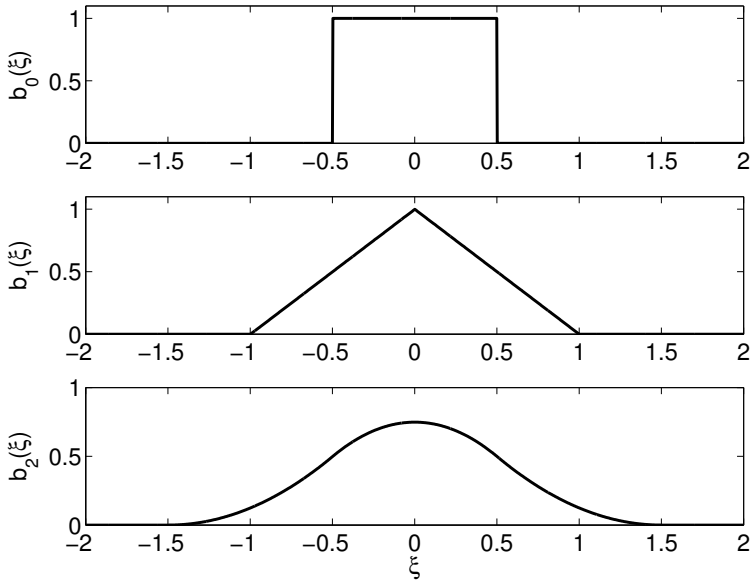

Figure 1.3: First three b-spline functions, $b_\ell(\xi)$.

Two crucial properties of splines are :

(a) when a b-spline of any order is evaluated on a uniform grid of step 1, the sum over all points of evaluation is unitary, regardless of the central point ξ of the b-spline:

$$\sum_i b_\ell(\xi + i) = 1, \tag{1.18}$$

a property of great convenience in particle interpolation;

(b) The integral of b-splines of any order is unitary:

$$\int_{-\infty}^{\infty} b_\ell(\xi) d\xi = 1. \tag{1.19}$$

(c) The Dirac's delta $\delta(\xi)$ can be regarded as $b_{-1}(\xi)$.

Based on the b-splines, the spatial shape function of PIC methods is chosen as:

$$S_{\mathbf{x}}(\mathbf{x}-\mathbf{x}_p) = \frac{1}{\Delta x_p \Delta y_p \Delta z_p} b_\ell\left(\frac{x-x_p}{\Delta x_p}\right) b_\ell\left(\frac{y-y_p}{\Delta y_p}\right) b_\ell\left(\frac{z-z_p}{\Delta z_p}\right) \tag{1.20}$$

where Δx_p, Δy_p and Δz_p are the lengths of the support of the computational particles (i.e. its size) in each spatial dimension.

A common choice in PIC is to use the shape functions equal to the b-splines of order 0, a choice referred to as *cloud in cell* because the particle is a uniform square cloud in space with infinitesimal span in the velocity directions.

1.3.3 Derivation of the equations of motion

Following the procedure in Ref. [40], to derive the evolution equations for the free parameters $\mathbf{x}_p$ and $\mathbf{v}_p$, we require the first moments of the Vlasov equation to be exactly satisfied by the functional forms chosen for the elements. This procedure requires some explanation.

First, the Vlasov equation is formally linear in f_s and the equation satisfied by each element is still the same Vlasov equation. The linear superposition of the elements gives the total distribution function and if each element satisfies the Vlasov equation, the superposition does too. A caveat is that the fields really depend on f_s making the Vlasov equation nonlinear. As a consequence the fields used in each Vlasov equation for each element must be the total fields due to all elements, the same entering the complete Vlasov equation for f_s. We can then write the Vlasov equation for each element f_p:

$$\frac{\partial f_p}{\partial t} + \mathbf{v}\cdot\frac{\partial f_p}{\partial \mathbf{x}} + \frac{q_s}{m_s}(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\frac{\partial f_p}{\partial \mathbf{v}} = 0 \tag{1.21}$$

and observe that their sum reproduces correctly the Vlasov equation for the complete distribution function f_s.

Second, the arbitrary functional form chosen for the elements does not satisfy exactly the Vlasov equation. If we substitute the definition (1.12) in eq. (1.21), the equation cannot be satisfied exactly at every point. This circumstance is typical of finite element methods [63] and is addressed by requiring that the equations be satisfied only in a weak sense, meaning that some averages are satisfied even though locally point by point the equation in not exactly valid.

In analogy with the usual procedure for finite element methods, we require that the moments of the equation are satisfied. The moments of the Vlasov equation are obtained multiplying by powers of $\mathbf{x}$ and $\mathbf{v}$ and integrating. Using the zeroth order moment and the six first order moments (one for each variable in phase space), the following complete set of evolution equations is obtained for the parameters defining the functional dependence of the distribution within each computational particle [40]:

$$\frac{dN_p}{dt} = 0$$

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p \tag{1.22}$$

$$\frac{d\mathbf{v}_p}{dt} = \frac{q_s}{m_s}(\mathbf{E}_p + \mathbf{v}_p \times \mathbf{B}_p)$$

A great advantage of the PIC method is that its evolution equations resemble the same Newton equations as followed by the regular physical particles. The key difference is that the field is computed as average over the shape function based on the definition of $\mathbf{E}_p$ and $\mathbf{B}_p$ as:

$$\mathbf{E}_p = \int S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p)\mathbf{E}(\mathbf{x})d\mathbf{x} \tag{1.23}$$

$$\mathbf{B}_p = \int S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p)\mathbf{B}(\mathbf{x})d\mathbf{x} \tag{1.24}$$

obtained in the course of the procedure for imposing the validity of the moments of the Vlasov equation.

The set of equations above provide a closed description for the Vlasov equation. Once accompanied by an algorithm to solve Maxwell's equations the full Vlasov-Maxwell system can be solved.

In summary, the particle method, as derived above, is a discretisation of phase space based on a finite set of computational particles. The continuum distribution function is replaced by a discrete mathematical representation provided by the superposition of moving fixed-shape computational particles of finite size.

The pitfalls of this representation have been studied in depth in the past. A full discussion of this aspect is beyond the scope of the present review. We refer the reader to Ref. [4, 28, 14] for additional comments. In extreme summary, a element of phase space initially described correctly by the shape functions chosen for the discretisation would be distorted by non-uniform electric and magnetic fields. Instead, the computational particles have fixed shape and neglect this effect. The Liouville theorem requires the conservation of the phase-space volume of each element, a process that is still correctly described by the discretised system.

Alternatives to PIC have been developed to remove this error by introducing additional degrees of freedom for each computational element (superparticles) [14, 3]. The additional degrees of freedom can describe the particle shape and represent its distortion in phase space. However, the additional complexity of such methods have so far prevented their widespread use.

1.4 Coupling with the Field Equations: Spatial discretization on a grid

So far the attention has focused on the particle discretisation of the Vlasov part of the Vlasov-Maxwell system. To complete the simulation method, Maxwell's equations need to be discretised as well.

A wide variety of methods have been developed to do so. Essentially all electromagnetic solvers can be coupled with particle methods [28, 4, 26]. We use a general notation where the continuum operator ∇ is replaced by a discrete operator that approximate it, using the notation ∇_g. Th discretised Maxwell's equations become:

$$\begin{aligned} \nabla_g \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla_g \times \mathbf{B} &= \mu_0 \mathbf{J}_g + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \end{aligned} \tag{1.25}$$

supplemented by the two divergence conditions:

$$\begin{aligned} \nabla_g \cdot \mathbf{E} &= \frac{\rho_g}{\varepsilon_0} \\ \nabla_g \cdot \mathbf{B} &= 0 \end{aligned} \tag{1.26}$$

Maxwell's equations require two inputs, the sources ρ_g and $\mathbf{j}_g$ that come from the information carried by the particles. In turn, the outputs are the electric and magnetic fields (or, equivalently the vector and scalar potential). The main concern here is the coupling of Maxwell's equations with the particles and its consequences.

We assume that the fields are known at grid points, not necessarily the same for both fields, and are defined as point values or averages over control volumes: $\mathbf{E}_g$, $\mathbf{B}_g$ where the index g labels the discrete points. Similarly, we assume that the chosen Maxwell scheme needs the sources at discrete grid points defined as averages over control volumes V_g defining each discrete point:

$$\begin{aligned} \rho_g &= \sum_s \frac{q_s}{V_g} \int_{\mathbb{V}} \int_{V_g} f_s d\mathbf{v} d\mathbf{x} \\ \mathbf{j}_g &= \sum_s \frac{q_s}{V_g} \int_{\mathbb{V}} \int_{V_g} \mathbf{v} f_s d\mathbf{v} d\mathbf{x} \end{aligned} \tag{1.27}$$

In all the formulas above we indicated the grid index as g generically. The cell positions can be chosen differently for different fields and different sources. Staggered schemes and Yee lattices are commonly used [61, 62]. The considerations below are independent of the details on how space is discretised.

Recalling eq.(1.11-1.12), the definition of the sources for Maxwell's equations requires a 3-dimensional integral of the particle shape over the control volume of each grid point:

$$\begin{aligned} \rho_g &= \sum_p \frac{1}{V_g} \int_{V_g} S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p) d\mathbf{x} \\ \mathbf{j}_g &= \sum_p \mathbf{v}_p \frac{1}{V_g} \int_{V_g} S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p) d\mathbf{x} \end{aligned} \tag{1.28}$$

where the integrations in velocity space were done easily recalling the definition of $S_{\mathbf{v}}$. The summation is over all particles of all species (for this reason the summation over the species is not indicated explicitly).

The integral in space is straightforward but can involve complicated geometric operations if one allows general shapes for the control volumes. Instead, the use of b-splines introduced above simplifies this operation tremendously. If the particle shapes are chosen as in eq. (1.20) and the grid is chosen uniform, Cartesian and with cell sizes in each dimension equal to the particle sizes, a simple and elegant reformulation of the integration step can be obtained recalling property (1.17) of b-splines. Each dimension can be done separately thanks to the choice of a Cartesian uniform grid and of the shape of particles as products of shapes in each direction. If we use the properties of the top hat function (i.e. the b-spline of order 0), the integral needed in each direction can be reformulated as

$$\int_{\Delta x_g} S_x(x-x_p) = \int_{-\infty}^{\infty} S_x(x-x_p) b_0\left(\frac{x-x_g}{\Delta x_g}\right) dx \tag{1.29}$$

where Δx_g is the interval in x relative to control volume V_g. Choosing, $\Delta x_g = \Delta x_p$ (called simply Δx), recalling the definition of S_x, eq. (1.20) and using eq. (1.17):

$$\int_{-\infty}^{\infty} S_x(x-x_p) b_0\left(\frac{x-x_g}{\Delta x}\right) dx = b_{l+1}\left(\frac{x_g-x_p}{\Delta x}\right) \tag{1.30}$$

Collecting the integral in each direction, the so-called interpolation function can be defined:

$$W(\mathbf{x}_g-\mathbf{x}_p) = b_{l+1}\left(\frac{x_g-x_p}{\Delta x}\right) b_{l+1}\left(\frac{y_g-y_p}{\Delta y}\right) b_{l+1}\left(\frac{z_g-z_p}{\Delta z}\right) \tag{1.31}$$

The summation property of the b-splines, eq. (1.18), ensures that:

$$\sum_g W(\mathbf{x}_g-\mathbf{x}_p) = 1 \tag{1.32}$$

and the sum of the fractional contributions of a particle to the grid is unitary regardless of the position of the particle.

The interpolation function is a direct consequence of the choice made for the shape function. Using b-splines proves a powerful choice as it allows us to write the interpolation function just as the b-spline one order higher than that used for the shape. The calculation of the sources for Maxwell's equations becomes then just the sum of a number of function evaluations without requiring geometrically complex integrals:

$$\begin{aligned} \rho_g &= \frac{1}{V_g}\sum_p q_p W(\mathbf{x}_g-\mathbf{x}_p) \\ \mathbf{j}_g &= \frac{1}{V_g}\sum_p q_p \mathbf{v}_p W(\mathbf{x}_g-\mathbf{x}_p) \end{aligned} \tag{1.33}$$

An analogous set of operations can be carried out for connecting the output of the Maxwell's equations to the particle equations of motion. The needed quantities are the electric and magnetic fields derived above in eq. (1.23,1.24). The definitions for the particle fields require continuum fields. These can be obtained once again using interpolation:

$$\begin{aligned} \mathbf{E}(\mathbf{x}) &= \sum_g \mathbf{E}_g S_{\mathbf{E}}(\mathbf{x}-\mathbf{x}_g) \\ \mathbf{B}(\mathbf{x}) &= \sum_g \mathbf{B}_g S_{\mathbf{B}}(\mathbf{x}-\mathbf{x}_g) \end{aligned} \tag{1.34}$$

where $S_{\mathbf{E}}(\mathbf{x}-\mathbf{x}_g)$ and $S_{\mathbf{B}}(\mathbf{x}-\mathbf{x}_g)$ are the fields interpolation functions. Upon substitution in the definition of the particle fields,

$$\begin{aligned}\mathbf{E}_p &= \sum_g \int S_{\mathbf{x}}(\mathbf{x}-\mathbf{x}_p)\mathbf{E}_g S_{\mathbf{E}}(\mathbf{x}-\mathbf{x}_g)d\mathbf{x} \\ \mathbf{B}_p &= \sum_g \int S_{\mathbf{x}}(\mathbf{x}-\mathbf{x}_p)\mathbf{B}_g S_{\mathbf{B}}(\mathbf{x}-\mathbf{x}_g)d\mathbf{x}\end{aligned} \tag{1.35}$$

The geometrical complexity of the integrals can be avoided choosing the interpolation functions for the electric and magnetic fields as b-splines of order 0:

$$S_{\mathbf{E},\mathbf{B}}(\mathbf{x}-\mathbf{x}_g) = b_0\left(\frac{x_g-x_p}{\Delta_x}\right) b_0\left(\frac{y_g-y_p}{\Delta_y}\right) b_0\left(\frac{z_g-z_p}{\Delta_z}\right) \tag{1.36}$$

leading to

$$\begin{aligned}\mathbf{E}_p &= \sum_g \mathbf{E}_g W(\mathbf{x}_g-\mathbf{x}_p)d\mathbf{x} \\ \mathbf{B}_p &= \sum_g \mathbf{B}_g W(\mathbf{x}_g-\mathbf{x}_p)d\mathbf{x}\end{aligned} \tag{1.37}$$

with the same interpolation function used for the sources.

The majority of PIC codes use b-spline interpolations and can take advantage of the convenient interpolation function defined above. For staggered grid, the interpolation will happen to different grid points for different quantities, but using the same interpolation function. In some implementations, different orders of interpolations can be used for different quantities [67]. A more radical solution is to consider non-uniform [39, 17] or even unstructured grids [31]. In the most complex cases, the exchange of information between particles and cells cannot use the interpolation function derived using b-splines and the integral definitions need to be used directly.

1.5 Temporal Discretisation of the Particle Methods

The coupled Vlasov-Maxwell system requires three discretisations. We have first described how to discretise phase space in the Vlasov equation using finite-size computational particles that resemble flat strings of finite size in space but of infinitesimal size in velocity. Next we have given some indications on how to discretise space. Space needs to be discretises on Maxwell's equations and a vast array of possibilities exist in the computational electromagnetism. We focused on how to couple these discretisations with the computational particle information.

The last and most important discretisation remaining is time. Again for Maxwell's equations the temporal discretisation can be done in any of the various choices used in computational electromagnetism. What we need to focus on is how to combine the time discretisation of the Maxwell equations with that of the particles. Figure 1.4 illustrates the concept. The field equations are solved on a grid where the fields are advanced based on the sources provided by the particles. The particle equations are solved using the fields computed on the

grid. The interpolation formulas derived above provide the rule to exchange information between grid and particles.

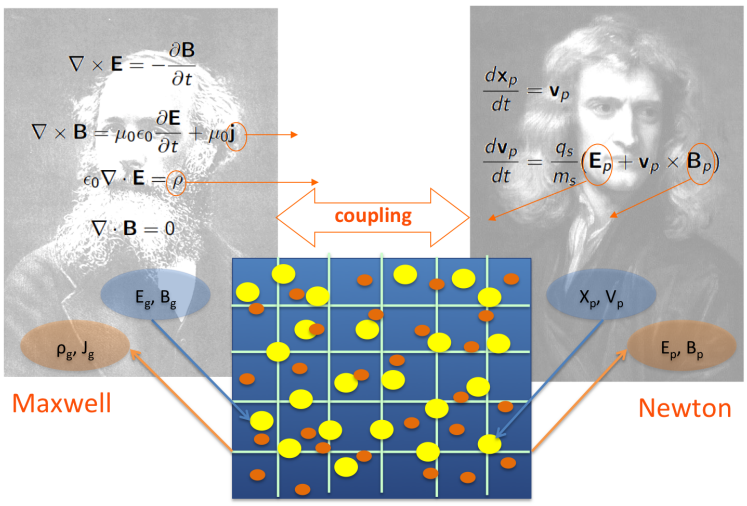

Figure 1.4: Illustration of the coupling between Maxwell's and Newton's equations. Newton's equations need the electric and magnetic fields and Maxwell's equations need the particle positions to compute the sources: current and density. The computational particles are indicated as coloured dots, bigger yellow ions and smaller orange electrons. The dot size is indicative of the mass of the particles, not their sizes. All computational particles have the same size: their size is identical to the size of the cells but with their centre on the centre of the particles.

The question is how to deal with the coupling of Maxwell's equations for the fields and Newton's equations for the computational particles. In principle as the particles move, the fields evolve as well, any change in one side of Fig. 1.4 reflects on the other. The first point to keep in mind is the vastness of the information being exchanged. Modern PIC methods on supercomputers (i.e. parallel computers made of hundreds of thousands of processors) use millions of cells and billions of particles. In fact, the state of the art is starting to reach the trillion particle level. The desire to use more and more particles to cover phase space more accurately suggests to keep the operations per particle as simple as possible.

This guiding principle has long suggested to use explicit methods, a decision whose wisdom will be questioned below. In explicit methods, the two sets of equations, Maxwell's and Newton's are solved in a marching order. Using the visualisation scheme in Fig. 1.4, we can assume that each side can be advanced for a small time step, while the other is assumed temporarily frozen. We can advance the particles for a small time step in given fields. We use then the new particle positions and speeds to compute the current and density to advance the fields for the same small time step. If the time step is small enough, this procedure has a small error. The method just described is called explicit. The advantages of the explicit approach is to be extremely simple but the disadvantage is that the time step has to be very small.

1.5.1 Explicit Temporal Discretisation of the Particle Equations

Let us start with discretising in time the Newton's equations for the particles. The simplest and most widely used explicit algorithm is the so-called *leap-frog algorithm* based on staggering the time levels of the velocity and position by half time step: $\mathbf{x}_p(t = n\Delta t) \equiv \mathbf{x}_p^n$ and $\mathbf{v}_p(t = (n+1/2)\Delta t) \equiv \mathbf{v}_p^{n+1/2}$. The update of position from time level n to time level $n+1$ uses the velocity at mid-point $\mathbf{v}_p^{n+1/2}$, and similarly the update of the velocity from time level $n-1/2$ to $n+1/2$ uses the mid point position x_p^n. This stepping of velocity over position and of position over velocity gives the method its name for its resemblance to the children's game bearing the same name (see Fig. 1.5).

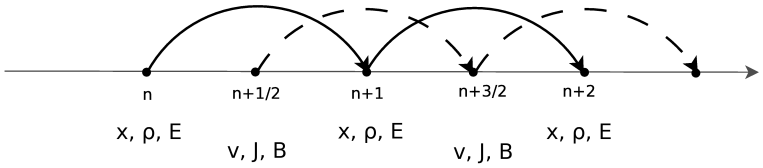

Figure 1.5: Visual representation of the leap-frog algorithm. The time discretisation is staggered, with the electric field, charge density and particle positions at integer times and the magnetic field, current and particle velocity at half time steps.

The scheme is summarised by:

$$\begin{aligned} &\frac{\mathbf{x}_p^{n+1} - \mathbf{x}_p^n}{\Delta t} = \mathbf{v}_p^{n+1/2} \\ &\frac{\mathbf{v}_p^{n+1/2} - \mathbf{v}_p^{n-1/2}}{\Delta t} = \frac{q_s}{m_s}\mathbf{E}^n(x_p^n) + \frac{q_s}{m_s}\left(\frac{\mathbf{v}_p^{n+1/2} + \mathbf{v}_p^{n-1/2}}{2}\right) \times \mathbf{B}^n(x_p^n) \end{aligned} \tag{1.38}$$

where the use of $\mathbf{E}_p^n$ and $\mathbf{B}_p^n$ implies knowing the solution of Maxwell's equations given the particle information. The first equation is clearly explicit, the second can be formulated explicitly as a roto-translation of the vector $\mathbf{v}_p^{n-1/2}$. The velocity equation can be rewritten in the equivalent explicit form:

$$\mathbf{v}_p^{n+1/2} = 2\left(\widehat{\mathbf{v}}_p + \beta_s \widehat{\mathbf{E}}_p\right) - \mathbf{v}_p^{n-1/2} \tag{1.39}$$

where hatted quantities have been rotated by the magnetic field:

$$\begin{aligned} \widehat{\mathbf{v}}_p &= \alpha_p^n \mathbf{v}_p^{n-1/2} \\ \widehat{\mathbf{E}}_p &= \alpha_p^n \mathbf{E}_p^n \end{aligned} \tag{1.40}$$

via a rotation matrix α_p^n defined as:

$$\alpha_p^n = \frac{1}{1 + (\beta_s B_p^n)^2}\left(\mathbb{I} - \beta_s \mathbb{I} \times \mathbf{B}_p^n + \beta_s^2 \mathbf{B}_p^n \mathbf{B}_p^n\right) \tag{1.41}$$

where $\mathbb{I}$ is the dyadic tensor (matrix with diagonal of 1) and $\beta_s = q_p \Delta t / 2 m_p$ (independent of the particle weight and unique to a given species). The shorter notation $\mathbf{E}^n(\mathbf{x}_p^n) = \mathbf{E}_p^n$ has been introduced for all fields. An alternative implementation of the rotation procedure above is called the Boris mover [4].

Note that technically the leap-frog algorithm is second order accurate, when instead the regular explicit Euler-scheme is only first order. Nevertheless, the two differ in practice only for the fact that the velocity is staggered by half time step. This staggering is achieved by moving the initial velocity of the first time cycle by half a time step using an explicit method:

$$\frac{\mathbf{v}_p^{1/2} - \mathbf{v}_p^0}{\Delta t/2} = \frac{q_s}{m_s}\mathbf{E}_p^0(x_p^0) + \frac{q_s}{m_s}\left(\frac{\mathbf{v}_p^{1/2} + \mathbf{v}_p^0}{2}\right) \times \mathbf{B}_p^0(x_p^0) \tag{1.42}$$

after the first step all subsequent steps would be identical to the Euler explicit scheme.

1.5.2 Explicit PIC Cycle

After discretising the particles equation of motion, the Maxwell's equations need also to be discretised in time. As for space, also in time different methods are available and the interested reader is referred to textbooks on that subject [62]. The simplest approach is to use again the leap-frog algorithm also for the fields. The electric field is computed at the integer levels while the magnetic field is computed at the intermediate times (see Fig. 1.5). Starting form the partially discretised equations 1.25 introduced above, time can be discretised as:

$$\begin{aligned} \nabla_g \times \mathbf{E}^n &= \frac{\mathbf{B}^{n+1/2} - \mathbf{B}^{n-1/2}}{\Delta t} \\ \nabla_g \times \mathbf{B}^{n+1/2} &= \mu_0 \mathbf{J}_g^{n+1/2} + \mu_0 \varepsilon_0 \frac{\mathbf{E}^{n+1} - \mathbf{E}^n}{\Delta t} \end{aligned} \tag{1.43}$$

where all differences are time-centered, resulting in a second order accurate scheme.

The current needs to be accumulated from the particle information at intermediate times $n+1/2$. Smartly, the leap-frog algorithm for the particles has the particle velocities at that same time level, but unfortunately the position is staggered by half time step. A simple choice is to use an avenging method:

$$\mathbf{j}_g^{n+1/2} = \frac{1}{V_g}\sum_p q_p \mathbf{v}_p^{n+1/2} \frac{W(\mathbf{x}_g - \mathbf{x}_p^n) + W(\mathbf{x}_g - \mathbf{x}_p^n)}{2} \tag{1.44}$$

In a similar manner the magnetic field at integer times needed in the mover is obtained by averaging from the neighbouring intermediate times:

$$\mathbf{B}^n = \frac{1}{2}\left(\mathbf{B}^{n+1/2} + \mathbf{B}^{n-1/2}\right) \tag{1.45}$$

In the continuum, the divergence conditions are valid at all times if they are valid at the initial time and if the charge continuity equation is satisfied. In PIC the latter condition is not necessarily valid. The divergence of the magnetic field is always zero, if the the discretised

operator retains the property that the divergence of the curl is zero. This condition is often easily satisfied. But the other divergence condition is a major issue in PIC.

Historically two solutions have been adopted. First, the charge and current interpolation from the particle to the grid can be modified in a manner that ensures the validity of the charge continuity equation: these schemes are called charge conserving [21]. Second, the divergence condition can be applied as a correction to the electric field. At each time step, or at regular intervals, the electric field can be *cleaned* for any component not satisfying the Gauss law. The operation is called divergence cleaning and is numerically rather expensive because it requires solving an elliptic problem with an iterative method. Simplified approaches have been developed [35].

In practice, the two methods can be used in combination because even though a charge conserving scheme might do a good job at keeping the divergence correct, numerical errors will start to accumulate and a divergence cleaning operation can be judiciously applied regularly after a number of time steps.

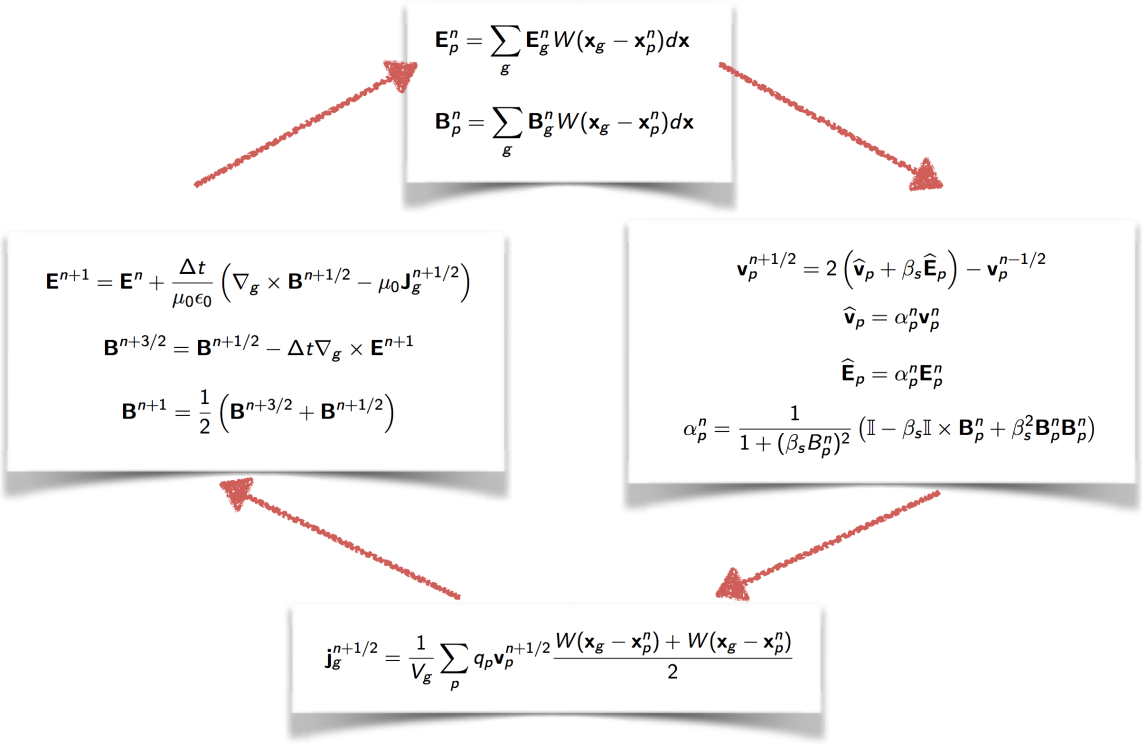

Figure 1.6: Explicit PIC cycle. Four steps are followed in sequence over a time step Δt. Starting for example from the top, the electric and magnetic fields from the previous time step are interpolated to the particles. Next the particles are advanced over Δt to find the new position and velocity. The particle information is then interpolated to the grid to obtain the current density in each cell. The fields are then advanced for the same Δt on the grid. A new time step is ready to start.

The end result is a simple marching algorithm that is illustrated in Fig. 1.6. Each step in the figure is self-contained and requires only information that is available from the previous

steps in the cycle. There is no iteration needed. This is the critical advantage of explicit PIC: the algorithm advances forward without any iterations. But this comes at a price. We have assumed in each step that the others remain frozen. During the time step Δt of the cycle, the fields are frozen while the particles move and the particles are frozen while the fields evolve. This is only acceptable if Δt is small. Very small.

1.5.3 Electrostatic Explicit Methods

In many applications, the currents are sufficiently small to justify an electrostatic approximation. In this case the two curl equations can be ignored and the only remaining equation in the Maxwell system is the Gauss law that can be rewritten in terms of the electrostatic potential:

$$\Delta_g \phi = -\frac{\rho_g}{\varepsilon_0} \tag{1.46}$$

where the laplacian operator is discretised on the grid.

In this case there is no escape form the need of solving the elliptic problem with the curious consequence that explicit electrostatic PIC is in fact more complex and expensive than explicit electromagnetic PIC. But note that using an electromagnetic PIC for an electrostatic case requires one to make sure the divergence cleaning is very accurate. In essence this requires again solving the elliptic Poisson equation and the cost is the same. There is no forgiveness in electromagnetism.

1.5.4 Stability of the Explicit PIC Method

Explicit PIC is subject to three stability conditions. Exceed them and energy explodes rendering the simulation meaningless. Users must never exceed these limits.

Stability of the particle mover

The discretisation of the equations of motion using the leap-frog algorithm (or other similar explicit time differencing) introduces a stability constraint. As usual stability can be studied with the Von Neumann analysis [30, 28, 4]: the equations are linearised and their stability is studied by Fourier analysis. Note that the leap-frog algorithm is implicit in the Lorentz force term, but explicit on the electric force. As a consequence only the latter is of primary concern and stability can be studied in the limit of zero magnetic field.

Stability needs to address all temporal scales in the system. Obviously the fastest are the most challenging for an explicit scheme. Among the plasma waves, the fastest electrostatic wave is the Langmuir (i.e. electron plasma) wave.

We proceed using the Von Neumann analysis [30], meaning we assume the temporal evolution to be harmonic so that all quantities depend on time as $e^{i\omega t}$:

$$\begin{aligned} \mathbf{x}_p^n &= \widetilde{\mathbf{x}}_p e^{i\omega n\Delta t} \\ \mathbf{v}_p^{n+1/2} &= \widetilde{\mathbf{v}}_p e^{i\omega (n+1/2)\Delta t} \\ \mathbf{E}_p^n &= \widetilde{\mathbf{E}}_p e^{i\omega n\Delta t} \end{aligned} \tag{1.47}$$

where ω is the temporal frequency of the numerical response.

We consider the mover introduced in eq. (1.38) repeated here for the case of a Langmuir wave (i.e. without the magnetic field) in a single spatial and single velocity dimension:

$$\begin{aligned} x_p^{n+1} &= x_p^n + \Delta t v_p^{n+1/2} \\ v_p^{n+1/2} &= v_p^{n-1/2} + \frac{q_p \Delta t}{m_p} E_p^n \end{aligned} \tag{1.48}$$

Substituting in each term the harmonic dependence, we obtain in the Fourier transformed space:

$$\begin{aligned} \widetilde{x}_p \left(e^{i\omega\Delta t} - 1 \right) &= \widetilde{v}_p \Delta t e^{i\omega\Delta t/2} \\ \widetilde{v}_p \left(e^{i\omega\Delta t/2} - e^{-i\omega\Delta t/2} \right) &= \frac{q_s \Delta t}{m_s} \widetilde{E}_p \end{aligned} \tag{1.49}$$

Recalling now the basic property of cold plasma Langmuir waves that relate the variation of the particle displacement with the electric field in Fourier space as [19]:

$$\frac{q_s \widetilde{E}}{m_s} = -\omega_{pe}^2 \widetilde{x} \tag{1.50}$$

the final solution is given by the algebraic system:

$$\begin{pmatrix} e^{i\omega\Delta t} - 1 & -\Delta t e^{i\omega\Delta t/2} \\ \omega_{pe}^2 \Delta t & e^{i\omega\Delta t/2} - e^{-i\omega\Delta t/2} \end{pmatrix} \begin{pmatrix} \widetilde{x}_p \\ \widetilde{v}_p \end{pmatrix} = 0 \tag{1.51}$$

The dispersion relation can be computed as the determinant of the system matrix [28, 4]:

$$\left(\frac{\omega_{pe}\Delta t}{2} \right)^2 = \sin^2 \left(\frac{\omega \Delta t}{2} \right) \tag{1.52}$$

where the Euler formula for the sin function was used. The dispersion relation (1.52) expresses the frequency of the numerical solution for particle motion in Langmuir waves with physical frequency ω_{pe}. The numerical frequency is not the same as the physically correct frequency.

Notoriously the sin function of real arguments has values only in the $[-1, 1]$ interval. As a consequence, for any values of $\omega_{pe}\Delta t > 2$, the Von Neumann analysis leads to a complex numerical oscillation frequency, ω. Indeed being the problem real to begin with, two complex conjugate solutions ω arise, one of the two being damped and the other a growing exponential. The numerical solution becomes unstable, with a numerical growth rate that has no physical equivalence. In practice, the particles heat unboundedly and quickly and the simulation fails within a few time steps.

The first cardinal rule of explicit plasma simulation is that the condition

$$\omega_{pe}\Delta t < 2 \tag{1.53}$$

should never be violated. Indeed, practice advises the use of a considerably smaller time step of order $\omega_{pe}\Delta t = .1$, to avoid numerical heating [4].

The temporal stability constraint produces a first great challenge to the application of PIC methods in practice. The range of time scales typical in plasmas covers several orders of magnitude. The electrons are much faster than the ions (due to the mass difference). What the stability constraint imposes is to choose a time step that is able to resolve the fastest frequency in the system: the electron plasma frequency. If we are interested in following longer time scales, we still need to use a time step that follows the Langmuir waves. Even if our insight tells us there is nothing of interest in those small scales.

Stability due to the Explicit Time Differencing of the Field Equations

Discretising the Maxwell's equations explicitly leads to the imposition of the Courant-Friedrich-Levy (CFL) condition. This condition arises whenever an hyperbolic equation is discretised explicitly [53]. The CFL condition states that the time step must not exceed the time taken by the characteristic signals of the hyperbolic equation to travel one cell. In the case of the Maxwell's equations the characteristic speed of the signal is the speed of light, c. The CFL condition then requires:

$$\Delta t < \Delta x / c \tag{1.54}$$

This condition is in a sense less damaging than the previous one because it can be satisfied by a large Δt if Δx is also large. Often in practice, one needs to resolve long term processes developing over macroscopic scales. In this case both Δt and Δx are large. Additionally it is relatively easy to discretise just the Maxwell's equations implicitly removing just this constraint [52]. As we will see below what is difficult is to deal with the coupling of particles and fields implicitly. But just discretising the Maxwell's equations implicitly is a simple task by comparison.

Space Resolution and Finite Grid Instability

The introduction of a grid to compute the field characterises the particle methods for plasma physics as particle in cell (PIC) methods. The particles move in a continuum space, but their information is projected to grid points using integration over control volumes. This operation reduces the information, collapsing the continuum particle shapes to discrete contributions in a set of points (the grid values). The loss of information allows one numerical instability to creep in: the finite grid instability.

The mathematical study of the finite grid instability is complex, requiring a careful and complete analysis of all computational steps with the Laplace transformation of time and the Fourier transformation of space. The reader interested in the details is referred to the textbooks on the subject [4, 28] and to the original work cited therein. The summary of the analysis is that to avoid the finite grid instability in explicit particles methods, the grid spacing must be chosen to satisfy the constraint

$$\Delta x / \lambda_{De} < \varsigma \tag{1.55}$$

where λ_{De} is the Debye length [28, 4] and the parameter ς is of order one and depends on the details of the implementation. For the widely used choice of shape functions as b-spline

of order 0 and consequently interpolation functions of order 1, the scheme referred to as cloud in cell (CIC), the literature reports, $\varsigma \approx \pi$. Using filtering methods and higher order interpolation this number can be increased significantly allowing the use of larger cells.

The practical consequence of the finite grid instability is a tremendous numerical heating of the plasma characterised by an alternatively positive-negative variation of the electric field accompanied by a correlated zig-zag perturbation of the phase-space. In a very short time, the energy reaches the bounds of overflow in the machine representation of numbers. The finite grid instability must be avoided, requiring its stability constraint to be respected everywhere in the domain for all directions.

The limit imposed by this stability constraint is devastating. The spatial scale to resolve is the Debye length in the highest density region of the domain where the Debye length is smallest. If using uniform grids, this constraint requires to use cell sizes that can be several orders of magnitude smaller than the scales of interest. Note that this consideration has two sides.

First, often the processes of interest do not reach down to the Debye scale. For example most electromagnetic processes are at their smallest on the electron inertial length, $d_e = c/\omega_{pe}$. This scale can be orders of magnitude larger than the Debye length. We remind that the Debye length is the electron thermal speed divided by the plasma frequency while the electron inertial length is the speed of light divided by the plasma frequency: the inertial length than is obviously larger than the Debye length by the ratio v_{the}/c. Having to resolve the Debye length purely for numerical stability reasons implies a great waste in each direction. When done in 3D, this waste compounds. If the time scale constraint encountered before hits the explicit approach once, the spatial stability constraint hits it three times, once per spatial dimension. The combined effect of these two constraint makes it imperative to look beyond explicit particle methods for many practical applications. The ability to relax these two constraints by one order of magnitude, for example, would result in a gain of four orders of magnitude: one for time and 3 for the 3 spatial dimensions.

Second, often plasmas present regions of high density in contact with regions of lower density, resulting in a large range of Debye lengths. To avoid the finite grid instability, uniform grids need to resolve the smallest Debye length in the system. Adaptive schemes can be devised to resolve in each region only the local Debye length, resulting in considerable savings [23, 29, 66]. However, even in this case, often the local scales of interest are much larger than the Debye length and the local grid resolution would be more desirably chosen according to other consideration than the need to resolve the Debye length.

1.6 Implicit Particle Methods

An approach to avoid the prohibitively small time steps and grid spacing required by explicit methods is to reduce the physics model used.

For example, the speed of light limitation can be eliminated by retaining only some of the full electromagnetic physics while removing from the Maxwell's equations the terms allowing the propagation of light. The approximation is called Darwin [16] and is used when slow-varying magnetic fields are present without any important role of electromagnetic waves.

Other approaches are based on eliminating some of the electron physics to eliminate the need for resolving the Debye length and the plasma frequency. If electrons are treated as fluids and neutrality is enforced, both conditions can be eliminated. The approach is called hybrid and can be implemented in varying degrees of complexity [45]. Another very important approximation is that of averaging over the gyration of particles in large magnetic fields. This approximation is called gyrokinetic and is widely used in fusion research and in many systems where large magnetic fields are present [43].

The other approach to eliminate the stability constraints is to retain the full first principle description of the electron physics and of the electromagnetic fields but to rely on more advanced numerical methods that can handle the presence of multiple scales in space and time. To avoid the prohibitively large number of time steps and the exceedingly high resolution needed to run realistic problems with explicit particle methods, implicit methods have been considered for several decades [51, 18, 8, 37].

The constraints of the explicit methods come from the artificial freezing of the coupling between fields and particles during the time step. The obvious solution is to reintroduce this coupling and solve the field equations on the grid and the particle equations maintaining their coupling. This coupling of course is non-linear as the particles are the sources of the current and density in the Maxwell's equations and the fields in the particle equations are produced by the Maxwell's equations.

In the years 70's, 80's and 90's, the consensus was that solving the full non-linear coupled system of field equations and particle equations was not feasible. But the situation has changed in recent years. First, computers are of course vastly more powerful. Second and more importantly, the progress made in the domain of Newton-Krylov solvers made this task possible [32, 33].

Two independent teams proposed fully implicit PIC codes[48, 11] capable of solving electrostatic, full electromagnetic [48], fully relativistic [42] or reduced Darwin descriptions of plasmas [10]. The key aspect of these methods is that they completely remove the need to satisfy any of the constraints above. Additionally and equally importantly, these methods are exactly energy conserving.

All previous PIC methods did not conserve energy. Explicit PIC methods can be readily formulated to conserve momentum exactly [25]. With respect to energy conservation, unfortunately the situation is far less rosy. The random aspect linked with the presence of discrete particles induces a tendency for the explicit PIC method to generate numerical heating: random fluctuations in the particle information leads to the generation of electromagnetic waves at small scales (noise). Energy is not only not conserved but it tends to increase in time [4, 28]. The increase tends to be secular at small time steps but it becomes exponential in presence of finite-grid instability [64]. The aforementioned stability constraints are then supplemented by the additional requirement of maintaining good energy conservation, typically limiting the time step by an additional order of magnitude to $\omega_{pe}\Delta t \sim 0.1$, but sometimes requiring even smaller time steps to retain an accurate description of particle energisation [42].

A class of so-called explicit "energy-conserving" PIC [44] (the quotation marks are used in the literature [36]) has been developed where the energy was conserved only in the limit of zero time step. Their peculiarity was the use of different interpolation functions for charge and current and for the magnetic and electric fields, resulting in improved energy conserva-

tion. In practice energy conservation was marginally better than in other explicit PIC, but still energy was by no means exactly conserved. For example, the code Celeste belonged to the class of "energy conserving" PIC of the old definition [67].

The new fully implicit PIC are exactly energy conserving, the acronym ECPIC (energy conserving PIC) is then properly deserved. One can prove that if the discretised equations defining the ECPIC are solved exactly, energy is conserved exactly. The ECPIC relies on a Newton iteration to solve the model equations. Of course, the Newton iteration is stopped at a given tolerance and the energy will then also have the same tolerance in the error. But the method is in principle exactly energy conserving. In practice one can easily achieve relative energy errors below 10^{-9} as in the examples below or any desired accuracy within machine precision. This accuracy is many orders of magnitude better than anything published before.

The basic common idea of the ECPIC methods is to replace the leap-frog algorithm with an implicit particle mover where the fields appear at the advanced time level, requiring an iteration with the field equations. For example, Ref.[48] proposes the θ-scheme:

$$\begin{aligned} \mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+\theta} \\ \mathbf{v}_p^{n+1} &= \mathbf{v}_p^n + \frac{q_p \Delta t}{m_p} \left(\mathbf{E}_p^{n+\theta}(\overline{\mathbf{x}}_p) + \overline{\mathbf{v}}_p \times \mathbf{B}_p^{n+\theta}(\overline{\mathbf{x}}_p) \right) \end{aligned} \tag{1.56}$$

where barred variables are average between time level n and $n+1$, $\overline{\mathbf{x}} = (\mathbf{x}^{n+1} + \mathbf{x}^n)/2$ and variables of time index $n+\theta$ are defined as $\mathbf{x}^{n+\theta} = \theta \mathbf{x}^{n+1} + (1-\theta)\mathbf{x}^n$. The same notation applies to all variables, particles and fields.

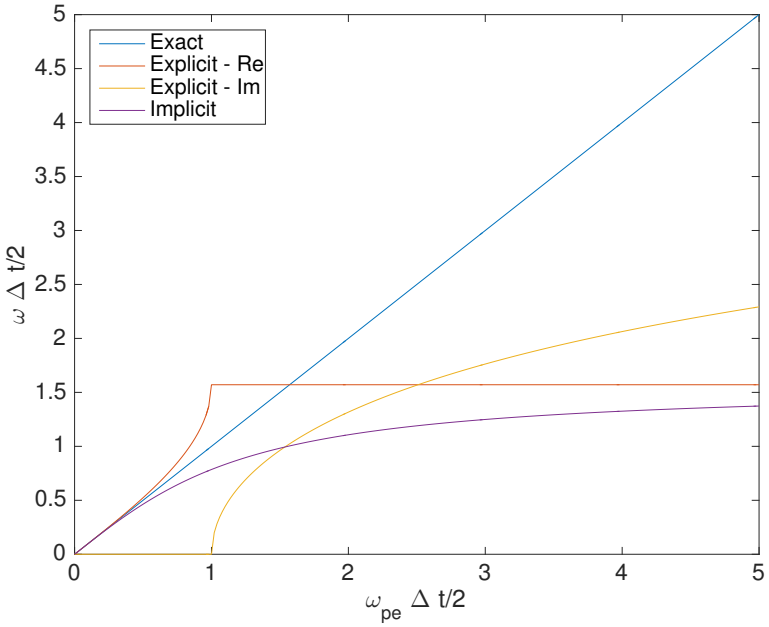

Figure 1.7: Results of the stability analysis of PIC particle movers. The explicit leap-frog mover is compared with the implicit mover in [48]. The explicit method is stable only when $\omega_{pe}\Delta t < 2$. For larger Δt, the numerical solution has an imaginary component of the frequency that leads to numerical growth. The implicit method is always stable.

The implicit scheme is unconditionally stable. If the same Von Neumann analysis done for the leap-frog scheme is repeated for eq. (1.56), the following relation is obtained:

$$\left(\frac{\omega_{pe}\Delta t}{2}\right)^2 = \tan^2\left(\frac{\omega\Delta t}{2}\right) \tag{1.57}$$

Now the tan function is unbounded and any values of Δt lead to real values for ω without any numerical instability. Figure 1.7 reports the stability analysis for explicit and implicit methods. The frequency of the numerical solution is compared with the actual correct analytical solution in presence of Langmuir waves. As Δt is increased, the numerical solution of the explicit method increases compared with the correct solution, until the stability limit is reached. When the limit is exceeded, an imaginary part to the the numerical frequency appears, leading to growing, numerically unstable solutions. The implicit method instead is always stable.

When $\theta = 1/2$, the method is exactly energy conserving [48, 11], provided the field and particle equations are solved to non-linear consistency within a Newton iteration.

Of course, accuracy still imposes bounds on the time step. Since the mover uses only one punctual value of the fields, if a particle travels more than one cell per time step the formula becomes inaccurate, leading to severe non-momentum conservation. Even this limitation is removed if sub-cycling is used [11].

As mentioned above, this modern solution was not available until recent times. Even today it remains computationally challenging and it requires the deployment of Newton-Krylov non-linear solvers, complicating significantly the optimisation of a massively parallel PIC codes. The use of non-linear solvers requires global communication and the storage of multiple states of the unknown to form the Krylov space orthogonalisation. When applied to a full particle-fields coupled system, this leads to extreme memory requirements. This problem can be circumvented with particle-enslavement and other preconditions [48, 12] but at the cost of further complexity in code development.

The solution adopted in the earlier days was that of linearising the particle-field coupling and solving the resulting linear system. This method is semi-implicit in the sense that the formulating equations are still fully coupled and require an iteration but the iteration is linear and not solved to non-linear consistency. Two approaches have been presented.

The direct implicit method, developed at Lawrence Livermore National Laboratory and implemented in codes such as AVANTI and the LSP [27, 37, 13, 20, 68]. This method is based on replacing the leap-frog algorithm with more advanced implicit movers. For example, the so-called D_1 algorithm (formulated with recursive filter on the fields [1]):

$$\begin{aligned} \mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1/2} \\ \mathbf{v}_p^{n+1/2} &= \mathbf{v}_p^{n-1/2} + \frac{q_p}{m_p}\left(\overline{\overline{\mathbf{E}}}_n + \frac{\mathbf{v}_p^{n+1/2} + \mathbf{v}_p^{n-1/2}}{2} \times \mathbf{B}^n(\mathbf{x}_p^n)\right) \end{aligned} \tag{1.58}$$

where the only difference with the standard leap-frog method of eq. (1.5) is the calculation of the electric field acceleration as:

$$\overline{\overline{\mathbf{E}}}_n = \left(\mathbf{E}_{n-1}(\mathbf{x}_n) + \mathbf{E}_{n+1}(\mathbf{x}_n)\right)/2 \tag{1.59}$$

A small formal difference but a huge numerical difference. The scheme now requires the new advanced field that can be calculated only after moving the particles. This non-linear iteration is avoided by the direct implicit method by computing the linear response of the plasma.

In the direct implicit method, the plasma response (i.e. the changes in the electric field in response to the particle motion) is obtained by linearising the particle equations of motion about an estimate (extrapolation) for their unknowns at the new time level $n+1$. The mathematical procedure used can vary from author to author, but it leads to a form of sensitivity matrix of the linearisation that assumes the form of a plasma susceptibility [7]. The field equations can then be formulated using this matrix and solved alone without iterating with the particles. After the new field is computed, the particles can be moved.

An alternative approach, also going back to the late 70's, is the implicit moment method (IMM), developed at Los Alamos National Laboratory[51, 8, 7]. In this approach, the linearisation is obtained by linearising the moments rather than the particle equations of motion. The method has been used in a family of codes originating with Venus [8], followed by Celeste [67, 38], Parsek2D [47] and now used in iPic3D [49].

The implicit moment method is based on a scheme similar to the θ-scheme described above in the fully implicit ECPIC method:

$$\begin{aligned} \mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+\theta} \\ \mathbf{v}_p^{n+1} &= \mathbf{v}_p^n + \frac{q_p \Delta t}{m_p}\left(\mathbf{E}^{n+\theta}(\overline{\mathbf{x}}_p) + \overline{\mathbf{v}}_p \times \mathbf{B}^n(\overline{\mathbf{x}}_p)\right) \end{aligned} \tag{1.60}$$

differing only in the time level of the magnetic field. As in the case of the leap-frog algorithm, eq. (1.61), the velocity equation can be rewritten in the equivalent form:

$$\overline{\mathbf{v}}_p = \widehat{\mathbf{v}}_p + \beta_s \widehat{\mathbf{E}}_p \tag{1.61}$$

where the same notation is used as in eq. (1.40-1.41) with the critical difference that now $\widehat{\mathbf{E}}_p = \alpha_p^n \mathbf{E}_p^{n+\theta}$ is based on the new advanced electric field that can only be known once the field equations are advanced.

The key defining property of the IMM is that the θ-scheme is now coupled to the Maxwell equations via a linearisation procedure. The two curl Maxwell equations are discretised in time with another θ-scheme:

$$\begin{aligned} \nabla_g \times \mathbf{E}^{n+\theta} + \frac{1}{c}\frac{\mathbf{B}_g^{n+1} - \mathbf{B}_g^n}{\Delta t} &= 0 \\ \nabla_g \times \mathbf{B}^{n+\theta} - \frac{1}{c}\frac{\mathbf{E}_g^{n+1} - \mathbf{E}_g^n}{\Delta t} &= \frac{4\pi}{c}\overline{\mathbf{J}}_g \end{aligned} \tag{1.62}$$

The spatial operators in eq. (1.62) are discretised on a grid. Consistent with the notation introduced earlier, we index the grid elements generically with the index g. ∇_g is a shorthand for the spatial discretisation used. For example in the IMM family of codes the discretisation used is different from the Yee scheme [69] and locates on the cell centres the magnetic field and on the cell vertices for the electric field [61, 49, 40]. But all the derivations below are not critically dependent on which spatial discretisation is used.

The current is computed as average of the old and new time level, requiring again to solve the particle equations of motion before the fields can be computed and vice-versa. This iteration is avoided by expanding the current in Taylor series truncated at the first order [67]. Computing the current separately for each species s, one obtains:

$$\bar{\mathbf{J}}_{sg} = \hat{\mathbf{J}}_{sg} - \frac{\Delta t}{2}\mu_{sg} \cdot E_\theta - \frac{\Delta t}{2}\nabla \cdot \hat{\Pi}_{sg} \tag{1.63}$$

where the following expressions were defined:

$$\begin{aligned} \hat{\mathbf{J}}_{sg} &= \textstyle\sum_p q_p \hat{\mathbf{v}}_p W(\mathbf{x}_g - \mathbf{x}_p^n) \\ \hat{\Pi}_{sg} &= \textstyle\sum_p q_p \hat{\mathbf{v}}_p \hat{\mathbf{v}}_p W(\mathbf{x}_g - \mathbf{x}_p^n) \end{aligned} \tag{1.64}$$

with the obvious meaning, respectively, of current and pressure tensor based on the hatted velocities.

An effective dielectric tensor is defined to express the feedback of the electric field on the plasma current and density:

$$\mu_{sg}^n = -\frac{q_s \rho_s^n}{m_s}\alpha_{sg}^n \tag{1.65}$$

where the rotation matrix α_s^n is defined in the same manner as that of the particles but based on the local field on the grid rather than the particle field:

$$\alpha_{sg}^n = \frac{1}{1+(\beta_s B_g^n)^2}\left(\mathbb{I} - \beta_s \mathbb{I} \times \mathbf{B}_g^n + \beta_s^2 \mathbf{B}_g^n \mathbf{B}_g^n\right) \tag{1.66}$$

With this approach, the current needed for the Maxwell equations can be computed using exclusively particle data from the time level n and no iteration is needed between particles and fields.

Regardless of the use of the direct implicit or IMM, two advantages arise in using semi-implicit methods.

First, these methods avoid any non-linear iteration. The resulting field equations obtained with the procedure described above are linear and can be solved with any linear solver. The three latest implementations of the IMM (Celeste, Parsek and iPic3D) use the GMRES approach [59, 58]. These methods are sill requiring global communication when implemented in parallel but converge very quickly due to the diagonal dominance of the two curl Maxwell equations. Furthermore these solvers deal only with the grid and the memory requirement is vastly smaller than that of an iteration scheme dealing also with particles [46, 34]. There are typically at least hundreds of particles per cell, the memory requirement of the iteration method is then reduced by that same factor.

Second, the semi-implicit method retains the same structure of the time step loop of the usual explicit PIC (see Fig. 1.6): the field equations are solved on the grid, the fields are interpolated to the particles, the particles are moved and then the moments on the grid are computed and the next cycle can continue. The field solver is somewhat more complex and the particle mover is also more complex but the overall scheme is the same as in explicit PIC.

It is important to realise what the added complexity is. The field solver, at first sight, is more complex than the explicit field solver of explicit PIC. In the simplest explicit PIC

methods, the fields are marched in a leap-frog manner and require no linear solver. However, even for the so-called charge conserving schemes the two divergence equations are not satisfied exactly requiring periodic elliptic solvers to apply the so-called divergence cleaning [54]. The need for elliptic solvers is completely eliminated in IMM because of the natural tendency of the IMM to damp any divergence error [58]. As noted above the application of GMRES to the two curl equations is much simpler and converges faster than for elliptic divergence cleaners [34].

The particle mover is also more complex. The mover in eq. (1.60) is not explicit in itself. The equation for the velocity requires to compute the fields at the position $\bar{\mathbf{x}}_p$ that is not know until the particle is moved, but to move the particle the velocity needs to be known. This requires a separate iteration for each particle. The iteration is just among the 6 equations of motions for each particle. All codes in the IMM family use a predictor correct approach that is a Picard-type iteration between the two Newton equations for each particle. The iteration can be truncated at a specified number (typical 3 is used) or until a convergence criterion is satisfied [57, 56]. As we will see below this iteration, even though it is local, still is an expensive operation because it is applied to each particle.

As a rule of thumb, compared with an explicit PIC the IMM is about 3 times more expensive per time step. The mover requires 3 steps where the leap-frog algorithm requires only one. The field solver is a small fraction of the cost. The IMM is not bound by the stability constraint and can use larger grid spacing and time steps, winning a large victory in terms of the time needed to complete the simulation of a given box size for a given time interval.

The IMM does not have to satisfy any stability constraints because its model equations are essentially the same as those of the ECPIC. A complete stability analysis of the IMM shows that waves and scales not resolved by a certain simulation are averaged and damped at the sub-grid level [8]. There is however a catch. The two great advantages of the semi-implicit methods compared with the ECPIC come at a great loss: energy is no longer conserved. The lack of full non-linear consistency breaks energy conservation and introduces an accuracy constraint. The average particle motion cannot exceed one cell per time step. This limitation comes mathematically from the need for the Taylor series expansion truncated at the first order used above to be a valid approximation. Physically, the requirement is that the moment description introduced by the series expansion can be a good approximation of the particle evolution. The procedure of the IMM can be regarded as a first order Chapman-Enskog expansion [9]. The IMM in this regards uses a fluid moment description to guess the plasma response to the changes happening to the particles during one time step. This description is not completely consistent with the actual particle evolution. To keep the error under control the time step becomes limited. The limit is usually expressed as $v_{the}\Delta t/\Delta x < 1$ [8]. If this condition is satisfied the method is accurate and the energy error is under control.

A new variant of the IMM has been recently developed to exactly conserve energy [41]. The new method is a variant of the IMM just described but introduces a new approach to compute the current to enforce exact energy conservation. The advantage is that the finite grid instability is completely eliminated. The method is very new at the moment of writing but it holds promise of being a quantum step forward. More research is needed to verify if this promise is realised in practice.

1.7 Annotated Python Code

To make it all more concrete, we present now a complete functioning PIC code. We choose the simplest case: 1D and electrostatic with electrons being followed as particles but ions forming a fixed background of uniform density. This last assumption is justified by the heavy mass of the ions. Once this code is fully understood going to full 3D electromagnetic is truly a simple step.

We choose the python language, but on the web site of the author other versions (classical, relativistic, electrostatic, electromagnetic, implicit, explicit) are available: https://perswww.kuleuven.be/~u0052182/. The graphical parts to output the data are also available on the web site but are not discussed here.

Initialisation

As a first step, some python initialisations are needed.

```
import numpy as np
import pylab as plt
from scipy import sparse
from scipy.sparse import linalg
```

We then need to initialise the definition of the simulation, in terms of domain size and of the grid used to discretise it. Next, the time step is set with the number of cycles to be run. The next step is to define the plasma density. We use normalised units where the plasma frequency is set to be unitary. But physical units can be used.

The ion uniform background is set to exactly balance the charge of the particles (that are all electrons in this simple model).

```
# Simulation parameters
L = 20*np.pi #20*np.pi # Domain size

NG = 80   # Number of grid cells
N = NG * 200 # Number of particles (200 per cell for example)
dx = L / NG # Cell size

DT = 0.005 # Time step
NT = 50000  # Number of time steps


WP = 1. # Plasma frequency
QM = -1. # Charge/mass ratio
Q = WP**2 / (QM*N/L)  # rho0*L/N: charge carried by a single particle
rho_back = -Q*N/L  # Background charge density
```

Particle Initialisation

We then initialise the electrons. We choose a classic problem of two-stream instability [24] where the initial electrons are subdivided into two equal beams of opposite mean velocity but equal density and thermal speed.

```
#Particle initial properties
V0 = 0.9 # Stream velocity
VT = 0.0000001 # Thermal speed

# perturbation
XP1 = 1.0
mode = 1

# particles (electrons)
xp = np.linspace(0, L-L/N, N).T   # Particle positions
vp = VT * np.random.randn(N) # Particle momentum, initially Maxwellian
pm = np.arange(N)
pm = 1 - 2 * np.mod(pm+1, 2) # Even and odd particles have opposite speed
vp += pm * V0  # Momentum + stream velocity
np.random.shuffle(v) # We reshuffle the indices to avoid any bias

# Add electron perturbation to excite the desired mode
xp += XP1 * (L/N) * np.sin(2 * np.pi * xp / L * mode)
xp[np.where(xp < 0)] += L
xp[np.where(xp >= L)] -= L
```

Grid Initialisation

In the 1D electrostatic limit, the Maxwell equations reduce to just the Poisson equation:

$$-\varepsilon_0 \nabla^2 \phi = en_i - en_e \tag{1.67}$$

where n_i is the uniform background ion density and n_e is the electron density projected to the grid from the particles. The ∇^2 operator is discretised as in the simplest textbook finite difference method [53]:

$$-\varepsilon_0(\phi_{i+1} + \phi_{i-1} - 2\phi_i) = (en_i - en_e)\Delta x^2 \tag{1.68}$$

The potential is computed in the cell centres. The continuum is subdivided in cells indexed by i with $i \in [0, \mathrm{NG}-1]$ (note than python as C and C++ counts the indices in vectors from 0, in MATLAB the same index range would be 1 to NG).

Of course, the first cell and the last cell have a problem: we do not have the neighbouring cell outside. This challenge is called *boundary condition* and it is the worst nightmare any computational scientist can experience. Boundary conditions are pure distilled evil. An evil we will avoid with a double trick. First, we use periodic boundary conditions: the left neighbour of the first cell is the last cells and the right neighbour of the last cell is the first

cell. Second, the potential is defined minus a constant and we use this freedom to set the potential in the last cell to 0. The number of unknown potentials then is just NG-1. This tricks works in 1D for periodic boundary conditions, more general boundary conditions in 3D quickly become the aforementioned nightmare.

With these assumptions, the matrix of the discretised Poisson equation is then very simple, it has -2 on the main diagonal and 1 on the two neighbouring ones.

```
# Auxiliary vectors
p = np.concatenate([np.arange(N), np.arange(N)])  # Particle indices up to N
Poisson = sparse.spdiags(([1, -2, 1] * np.ones((1, NG-1), dtype=int).T).T, \
                         [-1, 0, 1], NG-1, NG-1)
Poisson = Poisson.tocsc()
```

Main Cycle

At this point we have prepared the simulation for the main cycle where the four steps of pic are followed. First, the position is advanced, then the particles are projected to the grid to compute the density. With the density, the Poisson equation is solved. The electric field is computed to obtain the force on the particles and advance the particle velocity.

```
# Main cycle
for it in xrange(NT+1):
    # update particle position xp
    xp += vp * DT
    # Periodic boundary condition
    xp[np.where(xp < 0)] += L
    xp[np.where(xp >= L)] -= L

    # Project particles->grid
    csi = xp/dx
    g1 = np.floor(csi - 0.5) # Distance from the centre of the cell
    g = np.concatenate((g1, g1+1))
    fraz1 = 1 - np.abs(xp/dx - g1 - 0.5)
    fraz = np.concatenate((fraz1, 1-fraz1))
    g[np.where(g < 0)] += NG
    g[np.where(g > NG-1)] -= NG
    mat = sparse.csc_matrix((fraz, (p, g)), shape=(N, NG))
    rho = Q / dx * mat.toarray().sum(axis=0) + rho_back

    # Compute electric field potential
    Phi = linalg.spsolve(Poisson, -dx**2 * rho[0:NG-1])
    Phi = np.concatenate((Phi,[0]))

    # Electric field on the grid
    Eg = (np.roll(Phi, 1) - np.roll(Phi, -1)) / (2*dx)
```

```
# interpolation grid->particle and velocity update
vp += mat * QM * Eg * DT
```

Let us comment the particle steps in each of these phases:

1. *Position advancement*: When particles are advanced, the possibility exists for the particles to exit the system. Consistent with the periodicity used above, also particles exiting form the left re-enter on the right and vice versa.

2. *Particle projection*: We introduce a *logical coordinate* defined as $\xi = x/\Delta x$. The particles and the cells have logical coordinates, where the interpolation function is more easily defined as $W_{ip} = b_\ell(\xi_p - \xi_i)$. Cell centres have coordinates $\xi_i = (i+1/2)$, $i \in [0, \mathrm{NG}-1]$ and particles have $\xi_p = x/\Delta x$. We consider the case of particle shapes given by b-splines of order 0 and interpolation functions, consequently, of order 1:

$$W_{ip} = b_1(\xi_p - \xi_i) \equiv \begin{cases} 1 - |\xi_p - \xi_i|, \text{ if} : |\xi_p - \xi_i| < 1 \\ \\ 0, \text{ otherwise} \end{cases} \tag{1.69}$$

With this choice of interpolation, each particle can only contribute to maximum of two cells. The choice of making the particle size equal to the cell size means that it is impossible for a particle to overlap more than 2 cells. Figure 1.8 shows the concept. The contribution of a particle of finite length to a cell is equal to the fraction of the

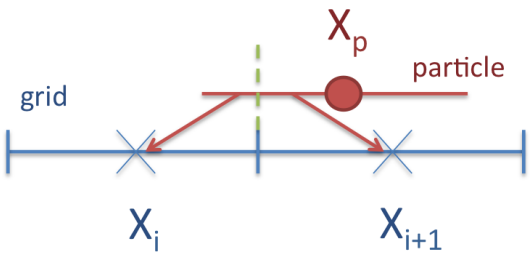

Figure 1.8: Particle interpolation in 1D: the contribution of a particle to a cell is proportional to the length of the interval of overlap with that cell. When the particle shape is a b-spline of order zero (a case called cloud-in-cell), the particle is subdivided to the cells according to its overlap.

length of the particle that overlaps that cell. This is computed by first identifying the leftmost cell using the floor command and then computing the two fractions (one being 1 minus the other since the total contribution is unity).

Once the two fractions are computed the contribution to the charge of a cell is the fraction times the charge of the particle. Periodic boundary conditions are applied also to charge projection.

The code in the example uses a matrix notation to project the particle avoiding loops but a simple loop over the particles would work as well. The matrix ”mat” is equal to

the interpolation function $W_{ip} = W(x_p - x_i)$ that indeed has two indexes. The charge is than the matrix W_{ip} applied to the vector q_p:

$$q_i = W\mathbf{q} \equiv \sum_p W_{ip} q_p \tag{1.70}$$

The matrix notation can in some computer languages produce a faster execution, but this issue is largely eliminated in the latest version of most languages. The computational performance of the interpolation steps depends dramatically on the precise style of code writing, on the language and on the hardware [50, 6, 65, 15].

In alternative to a matrix formulation, a loop over the particles can be used:

```
qodx = Q / dx
rho = np.zeros(NG)
for i in range(N):
      csi = xp[i]/dx
      g1 = np.floor(csi - 0.5)
      g2 = g1 + 1
      fraz1 = 1 - np.abs(csi - g1 - 0.5)
      fraz2 = 1.0 - fraz1
      rho[j1] = rhoe[j1] + qodx*fraz1
      rho[j2] = rhoe[j2] + qodx*fraz2
rho[0] += rho[NG-1]
```

In both cases the charge in the cell must be divided by Δx to obtain the density.

3. *Field solver*: In 1D electrostatic there are many alternatives. Some are very simple [28], but we use here the most general approach that would work also with more complex boundary conditions and in more dimensions. We use a linear matrix solver provided by python. The boundary conditions are then imposed (setting the last cell to 0) and the electric field is computed from the potential:

$$E_i = -\frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x} \tag{1.71}$$

4. *Velocity update*: The electric field is used to advance the particle position. The same interpolation method used for the charge is used for the electric field.

Note that we have chosen the position update as our initial step in the cycle but the starting point is completely arbitrary, the cycle can start at any point. This means also that logically position and velocity advancement happen in sequence, the last operation of one cycle being followed by the first of the next.

Bibliography

[1] DC Barnes, T Kamimura, J-N Leboeuf, and T Tajima. Implicit particle simulation of magnetized plasmas. *Journal of Computational Physics*, 52(3):480–502, 1983.

[2] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324:4446–449, 1986.

[3] William B. Bateson and Dennis W. Hewett. Grid and particle hydrodynamics:: Beyond hydrodynamics via fluid element particle-in-cell. *Journal of Computational Physics*, 144(2):358–378, 1998.

[4] C.K. Birdsall and A.B. Langdon. *Plasma Physics Via Computer Simulation*. Taylor & Francis, London, 2004.

[5] C. De Boor. *A practical guide to splines*. Springer, 1978.

[6] K. J. Bowers, B. J. Albright, L. Yin, W. Daughton, V. Roytershteyn, B. Bergen, and T. J. T. Kwan. Advances in petascale kinetic plasma simulation with VPIC and Roadrunner. *Journal of Physics Conference Series*, 180(1):012055–+, July 2009.

[7] Jeremiah U Brackbill and Bruce I Cohen. *Multiple time scales*, volume 3. Academic Press, 1985.

[8] J.U. Brackbill and D.W. Forslund. Simulation of low frequency, electromagnetic phenomena in plasmas. *J. Computat. Phys.*, 46:271, 1982.

[9] Sydney Chapman and Thomas George Cowling. *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge university press, 1970.

[10] Guangye Chen and Luis Chacon. A multi-dimensional, energy-and charge-conserving, nonlinearly implicit, electromagnetic vlasov–darwin particle-in-cell algorithm. *Computer Physics Communications*, 197:73–87, 2015.

[11] Guangye Chen, Luis Chacón, and Daniel C Barnes. An energy-and charge-conserving, implicit, electrostatic particle-in-cell algorithm. *Journal of Computational Physics*, 230(18):7018–7036, 2011.

[12] Guangye Chen, Luis Chacón, Christopher A Leibs, Dana A Knoll, and William Taitano. Fluid preconditioning for newton–krylov-based, fully implicit, electrostatic particle-in-cell simulations. *Journal of Computational Physics*, 258:555–567, 2014.

[13] B. I. Cohen, A. B. Langdon, D. W. Hewett, and R. J. Procassini. Performance and Optimization of Direct Implicit Particle Simulation. *Journal of Computational Physics*, 81:151–+, March 1989.

[14] G. G. M. Coppa, G. Lapenta, G. Dellapiana, F. Donato, and V. Riccardo. Blob method for kinetic plasma simulation with variable-size particles. *Journal of Computational Physics*, 127(2):268–284, 1996.

[15] Viktor K Decyk. Skeleton particle-in-cell codes on emerging computer architectures. *Computing in Science & Engineering*, 17(2):47–52, 2015.

[16] Pierre Degond and Pierre-Arnaud Raviart. An analysis of the darwin model of approximation to maxwell?s equations. In *Forum Mathematicum*, volume 4, pages 13–44, 1992.

[17] Gian Luca Delzanno, Enrico Camporeale, J David Moulton, Joseph E Borovsky, Elizabeth A MacDonald, and Michelle F Thomsen. Cpic: A curvilinear particle-in-cell code for plasma–material interaction studies. *Plasma Science, IEEE Transactions on*, 41(12):3577–3587, 2013.

[18] J. Denavit. Time-filtering particle stimulations with $\omega_{pe}\Delta t \gg 1$. *J. Computat. Phys.*, 42:337, 1981.

[19] Richard O Dendy. *Plasma dynamics*. Oxford University Press, Oxford, UK, 1990.

[20] M. Drouin, L. Gremillet, J.-C. Adam, and A. Héron. Particle-in-cell modeling of relativistic laser-plasma interaction with the adjustable-damping, direct implicit method. *Journal of Computational Physics*, 229(12):4781–4812, 2010.

[21] T.Zh. Esirkepov. Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor. *Computer Physics Communications*, 135(2):144 – 153, 2001.

[22] D. Frenkel and B. Smit. *Understanding Molecular Simulation*. Academic Press, San Diego, 2002.

[23] K. Fujimoto and R. D. Sydora. Electromagnetic particle-in-cell simulations on magnetic reconnection with adaptive mesh refinement. *Computer Physics Communications*, 178:915–923, June 2008.

[24] R.J. Goldston and P.H. Rutherford. *Introduction to plasma physics*. Taylor & Francis, 1995.

[25] Yu. N. Grigoryev, V. A. Vshivkov, and M. P. Fedoruk. *Numerical Particle-in-Cell Methods: Theory and Applications*. de Gruyter, 2002.

[26] Yu. N. Grigoryev, Vitali Andreevich Vshivkov, and Mikhail Petrovich Fedoruk. *Numerical particle-in-cell methods: theory and applications*. VSP BV, AH Zeist, 2005.

[27] D. W. Hewett and A. B. Langdon. Electromagnetic direct implicit plasma simulation. *Journal of Computational Physics*, 72:121–155, September 1987.

[28] R.W. Hockney and J.W. Eastwood. *Computer simulation using particles*. Taylor & Francis, 1988.

[29] Maria Elena Innocenti, Giovanni Lapenta, Stefano Markidis, Arnaud Beck, and Alexander Vapirev. A multi level multi domain method for particle in cell plasma simulations. *Journal of Computational Physics*, 238:115–140, 2013.

[30] Eugene Isaacson and Herbert Bishop Keller. *Analysis of numerical methods*. Courier Corporation, 1994.

[31] G. B. Jacobs and J. S. Hesthaven. High-order nodal discontinuous Galerkin particle-in-cell method on unstructured grids. *Journal of Computational Physics*, 214:96–121, May 2006.

[32] Carl T Kelley. *Solving nonlinear equations with Newton's method*, volume 1. Siam, 2003.

[33] Dana A Knoll and David E Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[34] Pawan Kumar, Stefano Markidis, Giovanni Lapenta, Karl Meerbergen, and Dirk Roose. High performance solvers for implicit particle in cell simulation. *Procedia Computer Science*, 18:2251–2258, 2013.

[35] A. B. Langdon. On enforcing Gauss' law in electromagnetic particle-in-cell codes. *Computer Physics Communications*, 70:447–450, July 1992.

[36] A Bruce Langdon. "Energy-conserving" plasma simulation algorithms. *Journal of Computational Physics*, 12(2):247–268, 1973.

[37] A.B. Langdon, BI Cohen, and A Friedman. Direct implicit large time-step particle simulation of plasmas. *J. Computat. Phys.*, 51:107–138, 1983.

[38] G. Lapenta, J. U. Brackbill, and P. Ricci. Kinetic approach to microscopic-macroscopic coupling in space and laboratory plasmas. *Physics of Plasmas*, 13(5):055904, May 2006.

[39] Giovanni Lapenta. Democritus: An adaptive particle in cell (pic) code for object-plasma interactions. *Journal of Computational Physics*, 230(12):4679–4695, 2011.

[40] Giovanni Lapenta. Particle simulations of space weather. *Journal of Computational Physics*, 231(3):795–821, 2012.

[41] Giovanni Lapenta. Exactly energy conserving implicit moment particle in cell formulation. *arXiv preprint arXiv:1602.06326*, 2016.

[42] Giovanni Lapenta and Stefano Markidis. Particle acceleration and energy conservation in particle in cell simulations. *Physics of Plasmas (1994-present)*, 18(7):072101, 2011.

[43] WW Lee. Gyrokinetic particle simulation model. *Journal of Computational Physics*, 72(1):243–269, 1987.

[44] H Ralph Lewis. Energy-conserving numerical approximations for vlasov plasmas. *Journal of Computational Physics*, 6(1):136–141, 1970.

[45] Alexander S. S. Lipatov. *The Hybrid Multiscale Simulation Technology*. Springer, Berlin, 2002.

[46] Damian Alvarez Mallon, Norbert Eicker, Maria Elena Innocenti, Giovanni Lapenta, Thomas Lippert, and Estela Suarez. On the scalability of the clusters-booster concept: a critical assessment of the deep architecture. In *Proceedings of the Future HPC Systems: the Challenges of Power-Constrained Performance*, page 3. ACM, 2012.

[47] S. Markidis, E. Camporeale, D. Burgess, Rizwan-Uddin, and G. Lapenta. Parsek2D: An Implicit Parallel Particle-in-Cell Code. In N. V. Pogorelov, E. Audit, P. Colella, & G. P. Zank, editor, *Astronomical Society of the Pacific Conference Series*, volume 406 of *Astronomical Society of the Pacific Conference Series*, pages 237–+, April 2009.

[48] Stefano Markidis and Giovanni Lapenta. The energy conserving particle-in-cell method. *Journal of Computational Physics*, 230(18):7037–7052, 2011.

[49] Stefano Markidis, Giovanni Lapenta, and Rizwan-uddin. Multi-scale simulations of plasma with ipic3d. *Mathematics and Computers in Simulation*, 80(7):1509–1519, 2010.

[50] Stefano Markidis, Giovanni Lapenta, WB VanderHeyden, and Z Budimlić. Implementation and performance of a particle-in-cell code written in java. *Concurrency and Computation: Practice and Experience*, 17(7-8):821–837, 2005.

[51] R.J. Mason. Implicit moment particle simulation of plasmas. *J. Computat. Phys.*, 41:233, 1981.

[52] R. L. Morse and C. W. Nielson. Numerical Simulation of the Weibel Instability in One and Two Dimensions. *Physics of Fluids*, 14:830–840, April 1971.

[53] K.W Morton and D.F. Mayers. *Iterative methods for linear and nonlinear equations*. SIAM, Philadelphia, 1995.

[54] C.-D. Munz, P. Omnes, R. Schneider, E. Sonnendrücker, and U. Voß. Divergence Correction Techniques for Maxwell Solvers Based on a Hyperbolic Model. *Journal of Computational Physics*, 161:484–511, July 2000.

[55] D.R. Nicholson. *Introduction to plasma theory*. Wiley, 1983.

[56] K. Noguchi, C. Tronci, G. Zuccaro, and G. Lapenta. Formulation of the relativistic moment implicit particle-in-cell method. *Physics of Plasmas*, 14(4):042308–+, April 2007.

[57] Ivy Bo Peng, Stefano Markidis, Andris Vaivads, Juris Vencels, Jorge Amaya, Andrey Divin, Erwin Laure, and Giovanni Lapenta. The formation of a magnetosphere with implicit particle-in-cell simulations. *Procedia Computer Science*, 51:1178–1187, 2015.

[58] P. Ricci, G. Lapenta, and J.U. Brackbill. A simplified implicit maxwell solver. *J. Computat. Phys.*, 183:117–141, 2002.

[59] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[60] John K Salmon and Michael S Warren. Skeletons from the treecode closet. *Journal of Computational Physics*, 111(1):136–155, 1994.

[61] Deborah Sulsky and JU Brackbill. A numerical method for suspension flow. *Journal of Computational Physics*, 96(2):339–368, 1991.

[62] A. Taflove and Susan C. Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Artech House Publishers, Norwood, 2005.

[63] Vidar Thomée. *Galerkin finite element methods for parabolic problems*, volume 1054. Springer, 1984.

[64] Hiroko Ueda, Yoshiharu Omura, Hiroshi Matsumoto, and Takashi Okuzawa. A study of the numerical heating in electrostatic particle simulations. *Computer physics communications*, 79(2):249–259, 1994.

[65] Alexander Vapirev, Jan Deca, Giovanni Lapenta, Stefano Markidis, I Hur, and J-L Cambier. Initial results on computational performance of intel many integrated core, sandy bridge, and graphical processing unit architectures: implementation of a 1d c++/openmp electrostatic particle-in-cell code. *Concurrency and Computation: Practice and Experience*, 27(3):581–593, 2015.

[66] J-L Vay, Phillip Colella, Alex Friedman, David P Grote, Peter McCorquodale, and DB Serafini. Implementations of mesh refinement schemes for particle-in-cell plasma simulations. *Computer physics communications*, 164(1):297–305, 2004.

[67] H. X. Vu and J. U. Brackbill. Celest1d: An implicit, fully-kinetic model for low-frequency, electromagnetic plasma simulation. *Comput. Phys. Comm.*, 69:253, 1992.

[68] Dale R Welch, David V Rose, Robert E Clark, Tom C Genoni, and TP Hughes. Implementation of an non-iterative implicit electromagnetic field solver for dense plasma simulation. *Computer physics communications*, 164(1):183–188, 2004.

[69] Kane S Yee and Jei S Chen. The finite-difference time-domain (fdtd) and the finite-volume time-domain (fvtd) methods in solving maxwell's equations. *Antennas and Propagation, IEEE Transactions on*, 45(3):354–363, 1997.