Contrib. Plasma Phys. **47**, No. 8-9, 563 – 594 (2007) / **DOI** 10.1002/ctpp.200710072

The Particle-In-Cell Method

D. Tskhakaya[*1,4], **K. Matyash**[2], **R. Schneider**[2], and **F. Taccogna**[3]

[1] Association Euratom-ÖAW, Institute of Theoretical Physics, University of Innsbruck, A-6020 Innsbruck, Austria
[2] Max-Planck-Institut für Plasmaphysik, EURATOM Association, Wendelsteinstrasse 1, D-17491, Greifswald, Germany
[3] Dipartimento di Chimica, Universita' degli Studi di Bari and Istituto di Metodologie Inorganiche e di Plasmi - CNR, Sect. Bari, via Orabona 4, Bari, 70126, Italy
[4] Permanent address: Institute of Physics, Georgian Academy of Sciences, 380076 Tbilisi, Georgia

Received 19 October 2007, accepted 5 November 2007
Published online 14 December 2007

Key words PIC simulations, MC simulations.
PACS 52.65.-y, 52.65.Rr

This paper is the first in a series of three papers to summarize the recent work of an European-wide collaboration which is ongoing since about one decade using Particle-in-Cell (PIC) methods in low temperature plasma physics. In the present first paper the main aspects of this computational technique will be presented. In the second paper, an overview of applications in low-temperature plasma modelling will be given, whereas the third part will put emphasis on the specific results of modelling ion thrusters.

1 Introduction

Particle-in-Cell (PIC) simulation represents a powerful tool for plasma studies having a number of advantages like the fully kinetic description of high-dimensional plasma and the ability to incorporate complicated atomic and plasma-surface interactions. PIC simulations are used practically in all branches of laboratory and astrophysical plasma physics. PIC era started in the late 1950s by Buneman [1] and Dawson [2] who simulated the motion of 100–1000 particles including interactions between them. Our day PIC codes can simulate up to 10^{10} particles and represent complicated numerical tools requiring high level of optimization. As a result the PIC codes are usually designed for a professional use.

The aim of the present paper is to introduce the reader to the basics of the PIC simulation technique. The paper is based on our previous work [3]. For the interested reader we can recommend two classical monographs, [4] and [5].

The paper is organized as follows. The main structure of the PIC algorithm is discussed in Sec. 2. In Sec. 3 we consider solvers of equations of motion. We discuss the accuracy and stability aspects of these solvers. Initialization of particle distribution, boundary effects and particle sources are described in Sec. 4. In Secs. 5 and 6 we show how plasma macro-parameters are calculated and discuss solvers of Maxwell's equations. In Sec. 7 we consider a linear theory of finite size particle plasmas. Different models of particle collisions are considered in Sec. 8. Optimization of PIC codes and Grid-free methods are briefly considered in Sec. 10.

2 General scheme of PIC simulation

The PIC simulation is based on a trivial idea: The code simulates the motion of each plasma particle and calculates all macro-quantities (like density, current density and so on) from the position and velocity of these particles. The macro-force acting on the particles is calculated from the field equations. The name *Particle-in-Cell* comes from

*Corresponding author: e-mail: david.tskhakaya@uibk.ac.at
Phone: +43 512 507 6225, Fax: +43 512 507 2919

the way of assigning macro-quantities to the simulation particles. In general, any numerical simulation model, which simultaneously solves the equations of motion of N *particles*,

$$\frac{d\vec{X}_i}{dt} = \vec{V}_i \qquad \text{and} \qquad \frac{d\vec{V}_i}{dt} = \vec{F}_i\left(t, \vec{X}_i, \vec{V}_i, A\right) \tag{1}$$

for $i = 1, \ldots, N$ and of macro fields $A = L_1(B)$, with the prescribed rule of calculation of macro quantities $B = L_2\left(\vec{X}_1, \vec{V}_1, \ldots, \vec{X}_N, \vec{V}_N\right)$ from the particle position and velocity can be called a PIC simulation. Here $\vec{X}_i$ and $\vec{V}_i$ are the generalized (multi-dimensional) coordinate and velocity of the particle i. A and B are macro fields acting on particles and some macro-quantities associated with particles, respectively. L_1 and L_2 are some operators and $\vec{F}_i$ is the force acting on a particle i. As one can see, PIC simulations have much broader applications then just plasma physics, like solid state or quantum physics. On the other hand, inside the plasma community PIC codes are usually associated with codes solving the equation of motion of particles with the Newton–Lorentz's force (for simplicity we consider an unrelativistic case)

$$\frac{d\vec{X}_i}{dt} = \vec{V}_i \qquad \text{and} \qquad \frac{d\vec{V}_i}{dt} = \frac{e_i}{m_i}\left(\vec{E}\left(\vec{X}_i\right) + \vec{V}_i \times \vec{B}\left(\vec{X}_i\right)\right) \tag{2}$$

for $i = 1, \ldots, N$ and the Maxwell's equations:

$$\begin{aligned} &\nabla\vec{D} = \rho\left(\vec{r}, t\right) , \qquad && \frac{\partial\vec{B}}{\partial t} = -\vec{\nabla}\times\vec{E} , && \vec{D} = \varepsilon\vec{E} , \\ &\nabla\vec{B} = 0 , && \frac{\partial\vec{D}}{\partial t} = \vec{\nabla}\times\vec{H} - \vec{J}\left(\vec{r}, t\right) , && \vec{B} = \mu\vec{H} , \end{aligned} \tag{3}$$

together with the prescribed rule of calculation of ρ and $\vec{J}$

$$\rho = \rho\left(\vec{X}_1, \vec{V}_1, \ldots, \vec{X}_N, \vec{V}_N\right) , \tag{4}$$

$$\vec{J} = \vec{J}\left(\vec{X}_1, \vec{V}_1, \ldots, \vec{X}_N, \vec{V}_N\right) . \tag{5}$$

Here ρ and $\vec{J}$ are the charge and current densities and ε and μ the permittivity and permeability of the medium, respectively. Below we will follow this definition of the PIC codes.

PIC codes usually are classified depending on the dimensionality of the code and on the set of Maxwell's equations used. The codes solving a whole set of Maxwell's equations are called electromagnetic codes, contrary electrostatic ones solve just the Poisson equation. The dimensionality of the PIC code is usually given as mDnV, where m and n define dimensionality in usual and velocity spaces, respectively. Some advanced codes are able to switch between different dimensionality and coordinate system, and use electrostatic, or electro-magnetic models (e.g. the XOOPIC code [6]).

A simplified scheme of the PIC simulation is given in Fig. 1. PIC simulation starts with an initialization and ends with the output of results. This part is similar to the input/output routines of any other numerical tool. Other parts represent specific PIC routines, which we consider separately.

3 Integration of Equations of Particle Motion

3.1 Description of Particle Movers

During a PIC simulation the trajectory of all particles is followed, which requires the solution of the equations of motion for each of them. This part of the code is frequently called *particle mover*.

The number of particles in a real plasma is extremely large and exceeds by orders of magnitude the maximum possible number of particles, which can be handled by the best supercomputers. Hence, during a PIC simulation it is usually assumed that one simulation particle consists of many physical particles. Because the ratio charge/mass is invariant to this *transformation*, this *superparticle* follows the same trajectory as the corresponding plasma particle. As a result the plasma model simulated by a superpaticle is absolutely similar to a real one (assuming that

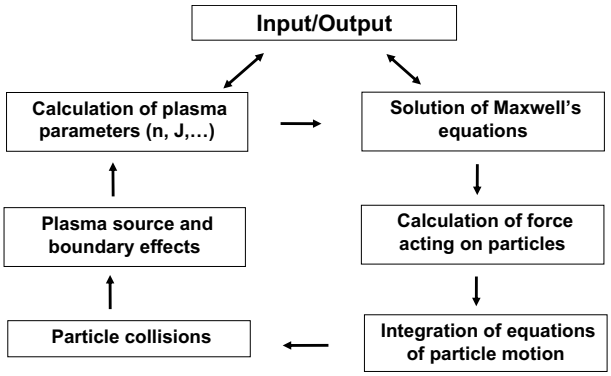

Fig. 1 Scheme of the PIC simulation.

all necessary parameters are properly re-scaled). One has to note that for 1D and 2D models this transformation can be easily avoided by choosing a sufficiently small simulated volume, so that the number of real plasma particles can be chosen arbitrary.

As we will see below, the number of simulated particles is defined by a set of physical and numerical restrictions, and usually it is extremely large ($> 10^5$). As a result, the main requirements to the particle mover are high accuracy and speed. One of such solvers represents the so called leap-frog method (see [4] and [5]), which we will consider in detail.

As in other numerical codes the time in PIC is divided into discrete time steps, in other words the time is gridded. This means that physical quantities are calculated only at given time steps. Usually, the interval between the time steps Δt, is constant, so that the simulated time can be given as: $t \to t_k = t_0 + k\Delta t$ and $A(t) \to A_k = A(t = t_k)$ with $k = 0, 1, 2, \ldots$, where t is the time, t_0 the initial time and A denotes any physical quantity. The leap-frog method calculates particle velocities not at usual time steps t_k, but between them $t_{k+1/2} = t_0 + (k + 1/2)\,\Delta t$. In this way equations become time-centered, so that they are sufficiently accurate and require relatively short calculation time:

$$\begin{aligned} \frac{\vec{X}_{k+1} - \vec{X}_k}{\Delta t} &= \vec{V}_{k+1/2}\ , \\ \frac{\vec{V}_{k+1/2} - \vec{V}_{k-1/2}}{\Delta t} &= \frac{e}{m}\left(\vec{E}_k + \frac{\vec{V}_{k+1/2} + \vec{V}_{k-1/2}}{2} \times \vec{B}_k\right)\ . \end{aligned} \tag{6}$$

The leap-frog scheme is an explicit solver, i.e. it depends on *old* forces from the previous time step k. Contrary to implicit schemes, when for calculation of particle velocity only quantities at the time step $k + 1$ are used, explicit solvers are simpler and faster, but their stability requires a smaller time step Δt.

Some remarks about the accuracy of this leap-frog solver. By substituting

$$\begin{aligned} \vec{V}_{k\pm1/2} &= \vec{V}_k \pm \frac{\Delta t}{2}\vec{V}'_k + \frac{\Delta t^2}{8}\vec{V}''_k \pm \frac{1}{6}\left(\frac{\Delta t}{2}\right)^3 \vec{V}'''_k + \ldots\ , \\ \vec{X}_{k+1} &= \vec{X}_k + \Delta t\vec{V}_k + \frac{\Delta t^2}{2}\vec{V}'_k + \frac{\Delta t^3}{6}\vec{V}''_k + \ldots \end{aligned} \tag{7}$$

into (6) we obtain the order of the error $\sim \Delta t^2$. It satisfies a general requirement for the scaling of numerical accuracy $\Delta t^{a>1}$. In order to understand this requirement we recall that for a fixed simulated time the number of simulated time steps scales as $N_t \sim \Delta t^{-1}$. Then, after N_t time steps an accumulated total error will scale as $N_t \Delta t^a \sim \Delta t^{a-1}$, where Δt^a is the scale of the error during one step. Thus, only $a > 1$ can guarantee, that the accuracy increases with decreasing Δt.

There exist different methods for solving finite-difference equations (6). Below we consider the Boris method (see [4]), which is frequently used in PIC codes:

$$\vec{X}_{k+1} = \vec{X}_k + \Delta t\vec{V}_{k+1/2} \qquad \text{and} \qquad \vec{V}_{k+1/2} = \vec{u}_+ + q\vec{E}_k\ , \tag{8}$$

with $\vec{u}_+ = \vec{u}_- + (\vec{u}_- + (\vec{u}_- \times \vec{h})) \times \vec{s}$, $\vec{u}_- = \vec{V}_{k-1/2} + q\vec{E}_k$, $\vec{h} = q\vec{B}_k$, $\vec{s} = 2\vec{h}/(1+h^2)$ and $q = e\Delta t/2m$. Although these equations look very simple, their solution represents the most time consuming part of PIC, because it is done for each particle separately. As a result, the optimization of the particle mover can significantly reduce the simulation time. One example for such an optimization is considered below.

In general, the Boris method requires 39 operations (18 adds and 21 multiplies), assuming that $\vec{B}$ is constant and $\vec{h}$, $\vec{s}$ and q are calculated only once at the beginning of simulation. But if $\vec{B}$ has one or two non-zero components, then the number of operations can be significantly reduced. E.g., if $\vec{B} \parallel \vec{z}$ and $\vec{E} \parallel \vec{x}$ then (8) can be reduced to the following ones

$$\begin{aligned} \vec{X}_{k+1} &= \vec{X}_k + \Delta t \vec{V}_{k+1/2} \ , \\ V^x_{k+1/2} &= u^x_- + \left(V^y_{k+1/2} + V^y_{k-1/2}\right) h + qE^x_k \ , \\ V^y_{k+1/2} &= V^y_{k-1/2}\,(1 - sh) - u^x_- s \end{aligned} \tag{9}$$

with $u^x_- = V^x_{k-1/2} + qE^x_k$. They require just 17 operations (8 multiplies and 9 adds), which can save up to 50% of the CPU time. Some advanced PIC codes include a subroutine for searching the fastest solver for a given simulation setup, which significantly decreases the CPU time (e.g. the BIT1 code [7]).

3.2 Accuracy and Stability of the Particle Mover

In order to find correct simulation parameters one has to know the absolute accuracy and corresponding stability conditions for the particle mover. They are different for different movers and the example considered below is applied just to the Boris scheme.

First of all let us consider the accuracy of a Larmor rotation. By assuming $\vec{V}_{k-1/2} \perp \vec{B}$ we can define the rotation angle during the time Δt from the following expression:

$$\cos(\omega\Delta t) = \frac{\vec{V}_{k+1/2}\vec{V}_{k-1/2}}{V^2_{k-1/2}} \ . \tag{10}$$

On the other hand, substituting (8) into (10) we obtain

$$\frac{\vec{V}_{k+1/2}\vec{V}_{k-1/2}}{V^2_{k-1/2}} = \frac{1 - \frac{(\Delta t\Omega)^2}{4}}{1 + \frac{(\Delta t\Omega)^2}{4}} \tag{11}$$

with $\Omega = eB/m$, so that for a small Δt we get $\omega = \Omega(1 - 1/12(\Delta t\Omega)^2) + \ldots$. E.g., for a 1% accuracy the following condition has to be satisfied: $\Delta t\Omega \leq 0.35$.

In order to formulate the general stability condition some complicated calculations are required (see [5]). Below we present simple estimates of the stability criteria for the (explicit) particle mover.

Let us consider the equation of a linear harmonic oscillator:

$$\frac{d^2X}{dt^2} = -\omega_0^2 X \ , \tag{12}$$

having the following analytic solution

$$X = Ae^{-i\omega_0 t} \ , \tag{13}$$

where A is an arbitrary imaginary number. The corresponding Leap-frog equations take the following form:

$$\frac{X_{k+1} - 2X_k + X_{k-1}}{\Delta t^2} = -\omega_0^2 X_k \ . \tag{14}$$

We assume that the solution has a form similar to (13), $X_k = A\exp(-i\omega t_k)$. After substitution into (14) and performing simple transformations we find

$$\sin\left(\frac{\omega\Delta t}{2}\right) = \pm\frac{\omega_0\Delta t}{2} \ . \tag{15}$$

Hence, for a stable solution $\mathrm{Im}(\omega) \leq 0$ the condition

$$\omega_0 \Delta t < 2 \tag{16}$$

is required. Frequently during PIC simulations much more restrictive condition is used:

$$\omega_0 \Delta t \leq 0.2 \,, \tag{17}$$

giving sufficiently accurate results. Interesting to note that this number has been derived few decades ago when the number of simulation time steps was typically of the order of $N_t \sim 10^4$. From (15) we obtain $\omega = \omega_0(1 - 1/24(\omega_0\Delta t)^2) + \ldots$. Hence, a cumulative phase error after N_t steps should be $\Delta(\omega\Delta t) \approx (N_t(\omega_0\Delta t)^3)/24$. Assuming $N_t = 10^4$ and $\Delta\,(\omega\Delta t) < \pi$ we obtain the condition (17). Although modern simulations contain much larger number of time steps up to $N_t = 10^7$, this condition still can work surprisingly well.

The restrictions on Δt described above can require the simulation of unacceptably large number of time steps. In order to avoid these restrictions different implicit schemes have been introduced: $\vec{V}_{k+1/2} = \vec{F}(\vec{E}_{k+1}, ...)$. The difference from the explicit scheme is that for the calculation of the velocity only quantities at the new time step are used.

One example of an implicit particle mover represents the so called *1 scheme* (see [8])

$$\begin{aligned} \frac{\vec{X}_{k+1} - \vec{X}_k}{\Delta t} &= \vec{V}_{k+1/2} \,, \\ \frac{\vec{V}_{k+1/2} - \vec{V}_{k-1/2}}{\Delta t} &= \frac{e}{m}\left(\frac{\vec{E}_{k+1}(x_{k+1}) + \vec{E}_{k-1}}{2} + \right. \\ &\qquad \left. + \frac{\vec{V}_{k+1/2} + \vec{V}_{k-1/2}}{2} \times \vec{B}_k \right) . \end{aligned} \tag{18}$$

It can be shown that for a harmonic oscillator (see (12))

$$V,\ X \sim \frac{1}{(\omega_0\Delta t)^{2/3}} \tag{19}$$

if $\omega_0\Delta t \gg 1$. Hence, the corresponding oscillations are heavily damped and the solver (see (18)) can filter unwanted oscillations. As a result, the condition (16) can be neglected.

More rigorous stability analysis will be done in Sec. 7.

4 Plasma Source and Boundary Effects

4.1 Boundary Effects

From the physics point of view, the boundary conditions for the simulated particles are relatively easy to formulate: Particles can be absorbed at boundaries, or injected from there with any distribution. On the other hand, an accurate numerical implementation of particle boundary conditions can be tricky. The problem is that (i) the velocity and position of particles are shifted in time ($\Delta t/2$) and (ii) the velocity of particles are known at discrete time steps, while a particle can cross the boundary at any moment between these steps.

In unbounded plasma simulations particles are usually reflected at the boundaries, or reinjected from the opposite side (see Fig. 2).

A frequently used reflection model, so called *specular reflection*, is given as

$$X^{refl}_{k+1} = -X_{k+1} \qquad \text{and} \qquad V^{x,refl}_{k+1/2} = -V^{x}_{k+1/2} \,. \tag{20}$$

Here, the boundary is assumed to be located at $x = 0$ (see Fig. 2). The specular reflection represents the simplest reflection model, but due to relatively low accuracy it can cause artificial effects. Let us estimate the accuracy

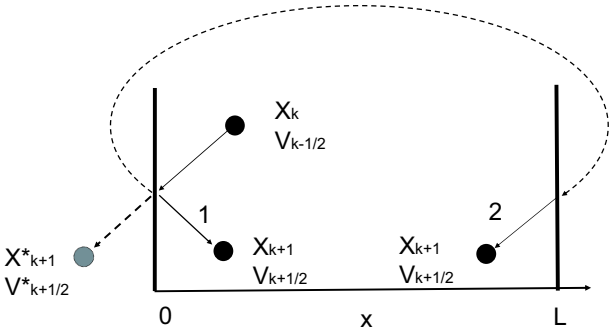

Fig. 2 Particle reflection (1) and reinjection (2) at the boundaries. X^*_{k+1} and V^*_{k+1} denote the virtual position and velocity of a particle if there would be no boundary.

of reflection (see (20)). The exact time when particle reaches a boundary and the corresponding velocity can be written as

$$\begin{aligned} t_0 &= t_k + \left| \frac{X_k}{V^x_{k-1/2}} \right| , \\ V_0 &= V^x_{k-1/2} + \left| \frac{X_k}{V^x_{k-1/2}} \right| \frac{e}{m} E^x_k . \end{aligned} \tag{21}$$

Accordingly, the velocity after the reflection will be

$$\begin{aligned} V^{x,refl}_{k+1/2} &= -V_0 + \left(\Delta t - \left| \frac{X_k}{V^x_{k-1/2}} \right| \right) \frac{e}{m} E^x_k \\ &= -V^x_{k-1/2} + \left(\Delta t - 2 \left| \frac{X_k}{V^x_{k-1/2}} \right| \right) \frac{e}{m} E^x_k . \end{aligned} \tag{22}$$

The second term on the right hand side of (22) represents the error made during the specular reflection, which can cause an artificial particle acceleration and heating.

Particle reinjection is applied usually when the fields satisfy periodic boundary conditions. The reinjection is given by $X^{reij}_{k+1} = L - X_{k+1}$ and $V^{x,reinj}_{k+1/2} = V^x_{k+1/2}$, where $x = L$ denotes the opposite boundary. If the fields are not periodic, then this expression has to be modified. Otherwise a significant numerical error can arise.

The PIC codes simulating bounded plasmas are usually modeling particle absorption and injection at the wall, and some of them are able to tackle complicated plasma-surface interactions too including secondary electron emission, plasma recycling and impurity sputtering (e.g. see [9]- [15]).

Numerically, particle absorption is the most trivial operation and done by removing the particle from memory. Contrary to this, for particle injection complicated numerical models can be required. When a new particle is injected it has to be taken into account that the initial coordinate and velocity are known at the same time, while the leap-frog scheme uses a time shifted values of them. In most cases the number of particles injected per time step is much smaller than the number of particles near the boundary, hence, the PIC code use simple injection models. E.g., an old version of the XPDP1 code (see [16]) has been used $\vec{V}_{k+1/2} = \vec{V} + e/m\Delta t\,(R - 0.5)\,\vec{E}_k$ and $X_{k+1} = R\Delta t V^x_{k+1/2}$, which assumes that particle has been injected at time $t_0 = t_{k+1} - R\Delta t$ with R being an uniformly distributed number between 0 and 1. $\vec{V}$ is the velocity obtained from a given injection distribution function (usually the Maxwellian one). The BIT1 code [17] uses a more simpler injection routine

$$\vec{V}_{k+1/2} = \vec{V}, \quad X_{k+1} = R\Delta t V^x_{k+1/2} , \tag{23}$$

which is independent of the field at the boundary and hence, insensitive to a possible field error there. Description of higher order schemes can be found in [18], [19].

Strictly speaking, the plasma-surface interaction processes can not be attributed to a classical PIC method, but probably all advanced PIC codes simulating bounded plasma contain elements of Monte-Carlo techniques [20]. A general scheme of plasma-surface interactions implemented in PIC codes is given below.

When a primary particle is absorbed at the wall, it can cause the emission of a secondary particle (a special case is reflection of the same particle). In general the emission probability F depends on the surface properties and primary particle energy ϵ and incidence angle α. Accordingly, the PIC code calculates $F(\epsilon, \alpha)$ and compares it to a random number R, uniformly distributed between 0 and 1. If $F > R$ then a secondary particle is injected. The velocity of a secondary particle is calculated according to a prescribed distribution $f_{sec}\left(\vec{V}\right)$. Some codes allow multiple secondary particle injection, including as a special case the thermal emission. The functions F and $f_{\rm sec}$ are determined by surface and solid state physics.

4.2 Particle Loading

The particles in a PIC simulation appear, either by initial loading, or via particle injection from the boundary or from a volumetric source. In any case the corresponding velocities have to be calculated from a given distribution function $f\left(\vec{V}\right)$. Important to note, that there is a significant difference between volumetric particle loading and particle injection from the wall. In the first case the particle velocity is calculated directly from $f\left(\vec{V}\right)$. Contrary to this, the velocity of particles injected from the wall has to be calculated according to $V^x f\left(\vec{V}\right)$, where V^x is the component of the velocity normal to the wall. This becomes clear if we recall that the injection distribution function is the probability that particles having a distribution $f\left(\vec{V}\right)$ will cross the boundary with a given velocity $\vec{V}$. For simplicity we do not distinguish below these two functions denoting them $f\left(\vec{V}\right)$.

There exist two possibilities of calculation of velocities according to a given distribution. The most effective way for a 1D case is to use a cumulative distribution function:

$$F(V) = \frac{\int\limits_{V_{\min}}^{V} f(V')\ \mathrm{d}\vec{V}'}{\int\limits_{V_{\min}}^{V_{\max}} f(V')\ \mathrm{d}V'} \tag{24}$$

with $F(V_{\min}) = 0$ and $F(V_{\max}) = 1$, representing a probability that the velocity of particle is between $V_{\min}$ and V. By equating this function to a sequence of uniformly distributed numbers U (or to random numbers R) between 0 and 1 and inverting it, we produce a sequence of V with the distribution $f(V)$ [4]:

$$F^{-1}(U) = V\ . \tag{25}$$

The same method can be applied to multi-dimensional cases which can be effectively reduced to 1D, e.g., by variable separation: $f\left(\vec{V}\right) = f_1(V^x)\ f_2(V^y)\ f_3(V^z)$. Often inversion of (25) can be done analytically, otherwise it is done numerically.

As an example we consider the injection of Maxwell-distributed particles: $f(V) \sim V\exp(-V^2/(2V_T^2))$. According to (24) and (25) we get

$$F(V) = 1 - \exp\left(-\frac{V^2}{2V_T^2}\right) \quad \text{and} \quad V = V_T\sqrt{-2\ln(1-U)}\ . \tag{26}$$

Another possibility is to use two sets of random numbers R_1 and R_2 (for simplicity we consider 1D case) in a von-Neumann rejection algorithm: $V = V_{\min} + R_1(V_{\max} - V_{\min})$, if $f(V)/(f_{\max}) > R_2$ use V, else try once more. This method requires random number generators of high level and it is time consuming. As a result, it is usually used when the method considered above can not be applied (e.g. for complicated multidimensional $f\left(\vec{V}\right)$).

In advanced codes these distributions are generated and saved at the beginning of a simulation, so that later no further calculations are required except getting $\vec{V}$ from the memory. The same methods are used for spatial distributions $f\left(\vec{X}\right)$, too.

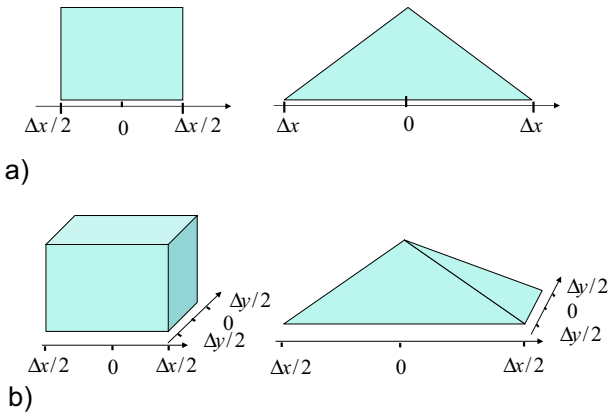

Fig. 3 Particle shapes for the NGP (left) and linear (right) weightings in 1D (a) and 2D (b) cases.

As it was mentioned above, required velocity distributions can be generated by a set of either ordered numbers U or by random numbers R, which are uniformly distributed between 0 and 1. A proper choice of these numbers is not a trivial task and depends on the simulated system; e.g., using random numbers can cause some noise. In addition, numerically generated random numbers in reality represent pseudo-random numbers, which can correlate and cause some unwanted effects. Contrary to this, the distributions generated by a set of ordered numbers, e.g. $U = (i + 0.5)/N$, $i = 1, \ldots, N-1$, are less noisy. On the other hand, in this case the generated distributions represent a multi-beam distribution, which sometimes can cause a beam instability [4].

5 Calculation of Plasma Parameters and Fields Acting on Particles

5.1 Particle Weighting

All numerical schemes considered up to now can be applied not only to PIC, but to any test particle simulation too. In order to simulate a *real* plasma one has to self-consistently obtain the force acting on particles, i.e. to calculate particle and current densities and solve Maxwell's equations. The part of the code calculating macro quantities associated with particles (n, $\vec{J}$, ...) is called *particle weighting*.

For a numerical solution of field equations it is necessary to grid the space: $\vec{x} \to \vec{x}_{\vec{i}}$ with $\vec{i} = 0, \ldots, \vec{N}_g$. Here $\vec{x}$ is a general 3D coordinate and $\vec{N}_g$ number of grid cells (e.g. for 3D Cartesian coordinates $\vec{N}_g = \left(N_g^x,\ N_g^y,\ N_g^z\right)$). Accordingly, the plasma parameters have to be known at these grid points: $A(\vec{x}) \to A_{\vec{i}} = A(\vec{x} = \vec{x}_i)$. The number of simulation particles at grid points is relatively low, so that one can not use an analytic approach of point particles, which is valid only when the number of these particles is very large. The solution is to associate macro parameters to each of the simulation particle. In other words to assume that particles have some shape $S\left(\vec{x} - \vec{X}\right)$, where $\vec{X}$ and $\vec{x}$ denote the particle position and observation point. Accordingly, the *distribution moments* at the grid point $\vec{i}$ associated with the particle "j" can be defined as

$$A_{\vec{i}}^m = a_j^m S\left(\vec{x}_{\vec{i}} - \vec{X}_j\right) \ , \tag{27}$$

where $A_{\vec{i}}^0 = n_{\vec{i}}$, $A_{\vec{i}}^1 = n_{\vec{i}}\vec{V}_{\vec{i}}$, $A_{\vec{i}}^2 = n_{\vec{i}}V_{\vec{i}}^2$ etc. and $a_j^0 = 1/V_g$, $a_j^1 = \vec{V}^j/V_g$, $a_j^2 = (V^j)^2/V_g$ etc. V_g is the volume occupied by the grid cell. The total distribution moments at a given grid point are expressed as

$$A_{\vec{i}}^m = \sum_{j=1}^{N} a_j^m S\left(\vec{x}_{\vec{i}} - \vec{X}_j\right) \ . \tag{28}$$

Stability and simulation speed of PIC simulations strongly depend on the choice of the shape function $S(\vec{x})$. It has to satisfy a number of conditions. The first two conditions correspond to space isotropy,

$$S(x) = S(-x) \ , \tag{29}$$

and charge conservation

$$\sum_{\vec{i}} S\left(\vec{x_{\vec{i}}} - \vec{X}\right) = 1 \; . \tag{30}$$

The rest of the conditions can be obtained requiring an increasing accuracy of the weighting scheme. To derive them consider a potential generated at the point x by a unit charge located at the point X, $G\left(x - X\right)$. In other words $G\left(x - X\right)$ is the Green's function (for simplicity we consider a 1D case). Introducing the weighting scheme we can write the potential generated by some particle located at X as

$$\phi\left(x\right) = e \sum_{i=1}^{m} S\left(x_i - X\right) G\left(x - x_i\right) \; , \tag{31}$$

here e is the particle charge and m the number of nearest grid points with assigned charge. Expanding $G\left(x - x_i\right)$ near $\left(x - X\right)$ we get

$$\begin{aligned} \phi\left(x\right) &= e \sum_{i=1}^{m} S\left(x_i - X\right) G\left(x - X\right) + \\ &\quad + e \sum_{i=1}^{m} S\left(x_i - X\right) \sum_{n=1}^{\infty} \frac{\left(X - x_i\right)^n}{n!} \, \frac{d^n G\left(x - X\right)}{dx^n} \\ &= eG\left(x - X\right) + \delta\phi\left(x\right) \; , \\ \delta\phi\left(x\right) &= e \sum_{n=1}^{\infty} \frac{1}{n!} \, \frac{d^n G\left(x - X\right)}{dx^n} \sum_{i=1}^{m} S\left(x_i - X\right) \left(X - x_i\right)^n \; . \end{aligned} \tag{32}$$

The first term on the right hand side of expression (32) represents a *physical* potential, while $\delta\phi$ is an unphysical part of it introduced by weighting. It is obvious to require this term to be as small as possible. This can be done by requiring

$$\sum_{i=1}^{m} S\left(x_i - X\right) \left(x_i - X\right)^n = 0 \tag{33}$$

with $n = 1, \ldots, n_{\max} - 1$. Substituting the expression (33) into (32) we get

$$\begin{aligned} \delta\phi\left(x\right) &= \sum_{n=n_{\max}}^{\infty} \frac{1}{n!} \, \frac{d^n G\left(x - X\right)}{dx^n} \sum_{i=1}^{m} S\left(x_i - X\right) \left(X - x_i\right)^n \\ &\sim G\left(x - X\right) \sum_{i=1}^{m} S\left(x_i - X\right) \sum_{n=n_{\max}}^{\infty} \frac{\left(X - x_i\right)^n}{n! \left(x - X\right)^n} \; . \end{aligned} \tag{34}$$

Thus, at large distance from the particle ($|X - x_i| < |x - X|$) $\delta\phi\left(x\right)$ decreases with increasing n_{max}.

The shape functions can be directly constructed from the conditions (29), (30) and (33). The later two represent algebraic equations for $S\left(x_i - X\right)$. Hence, the number of conditions (33), which can be satisfied depends on the maximum number of nearest grid points m to which the particle is weighted. The simplest shape function assigns density to the nearest grid point ($m = 1$) and satisfies just the first two conditions (29) and (30). For 1D Cartesian coordinates it is given as $S^0\left(x\right) = 1$, if $|x| < \Delta x/2$, otherwise $S^0\left(x\right) = 0$, where Δx is the size of spatial grid. This weighting is called zero order or NGP weighting and was used in the first PIC codes (see Fig. 3). Although the NGP scheme requires less CPU time, it is relatively noisy and probably not in use any more. The next, first order weighting scheme assigns density to two nearest grid points ($m = 2$) and given as $S^1\left(x\right) = 1 - |x|/(\Delta x)$, if $|x| < \Delta x$, otherwise $S^1\left(x\right) = 0$. Often it is called a cloud in cell (CIC) scheme. It satisfies one more condition in (33), with $n_{\max} = 1$, and is more accurate than the NGP scheme. Probably the CIC represents the most commonly used weighting scheme. Generalization to multidimensional Cartesian

coordinates is trivial: $S(\vec{x}) = S(x)\,S(y)\,S(z)$. The higher order schemes (see [5]) can increase the accuracy of simulation (when other parameters are fixed), but require significantly longer CPU time.

Some authors often use another definition of the particle shape $D(x)$ (e.g. see [5]):

$$S(x_i - x) = \int_{x_i-\Delta x/2}^{x_i+\Delta x/2} D(x' - x)\ \mathrm{d}x' \ . \tag{35}$$

The meaning of this expression is that the density at the grid point x_i assigned by the particle located at the point x represents the average of the particle *real* shape $D(x' - x)$ over the area $[x_i - \Delta x/2; x_i + \Delta x/2]$. For the nearest grid point and linear weightings $D(x) = \delta(x)$ and $D(x) = H(\Delta x/2 - |x|)$, respectively. Here $H(x)$ is the step-function: $H(x) = 1$, if $x > 0$, else $H(x) = 0$.

The described method can be generalized for curvilinear coordinates, e.g. see [4], [6], [21] and references there. Here we just note that this generalization is not straightforward and has to be done with a care, otherwise weighting will not conserve the charge and current densities ([21]- [23]).

5.2 Field Weighting

After the calculation of charge and current densities the code solves the Maxwell's equations (cf. Fig. 1) and delivers fields at the grid points $\vec{i} = 0, \ldots, \vec{N}_g$. These fields can not be used directly for the calculation of force acting on particles, which are located at any point and not necessarily at the grid points. Calculation of fields at any point is done in a similar way as charge assignment and called *field weighting*. So, we have $\vec{E}_{\vec{i}}$ and $\vec{B}_{\vec{i}}$ and want to calculate $\vec{E}(\vec{x})$ and $\vec{B}(\vec{x})$ at any point $\vec{x}$. This interpolation should conserve momentum, which can be done by requiring that the following conditions are satisfied:

(i) Weighting schemes for the field and particles are same:

$$\vec{E}(\vec{x}) = \sum_{\vec{i}} \vec{E}_{\vec{i}} \vec{S}\left(\vec{x}_{\vec{i}} - \vec{x}\right) \ . \tag{36}$$

(ii) The field solver has a correct spatial symmetry, i.e. formally the field can be expressed in the following form (for simplicity we consider the 1D case):

$$E_i = \sum_k g_{ik}\rho_k, \quad g_{ik} = -g_{ki}, \tag{37}$$

where ρ_k is the charge density at the grid point k. In order to understand this condition better, let us consider a 1D electrostatic system. By integrating the Poisson equation we obtain:

$$E(x) = \frac{1}{2\varepsilon_0}\left(\int_a^x \rho\ \mathrm{d}x - \int_x^b \rho\ \mathrm{d}x\right) + E_b + E_a \ , \tag{38}$$

where a and b define boundaries of the system. Assuming that either a and b are sufficiently far and $E_{a,b} = \rho_{a,b} = 0$, or the system (potential) is periodic $E_b = -E_a$, $\rho_b = \rho_a$, we obtain

$$\begin{aligned} E(x_i) &= \frac{1}{2\varepsilon_0}\left(\int_a^{x_i} \rho\ \mathrm{d}x - \int_{x_i}^b \rho\ \mathrm{d}x\right) \\ &= \frac{\Delta x}{4\varepsilon_0}\left(\sum_{k=1}^{i-1}(\rho_k + \rho_{k+1}) - \sum_{k=i}^{N_g-1}(\rho_k + \rho_{k+1})\right) = \\ &= \frac{\Delta x}{4\varepsilon_0}\sum_{k=1}^{N_g} g_{ik}\rho_k \end{aligned} \tag{39}$$

with $N_g \to \infty$, $\Delta x = [b,a]/N_g$ and

$$g_{ik} = \begin{cases} 2 & \text{if } i > k \\ -2 & \text{if } i < k \\ 0 & \text{if } i = k \end{cases} . \tag{40}$$

Thus, the condition (37) is satisfied.

Let us check different conservation constraints.

1. The self-force of the particle located at the point x can be calculated as follows:

$$\begin{aligned} F_{self} &= e\sum_i E_i S\left(x_i - x\right) = e\sum_{i,\,k} g_{ik} S\left(x_i - x\right)\rho_k \\ &= \frac{e^2}{V_g}\sum_{i,\,k} g_{ik} S\left(x_i - x\right) S\left(x_i - x\right) = (i \leftrightarrow k) \\ &= -\frac{e^2}{V_g}\sum_{i,\,k} g_{ik} S\left(x_i - x\right) S\left(x_i - x\right) = -F_{self} = 0 , \end{aligned} \tag{41}$$

2. The two-particle interaction force is given as:

$$\begin{aligned} F_{12} &= e_1 E_2\left(x_1\right) = e_1\sum_i E_{2,i} S\left(x_i - x_1\right) \\ &= \frac{e_1 e_2}{V_g}\sum_{i,\,k} g_{ik} S\left(x_i - x_1\right) S\left(x_k - x_2\right) \\ &= -\frac{e_1 e_2}{V_g}\sum_{i,\,k} g_{ki} S\left(x_i - x_1\right) S\left(x_k - x_2\right) \\ &= -e_2\sum_{i,\,k} g_{ki} S\left(x_k - x_2\right)\rho_{1,k} = -e_2 E_1\left(x_2\right) \\ &= -F_{21} . \end{aligned} \tag{42}$$

Here, E_p denotes the electric field generated by the particle p.

Momentum conservation:

$$\begin{aligned} \frac{d\vec{P}}{dt} &= \vec{F} = \sum_{p=1}^{N} e_p\left(\vec{E}\left(\vec{x}_p\right) + \vec{V}_p \times \vec{B}\left(\vec{x}_p\right)\right) \\ &= \sum_{p=1}^{N} e_p \sum_{\vec{i}} \vec{E}_{\vec{i}} S\left(\vec{x}_i - \vec{x}_p\right) + \sum_{p=1}^{N} e_p \vec{V}_p \times \sum_{\vec{i}} \vec{B}_{\vec{i}} S\left(\vec{x}_i - \vec{x}_p\right) \\ &= \sum_{\vec{i}} \vec{E}_{\vec{i}} \sum_{p=1}^{N} e_p S\left(\vec{x}_i - \vec{x}_p\right) - \sum_{\vec{i}} \vec{B}_{\vec{i}} \times \sum_{p=1}^{N} e_p \vec{V}_p S\left(\vec{x}_i - \vec{x}_p\right) \\ &= V_g \sum_{\vec{i}} \left(\rho_{\vec{i}} \vec{E}_{\vec{i}} + \vec{J}_{\vec{i}} \times \vec{B}_{\vec{i}}\right) . \end{aligned} \tag{43}$$

Representing fields as a sum of external and internal components $\vec{E}_{\vec{i}} = \vec{E}^{ext}_{\vec{i}} + \vec{E}^{int}_{\vec{i}}$ and $\vec{B}_{\vec{i}} = \vec{B}^{ext}_{\vec{i}}$, where $\vec{E}^{int}_{\vec{i}}$ is given in expression (36), after some trivial transformations we finally obtain the equation of momentum conservation

$$\frac{d\vec{P}}{dt} = V_g \sum_{\vec{i}} \left(\rho_{\vec{i}} \vec{E}^{ext}_{\vec{i}} + \vec{J}_{\vec{i}} \times \vec{B}_{\vec{i}}\right) . \tag{44}$$

As we see, the conditions (36) and (37) guarantee that during the force weighting the momentum is conserved and the inter-particle forces are calculated in a proper way. It has to be noted that:

1. We neglected contribution of an internal magnetic field $\vec{B}^{int}$;

2. The momentum conserving schemes considered above does not necessarily conserve the energy too (for energy conserving schemes see [4] and [5]);

3. The condition (37) is not satisfied in general for coordinate systems with nonuniform grids, causing the self-force and incorrect inter-particle forces. E.g., if we introduce a nonuniform grid $\Delta x^i = \Delta x \alpha^i$ with $\alpha^i \neq \alpha^{j \neq i}$, in expression (39) we obtain

$$E\left(x_i\right) = \frac{\Delta x}{4\varepsilon_0} \sum_{k=1}^{N_g} g_{ik} \rho_k \ , \tag{45}$$

with

$$g_{ik} = \begin{cases} \alpha^k + \alpha^{k-1} & \text{if } i > k \\ -\left(\alpha^k + \alpha^{k-1}\right) & \text{if } i < k, N_g \to \infty, \ \Delta x = \frac{[b,a]}{\sum_{i=1}^{N_g} \alpha^i} \\ \alpha^{i-1} - \alpha^i & \text{if } i = k \end{cases} \ ,$$

so that $g_{ki} \neq -g_{ik}$.

6 Solution of Maxwell's Equations

6.1 General Remarks

Numerical solution of Maxwell's equations is a continuously developing independent direction in numerical plasma physics (e.g., see [24]). Field solvers in general can be divided into three groups:

1. Mesh-relaxation methods, when the solution is initially guessed and then systematically adjusted until the solution is obtained with required accuracy;

2. Matrix methods, when Maxwell's equations are reduced to a set of linear finite difference equations and solved by some matrix method, and

3. Methods using the so called fast Fourier transform (FFT) and solving equations in Fourier space.

According to the type of the equations to be solved the field solvers can be explicit or implicit. E.g., the explicit solver of the Poisson equation solves the usual Poisson equation

$$\nabla \left[\varepsilon\left(\vec{x}\right) \nabla \varphi\left(\vec{x}, t\right)\right] = -\rho\left(\vec{x}, t\right) \ , \tag{46}$$

while an implicit one solves the following equation:

$$\nabla \left[\left(1 + \eta\left(\vec{x}\right)\right) \varepsilon\left(\vec{x}\right) \varphi\left(\vec{x}, t\right)\right] = -\rho\left(\vec{x}, t\right) \ . \tag{47}$$

Here $\eta\left(\vec{x}\right)$ is the implicit numerical factor, which arises due to the fact that in its implicit formulation a new position (and hence ρ) of particle is calculated from a new field given at the same moment.

As an example we consider some matrix methods, which are frequently used in different codes. For a general overview of different solvers the interested reader can see [4], [5], [25].

6.2 Electrostatic Case, Solution of Poisson Equation

Let us consider the Poisson equation in a Cartesian coordinate system

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\varphi\left(\vec{r}\right)=-\frac{1}{\varepsilon_0}\rho\left(\vec{r}\right)\ , \tag{48}$$

and formulate the corresponding finite difference equations. For this we use the transformation

$$\frac{\partial^2}{\partial x^2}\varphi\ \Rightarrow\ \frac{a\varphi_{i+1}+b\varphi_i+c\varphi_{i-1}}{\Delta x^2}\ . \tag{49}$$

Other components are treated in a similar way. Our aim is to choose the constants a, b and c, so that the error will be smallest. From the symmetry constraint we can write $a=c$. Then by expanding $\varphi_{i\pm1}$ at $x=x_i$

$$\begin{aligned}\varphi_{i\pm1}&=\varphi_i\pm\Delta x\left(\varphi_i\right)'+\frac{\Delta x^2}{2}\left(\varphi_i\right)''\pm\frac{\Delta x^3}{6}\left(\varphi_i\right)'''+\frac{\Delta x^4}{24}\left(\varphi_i\right)^{(4)}\dots\ ,\\ \left(\varphi_i\right)^{(k)}&=\left(\frac{\partial^k}{\partial x^k}\varphi\right)_{x=x_i}\ ,\end{aligned} \tag{50}$$

and substituting in (49) we obtain

$$a\varphi_{i+1}+b\varphi_i+c\varphi_{i-1}=\varphi_i\left(2a+b\right)+\left(\varphi_i\right)''a\Delta x^2+\left(\varphi_i\right)^{(4)}a\frac{\Delta x^4}{12}+\dots\ . \tag{51}$$

Hence, by choosing $a=1$ and $b=-2a=-2$ we get

$$\left(\frac{\partial^2\varphi}{\partial x^2}\right)_{x=x_i}-\frac{\varphi_{i+1}-2\varphi_i+\varphi_{i-1}}{\Delta x^2}=\frac{\Delta x^2}{12}\left(\varphi_i\right)^{(4)}+\Theta\left(\Delta x^4\right)\ . \tag{52}$$

Hence, the finite difference equation (49) with $b=-2$ and $a=c=1$ has second order accuracy ($\sim\Delta x^2$). Usually this accuracy is sufficient, otherwise one can consider a more accurate scheme

$$\frac{\partial^2}{\partial x^2}\varphi\ \Rightarrow\ \frac{a\varphi_{i+2}+b\varphi_{i+1}+c\varphi_i+d\varphi_{i-1}+e\varphi_{i-2}}{\Delta x^2}\ . \tag{53}$$

The x component of the electric field is calculated according to the following expression

$$E_i=-\frac{\varphi_{i+1}-\varphi_{i-1}}{2\Delta x} \tag{54}$$

Generalisation to the curvilinear coordinates is straightforward (see [4]).

6.2.1 1D Case: Bounded Plasma with External Circuit

An excellent example of an 1D Poisson solver has been introduced in [16]. The solver is applied to an 1D bounded plasma between two electrodes and solves Poisson and external circuit equations simultaneously. Later, this solver has been applied to a 2D plasma model [26]. Below we consider an simplified version of this solver assuming that the external circuit consists of a voltage (or current) source $V(t)$ $(I(t))$ and a capacitor C (see Fig. 4)).

The Poisson equation for a 1D plasma is given as

$$\varphi_{i+1}-2\varphi_i+\varphi_{i-1}=-\frac{\Delta x^2}{\varepsilon_0}\rho_i\ . \tag{55}$$

This equation is second order, so that we need two boundary conditions for the solution. The first one can be a potential at the right-hand-side (rhs) wall:

$$\varphi_{Ng}=0. \tag{56}$$

The second condition can be formulated at the left-hand-side (lhs) wall:

$$\frac{\varphi_0 - \varphi_1}{\Delta x} = E\left(x = \frac{\Delta x}{2}\right) = E_0 + \frac{1}{\varepsilon_0}\int_0^{\Delta x/2} \rho \, \mathrm{d}x \approx E_0 + \frac{\Delta x}{2\varepsilon_0}\rho_0 \, . \tag{57}$$

Recalling that E_0 is the electric field at the lhs wall, we can write $E_0 = \sigma_{lhs}/\varepsilon_0$, where σ_{lhs} is the surface charge density there. Hence, the second boundary condition can be formulated as

$$\varphi_0 - \varphi_1 = \frac{\Delta x}{\varepsilon_0}\left(\sigma_{lhs} + \frac{\Delta x}{2}\rho_0\right) \, . \tag{58}$$

In order to calculate σ_{lhs} we have to employ the circuit equation.

Voltage Driven Source with Finite C In this case charge conservation at the lhs wall can be written as

$$\sigma_{lhs}\left(t\right) = \sigma_{lhs}\left(t - \Delta t\right) + \frac{Q_{pl} + Q_{ci}}{S} \, , \tag{59}$$

where Q_{pl} and Q_{ci} are the charge deposited during Δt time on the lhs wall by the plasma and the external circuit, respectively. S is the area of the wall surface. Q_{pl} can be calculated by counting the charge of the plasma particles absorbed at the lhs wall, and Q_{ci} can be given as $Q_{ci} = Q^c\left(t\right) - Q^c\left(t - \Delta t\right)$, where Q^c is the charge at the capacitor. Q^c can be calculated using the Kirchhoff's law

$$\frac{Q^c}{C} = V\left(t\right) + \varphi_{N_g} - \varphi_0 = V\left(t\right) - \varphi_0 \, . \tag{60}$$

Substituting the expressions (59) and (60) into (58) we obtain

$$\begin{aligned} &\varphi_0\left(1 + \frac{C}{S}\frac{\Delta x}{\varepsilon_0}\right) - \varphi_1 \\ &= \frac{\Delta x}{\varepsilon_0}\left(\frac{Q_{pl} + C\left(V\left(t\right) - V\left(t - \Delta t\right) + \varphi_0\left(t - \Delta t\right)\right)}{S} + \right. \\ &\qquad \left. + \sigma_{lhs}\left(t - \Delta t\right) + \frac{\Delta x}{2}\rho_0\right) \, . \end{aligned} \tag{61}$$

Voltage Driven Source with $C \to \infty$ In this case in spite of (58) we use

$$\varphi_0 - \varphi_{Ng} = \varphi_0 = V\left(t\right) \, . \tag{62}$$

Open Circuit ($C = 0$) In this case we write

$$\sigma_{lhs}\left(t\right) = \sigma_{lhs}\left(t - \Delta t\right) + \frac{Q_{pl}}{S} \, , \tag{63}$$

so that the second boundary condition takes the following form

$$\varphi_0 - \varphi_1 = \frac{\Delta x}{\varepsilon_0}\left(\sigma_{lhs}\left(t - \Delta t\right) + \frac{Q_{pl}}{S} + \frac{\Delta x}{2}\rho_0\right) \, . \tag{64}$$

Current Driven Source In this case Q_{ci} can be directly calculate from the expression $Q_{ci} = \Delta t I(t)$. Then the second boundary condition can be given as

$$\varphi_0 - \varphi_1 = \frac{\Delta x}{\varepsilon_0}\left(\sigma_{lhs}\left(t - \Delta t\right) + \frac{Q_{pl} + \Delta t I\left(t\right)}{S} + \frac{\Delta x}{2}\rho_0\right) \, . \tag{65}$$

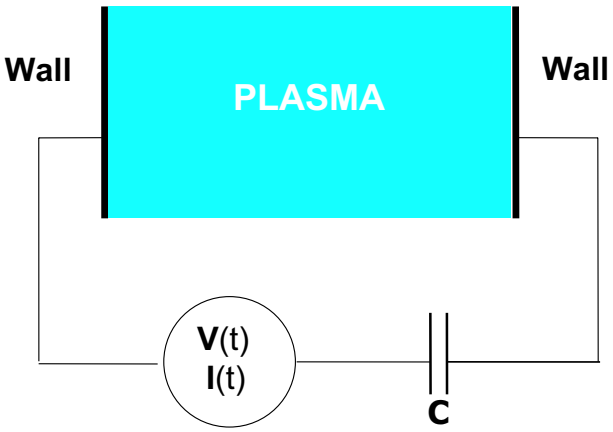

Fig. 4 Scheme of 1D bounded plasma with external circuit.

Combining equations (55), (56) and (61)–(65) we can write the set of difference equations in the following matrix form:

$$\begin{pmatrix} a & b & 0 & & & & & & \\ c & -2 & 1 & 0 & & & & & \\ 0 & 1 & -2 & 1 & 0 & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & & 0 & 1 & -2 & 1 \\ & & & & & & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} \varphi_0 \\ \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_{Ng-2} \\ \varphi_{Ng-1} \end{pmatrix} = -\frac{\Delta x^2}{\varepsilon_0} \begin{pmatrix} d/\Delta x \\ \rho_1 + e \\ \rho_2 \\ \vdots \\ \rho_{Ng-2} \\ \rho_{Ng-1} \end{pmatrix} . \tag{66}$$

Here, for the cases:

1. Voltage driven source or open circuit: $a = -1 - C/S\Delta x/\varepsilon_0$, $b = 1$, $c = 1$, $d = \sigma_{lhs}(t - \Delta t) + (Q_{pl} + C(V(t) - V(t - \Delta t) + \varphi_0(t - \Delta t)))/S + \Delta x/2\rho^0$ and $e = 0$.
2. Short circuit ($C \to \infty$): $a = b = c = d = 0$ and $e = \varepsilon_0/(\Delta x^2)V\,(t)$.
3. Current driven source: $a = -1$, $b = 1$, $c = 1$, $d = \sigma_{lhs}(t - \Delta t) + (Q_{pl} + \Delta t I(t))/S + \Delta x/2\rho^0$ and $e = 0$.

The matrix (66) can be solved by standard inverse matrix solvers (e.g., see [27]).

6.2.2 2D Case: Generalization of the 1D Solver

This 1D solver can be generalized for a 2D case. The main difference between the 1D and 2D cases represent the decomposition of the field and the boundary conditions at internal objects introduced in 2D (for details see [26]).

Field decomposition is given by the following expression:

$$\varphi\,(t, x, y) = \varphi^{pl}\,(t, x, y) + \varphi^{con}\,(t)\,\varphi^{vac}\,(x, y)\ . \tag{67}$$

Here φ^{pl} is the *plasma* field with the zero boundary conditions

$$\Delta\varphi^{pl}\,(t, x, y) = -\frac{1}{\varepsilon_0}\rho\,(x, y)\ , \qquad \varphi^{pl}\big|_b = 0\ , \tag{68}$$

where φ^{vac} is the *vacuum* field with the unit boundary conditions

$$\Delta\varphi^{vac}\,(x, y) = 0\ , \qquad \varphi^{vac}\big|_b = 1\ . \tag{69}$$

$\varphi^{con}\,(t)$ is the field at conductors, which is either calculated self-consistently (for electrodes), or prescribed (e.g., at the wall). The symbol $|_b$ denotes a plasma boundary.

It's easy to see that φ in (67) represents an exact solution of the Poisson equation with the given boundary conditions. The advantage of this decomposition is that (i) the vacuum field has to be calculated just once and (ii)

the Poisson equation (68) with the zero boundary conditions is easier to solve, than one with a general boundary conditions. As a result, the field decomposition can save a lot of CPU time.

The equation of the plasma field (68) is reduced to a set of finite difference equations

$$\frac{\varphi_{i+1,j}-2\varphi_{ij}+\varphi_{i-1,j}}{\Delta x^2}+\frac{\varphi_{i,j+1}-2\varphi_{ij}+\varphi_{i,j-1}}{\Delta y^2}=-\frac{1}{\varepsilon_0}\rho_{ij} \tag{70}$$

with $\varphi|_b=0$, which can be solved by matrix method. In a similar way the Laplace equation (69) can be solved for the vacuum field.

The corresponding boundary conditions at the wall of internal objects are calculated using Gauss' law:

$$\oint \varepsilon\vec{E}\,\mathrm{d}\vec{S}=\int \rho\,\mathrm{d}V+\oint \vec{\sigma}\,\mathrm{d}\vec{S}\,, \tag{71}$$

which in a finite volume representation can be written as

$$\begin{aligned}&\Delta y\Delta z\left(\varepsilon_{i+1/2,j}E_{i+1/2,j}-\varepsilon_{i-1/2,j}E_{i-1/2,j}\right)+\\&\quad+\Delta x\Delta z\left(\varepsilon_{i,j+1/2}E_{i,j+1/2}-\varepsilon_{i,j-1/2}E_{i,j-1/2}\right)\\&=\rho_{ij}\Delta V_{ij}+\sigma_{ij}\Delta S_{ij}\ .\end{aligned} \tag{72}$$

Here ΔV_{ij} and ΔS_{ij} are the volume and area associated with the given grid point i, j. The electric fields entering in this equation are calculated according to the following expressions

$$E_{i\pm1/2,j}=\pm\frac{\varphi_{i,j}-\varphi_{i\pm1,j}}{\Delta x},\quad E_{i,j\pm1/2}=\pm\frac{\varphi_{i,j}-\varphi_{i,j\pm1}}{\Delta x}\ . \tag{73}$$

Calculation of the potential at the plasma boundary $\varphi^{con}(t)$ consists in general of three parts. The potential at the outer wall is fixed and usually chosen as 0. The potential at the electrodes, which are connected to an external circuit is done in a similar way as for the 1D case considered above. For calculation of the potential at the internal object equation (72) is solved. We note that the later task is case dependent and not a trivial one, e.g., the solution depends on the object shape or material (conductor or dielectric). For further details see [26].

6.2.3 2D Case: Cartesian/Fourier Solver

The maximum number of operations to be performed by a matrix solver (per dimension) scales as $\sim N_g^2$ and drastically increases with N_g. This number can be significantly reduced by using a fast Fourier Transform (FFT) solver, which scales as $\sim N_g\ln N_g$ (see [27]). This scaling can be significantly improved by using different optimizations. One example when FFT solvers can be applied is a 2D plasma, which is bounded in one direction and unbounded or periodic in the other one. In this case one can apply a discrete Fourier transform along the periodic direction

$$A_{ij}=\frac{1}{2\pi}\sum_{k=0}^{Ny-1}A_i^k e^{(-i2\pi j/N_y k)} \tag{74}$$

with $A=\varphi,\rho$. By substituting this expression into (70) we obtain

$$\varphi_{i+1}^k-2\left(1+2\left(\frac{\Delta x}{\Delta y}\sin\left(\frac{\pi k}{N_y}\right)\right)^2\right)\varphi_i^k+\varphi_{i-1}^k=-\frac{\Delta x^2}{\varepsilon_0}\rho_i^k\ . \tag{75}$$

It is easy to see that (75) is similar to the one for the 1D model considered above and can be solved in the same way. The main difference are the boundary conditions along the x-axis. E.g., if the plasma is bounded between two conducting walls, then $\varphi_0^k=\varphi_{N_g}^k=0$ if $k>0$, and for the $k=0$ - component we have exactly the same equation as for 1D with the same boundary condition.

6.3 Electromagnetic Case

For sufficiently strong fields and/or very fast processes it is necessary to solve the complete set of Maxwell's equations (3). It is obvious that corresponding solvers are more complicated than ones considered above. Correspondingly a detailed description of them is out of the scope of this work. Here we present just one of possible schemes, which is implemented in the XOOPIC code [6].

In order to ensure high speed and accuracy it is convenient to introduce a leap-frog scheme also for the fields. The leap-frog scheme is applied to the space coordinates too, which means that electric and magnetic fields are shifted in time by $\Delta t/2$, and different components of them are shifted in space by $\Delta x/2$. In other words

1. $\vec{E}$ is defined at $t = n\Delta t$ and $\vec{B}$ and $\vec{J}$ at $t = (n + 1/2)\Delta t$;
2. "i" components of the electric field and current density are defined at the points $x_i + \Delta_i/2$, x_k and x_j, and same component of the magnetic field at x_i, $x_k + \Delta_k/2$ and x_j. Here, x_s and Δ_s for $s = i, k, j$ denote the grid point and grid size along the s-axis. i, k and j denote the indices of the right-handed Cartesian coordinate system.

As a result the finite-differenced Ampere's and Faraday's laws in Cartesian coordinates can be written as follows:

$$\begin{aligned}&\frac{D^{i,t}_{i+1/2,k,j} - D^{i,t-\Delta t}_{i+1/2,k,j}}{\Delta t}\\ &\quad= \frac{H^{j,t-\Delta t/2}_{i+1/2,k+1/2,j} - H^{j,t-\Delta t/2}_{i+1/2,k-1/2,j}}{\Delta x_k} +\\ &\quad\quad- \frac{H^{k,t-\Delta t/2}_{i+1/2,k,j+1/2} - H^{k,t-\Delta t/2}_{i+1/2,k,j-1/2}}{\Delta x_j} - J^{i,t-\Delta t/2}_{i+1/2,k,j}\ ,\end{aligned} \tag{76}$$

$$\begin{aligned}&\frac{B^{i,t+\Delta t/2}_{i,k+1/2,j+1/2} - B^{i,t-\Delta t/2}_{i,k+1/2,j+1/2}}{\Delta t}\\ &\quad= \frac{E^{k,t}_{i,k+1/2,j+1} - E^{k,t}_{i,k+1/2,j}}{\Delta x_j} - \frac{E^{j,t}_{i,k+1,j+1/2} - E^{j,t}_{i,k,j+1/2}}{\Delta x_k}.\end{aligned} \tag{77}$$

The solver works in the following way. The equations $\nabla\vec{D} = \rho$ and $\nabla\vec{B} = 0$ prescribe the initial electromagnetic fields. They remain satisfied due to Ampere's and Faraday's law, which are solved from the finite difference equations (76) and (77). The corresponding boundary conditions strongly depend on the simulated plasma model. E.g., at the wall representing an ideal conductor $E_\parallel = B_\perp = 0$, where $\parallel$ and $\perp$ denote the components parallel and normal to the wall, respectively.

As one can see, the components of the electromagnetic field obtained from (76) and (77) are defined at different time steps and spatial points than for the particle mover. Moreover, the current density obtained from the particle position is not defined at $t = (n + 1/2)\,\Delta t$ as it is required for Ampere's law(76). Hence, it is necessary to additionally couple the particle and field solvers, so that the plasma and field parameters are obtained at the required time steps and space points. An explicit method is used to calculate directly time averages:

$$\overrightarrow{B}^t = \frac{\overrightarrow{B}^{t+\Delta t/2} + \overrightarrow{B}^{t-\Delta t/2}}{2}, \quad \overrightarrow{J}^{t+\Delta t/2} = \sum_{n=1}^{N} \overrightarrow{V}^{t+\Delta t/2}_n \frac{\overrightarrow{r}^{\,t}_n + \overrightarrow{r}^{\,t-\Delta t}_n}{2}. \tag{78}$$

Other methods for coupling of particle and field solvers are considered in [4].

It is useful to derive a general stability criteria for the electromagnetic case. For this we consider electromagnetic waves in vacuum:

$$\vec{A} = \vec{A}_0 exp\left(i(\vec{k}\vec{x} - \omega t)\right)\ , \tag{79}$$

with $\vec{A} = \vec{E}, \vec{B}$. After substitution of (79) into field equations (76) and (77) and trivial transformations we obtain

$$\left(\frac{\sin(\omega t/2)}{c\Delta t}\right)^2 = \sum_{i=1}^{3}\left(\frac{\sin(k_i x_i/2)}{\Delta x_i}\right)^2 , \tag{80}$$

where $c = \sqrt{1/\varepsilon_0\mu_0}$is the speed of light. It is obvious that the solution is stable (i.e. Im$\omega < 0$) if

$$(c\Delta t)^2 < \left(\sum_{i=1}^{3}\frac{1}{\Delta x_i^2}\right)^{-1} . \tag{81}$$

Often, this so called *Courant* condition requires unnecessary small time step for the particle mover. In order to relax it one can introduce separate time steps for field and particles. This procedure is called *sub-cycling* (see [4]):

1. A cycle starts with particle velocities $\overrightarrow{V}$ and current density $\overrightarrow{J}$, particle coordinates $\overrightarrow{r}$, charge density ρ and the electric field $\overrightarrow{E}$ at t_n, and the magnetic field $\overrightarrow{B}$ at $t_{n-1/4}$;
2. Particle velocities and coordinates are advanced, $\overrightarrow{V}_{n-1/2} \rightarrow \overrightarrow{V}_{n+1/2}$, $\overrightarrow{r}_n \rightarrow \overrightarrow{r}$, and new charge and current densities are found ρ_{n+1}, $\overrightarrow{J}_{n+1/2}$;
3. The magnetic field is advanced using the Faraday's law (77) with $\Delta t \rightarrow \Delta t/2$: $\overrightarrow{B}_{n-1/4} \rightarrow \overrightarrow{B}_{n+1/4}$;
4. The electric field is advanced $\overrightarrow{E}_n \rightarrow \overrightarrow{E}_{n+1/2}$ using (76) with $\Delta t \rightarrow \Delta t/2$, $\overrightarrow{B}_{n+1/4}$ and $\overrightarrow{J}_{n+1/4} = 0.5\left(\overrightarrow{J}_n + \overrightarrow{J}_{n+1/2}\right)$;
5. The above two steps are repeated until we get $\overrightarrow{B}_{n+3/4}$ and $\overrightarrow{E}_{n+1}$.

As we see this *sub-cycling* allows using two times smaller time step for the fields than one for particles (see Fig. 5).

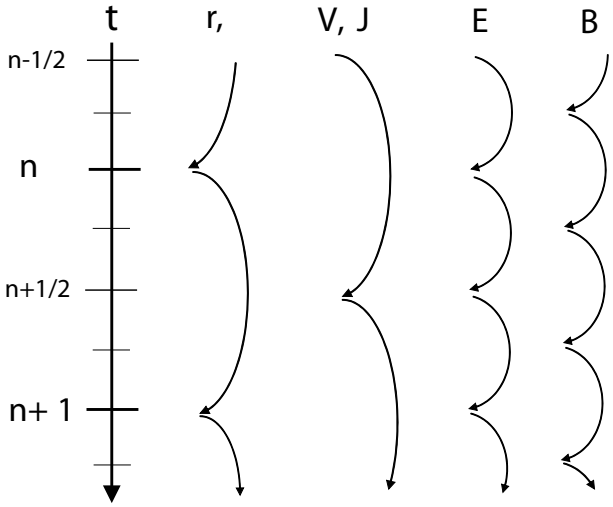

Fig. 5 Diagram of particle and field subcycling.

The routines described above namely: The field solver, the particle mover with proper boundary conditions and the particle source, weighting of particles and fields represent a complete PIC code in its classical understanding. Starting from 1970s a number of PIC codes include different models of particle collisions. Today the majority of PIC codes include at least some kind of collision operator, which have to be attributed to a PIC technique. These operators are usually based on statistical methods and correspondingly are called Monte Carlo (MC) models. Often different authors use the name PIC-MC, or P^3M codes (Particle-Particle for collisions and Particle-Mesh for usual PIC). The MC simulations represent an independent branch of numerical physics and the interested

reader can find more on MC method in corresponding literature. Below we consider the main features of the collision models used in PIC codes.

Before introducing collision models it is useful to consider analytic linear theory of plasma simulated by PIC. This will allow us to better understand the accuracy and corresponding restrictions for PIC simulations.

7 Linear theory of unmagnetized PIC simulated plasma

It is important to estimate analytically the deviation of PIC results from the correct solution. The best candidate for this check is the linear theory of plasma oscillations. In the given paragraph we consider linear plasma oscillations of 1D unmagnetized plasma.

Contrary to a real plasma the PIC simulated plasma has the following nature: (i) the simulated particles have finite size, (ii) the number of these particles is significantly lower as in real plasma and (iii) time and space are gridded. In this paragraph we consider effects, which appiar in PIC simulations due to these approximations. This will help us to understand the main differences between the PIC simulated and continues plasma models. For simplicity we assume that time is continues.

We consider high frequency plasma oscillations with $\omega \gtrsim \omega_p = \sqrt{ne^2/m_e\varepsilon_0}$ (e.g. see [28]), so that ions can assumed to be immobile. 1D kinetic equation for electrons is given as:

$$\left(\frac{\partial}{\partial t} + V\frac{\partial}{\partial x} + \frac{F}{m_e}\frac{\partial}{\partial V}\right) f\left(x, V, t\right) = 0. \tag{82}$$

The difference with the continues case is that the force in not local any more (see (36))

$$F = e\sum_j E_j S\left(x_j - x\right), \tag{83}$$

and the charge density is calculated according to

$$\rho\left(x\right) = \frac{e}{\Delta x}\int_{-\infty}^{+\infty} n\left(x'\right) S\left(x - x'\right) \mathrm{d}x', \tag{84}$$

where $n\left(x\right) = \int f \mathrm{d}V$ is the distribution of simulated particle centers.

After standard linearization of (82) for small amplitude waves $\sim \exp\left(-i\omega t\right)$ we write

$$\left(-i\omega + V\frac{\partial}{\partial x}\right) f_1\left(x, V\right) = -\frac{F}{m_e}\frac{\partial}{\partial V} f_0\left(V\right), \tag{85}$$

where

$$f\left(x, V, t\right) \approx n_0 f_0\left(V\right) + f_1\left(x, V\right)\exp\left(-i\omega t\right), \quad \frac{\left|f_1\right|}{n_0\left|f_0\right|} \ll 1. \tag{86}$$

In order to derive dispersion relation it is necessary to employ Poisson's equation (55) and transfer equations in a Fourier space. Here, contrary to a continues case it is necessary to introduce two kinds of transforms. The variable x in (85) corresponds to a continues trajectory of a simulation particle (for $\Delta t \to 0$), so that the corresponding Fourier and inverse Fourier transformations are given as

$$A\left(x\right) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \widehat{A}\left(k\right)\exp\left(ikx\right)\mathrm{d}k, \quad \widehat{A}\left(k\right) = \int_{-\infty}^{+\infty} A\left(x\right)\exp\left(-ikx\right)\mathrm{d}x, \tag{87}$$

where $A\left(x\right)$ can be any physical quantity in a continues space x. Whereas the Eq. (55) operates in a discretized space x_j. The corresponding k space becomes periodic with the period

$$k_p = \frac{2\pi}{\Delta x}. \tag{88}$$

Hence, the corresponding Fourier transform becomes

$$B_j = \frac{1}{2\pi}\int_{-\pi/\Delta x}^{\pi/\Delta x} B'(k)\exp(ikx)\,\mathrm{d}k, \quad B'(k) = \Delta x \sum_{j=-\infty}^{+\infty} B_j \exp(-ikx_j). \tag{89}$$

Applying the transformation (87) to (85) we obtain

$$\widehat{n}_1(k) = \int \widehat{f}_1(k,V)\,\mathrm{d}V = -\frac{in_0}{m_e}\widehat{F}(k)\int \frac{\frac{\partial f_0(V)}{\partial V}}{\omega - kV + i\epsilon}\mathrm{d}V, \tag{90}$$

where $\epsilon \ll 1$ has been introduced in order to properly avoid the singularity at $\omega = kV$ [29]. $\widehat{F}(k)$ and $\widehat{\rho}(k)$ (which we will need later) can be calculated according to (83) and (84):

$$\begin{aligned}
\widehat{F}(k) &= e\int_{-\infty}^{+\infty}\sum_j E_j S(x_j - x)\exp(-ikx)\,\mathrm{d}x \\
&= e\sum_j E_j \exp(-ikx_j)\int_{-\infty}^{+\infty} S(x_j - x)\exp(-ik(x - x_j))\,\mathrm{d}x \\
&= \frac{e}{\Delta x}E'(k)\,\widehat{S}(-k), \\
\widehat{\rho}(k) &= \frac{e}{\Delta x}\widehat{n}(k)\,\widehat{S}(k).
\end{aligned} \tag{91}$$

Applying the transformation (89) to (55) we get

$$\begin{aligned}
i\xi(k)\,E'(k) &= \frac{1}{\varepsilon_0}\rho'(k), \\
\xi(k) &= \frac{4}{\Delta x}\frac{\sin^2\left(\frac{k\Delta x}{2}\right)}{\sin(k\Delta x)}.
\end{aligned} \tag{92}$$

In order to map discrete and continues Fourier transforms we consider transformation of ρ:

$$\begin{aligned}
\rho(x_j) &= \frac{1}{2\pi}\int_{-\infty}^{+\infty}\widehat{\rho}(k)\exp(ikx_j)\,\mathrm{d}k \\
&= \frac{1}{2\pi}\left[\int_{-\pi/\Delta x}^{\pi/\Delta x} + \int_{-3\pi/\Delta x}^{-\pi/\Delta x} + \int_{\pi/\Delta x}^{3\pi/\Delta x} + \ldots\right]\widehat{\rho}(k)\exp(ikx_j)\,\mathrm{d}k \\
&= \frac{1}{2\pi}\int_{-\pi/\Delta x}^{\pi/\Delta x}\sum_{n=-\infty}^{\infty}\widehat{\rho}(k - nk_p)\exp(ikx_j)\,\mathrm{d}k.
\end{aligned} \tag{93}$$

On the other hand

$$\rho_j = \frac{1}{2\pi}\int_{-\pi/\Delta x}^{\pi/\Delta x}\rho'(k)\exp(ikx_j)\,\mathrm{d}k \equiv \rho(x_j), \tag{94}$$

so that

$$\rho'(k) = \sum_{n=-\infty}^{\infty}\widehat{\rho}(k - nk_p) = \frac{e}{\Delta x}\sum_{n=-\infty}^{\infty}\widehat{n}(k - nk_p)\,\widehat{S}(k - nk_p). \tag{95}$$

Finally, from the expressions (90 - 92) and (95) after some transformations we obtain the dispersion relation

$$1+\frac{\omega_p^2}{\xi\left(k\right)\Delta x^2}\sum_{n=-\infty}^{\infty}\left|\widehat{S}\left(k-nk_p\right)\right|^2\int\frac{\frac{\partial f_0(V)}{\partial V}}{\omega-\left(k-nk_p\right)V+i\epsilon}\mathrm{d}V=0. \tag{96}$$

Or, assuming the unperturbed distribution $f_0\left(V\right)$ to be Maxwellian,

$$1=\frac{\omega_p^2}{2V_T^2\xi\left(k\right)\Delta x^2}\sum_{n=-\infty}^{\infty}\frac{\left|\widehat{S}\left(k-nk_p\right)\right|^2}{k-nk_p}Z'\left(\frac{\omega}{\sqrt{2}\left|k-nk_p\right|V_T}\right), \tag{97}$$

where Z' is the derivative of the dispersion function [30]. In the limit $\Delta x\rightarrow 0$ we have $\widehat{S}\left(k\right)\rightarrow\Delta x$, $\xi\left(k\right)\rightarrow k$ and $k_p\rightarrow\infty$, so that (97) reduces to the well known plasma dispersion relation (e.g. see [28])

$$1=\frac{\omega_p^2}{2k^2V_T^2}Z'\left(\frac{\omega}{\sqrt{2}kV_T}\right)\Rightarrow Re\omega^2\approx\omega_p^2+1.5k^2V_T^2, \tag{98}$$

with a Landau damping term

$$Im\omega\approx-\sqrt{\pi/8}\frac{\omega_p}{k^3\lambda_D^3}\exp\left(-\frac{1}{2k^2\lambda_D^2}-\frac{3}{2}\right). \tag{99}$$

In the Fig. 6 dispersion relations are plotted for CIC scheme (with $\widehat{S}\left(k\right)=\Delta x\left(2\sin\left(k\Delta x/2\right)/k\Delta x\right)^2$) for different $\Delta x/\lambda_D$. As we see, contrary to the continuous case the dispersion relation (97) has solutions with growing modes. These unphysical modes grow due to the resonance of particles having an average velocity V_T with the phase velocity of a corresponding mode,

$$V_{ph}^n=\frac{\omega}{\left|k-nk_p\right|}\approx\frac{\omega_p}{\left|k-nk_p\right|}=\frac{V_T}{\left|k-nk_p\right|\lambda_D}. \tag{100}$$

If $\Delta x\leq\lambda_D$ then $k_p\lambda_D\gg 1$ and $V_{ph}^{n>1}\ll V_T$ and the interaction is weak (the term $\partial f_0\left(V\right)/\partial V$ in (96) is small). As a result, the principle mode ($n=0$) is the dominant one and is Landau-damped. If $\Delta x>\lambda_D$, then $k_p\lambda_D<2\pi$ and for sufficiently small k ($k\lambda_D\ll 1$) the principle mode is heavily damped and its contribution is negligible. On the other hand, for some n the there exists a resonance $V_{ph}^n\sim V_T$ resulting in a strong particle-wave interaction and growth of the corresponding mode.

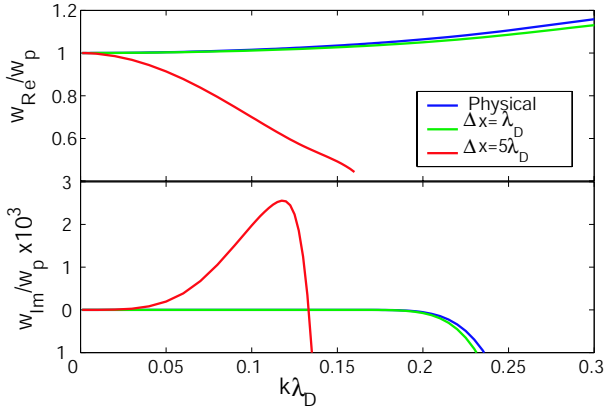

Fig. 6 Dispersion relation (97) for CIC scheme and different $\Delta x/\lambda_D$. For comparison the corresponding physical mode (98, 99) is also plotted. (Online colour: www.cpp-journal.org).

One can conclude that for proper simulations it is necessary to require $\Delta x\leq\lambda_D$. More rigorous analysis result in the following condition [5]:

$$\Delta x<3.4\lambda_D. \tag{101}$$

In the above presented analysis it has been assumed that the number of particles inside the Debye radius, N_D is large. Typically in the PIC simulations $N_D \leq 10^3$, whereas in a real plasma $N_D > 10^6$. From the statistical theory it is known that oscillation amplitude scales as$\sqrt{N}$ (see [31]). Hence, the oscillation amplitude in PIC simulated plasma can be much higher than in the real plasma. Moreover, this can cause unphysical oscillations with the growth rates higher than ones predicted by theory (even if the condition (101) is satisfied). In Fig. 7 the normalized growth rate of the unphysical mode is plotted as a function of N_D for the CIC plasma with $\Delta x = 10\lambda_D$. As one can see the growth rate for $N_D \leq 10^4$ is significantly higher than the analytical prediction from (97).

Often in order to compensate effects of small number of simulated particles the corresponding densities are smoothed. For smoothing different filtering operators are used [4]:

$$A'_i = \frac{WA_{j-1} + A_i + WA_{j+1}}{1+2W}, \quad W = const. \tag{102}$$

Probably the most common one is so called 1-2-1 with $W = 0.5$. In a Fourier space for 1-2-1 filter we get

$$A'(k) = \sin^2\left(\frac{k\Delta x}{2}\right) A(k), \quad |k|\,\Delta x \leq \pi, \tag{103}$$

directly indicating filtering effects for high $|k| \sim \pi/\Delta x$. In order to reduce significantly the noise these filters are applied multiple times. In Fig. 7 we plot results for the simulation when the 1-2-1 filter has been applied 5 times (per time step), indicating significant reduction of the mode growth rate.

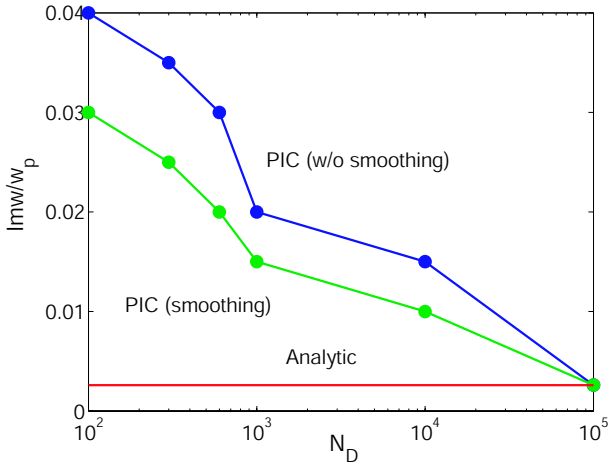

Fig. 7 Normalized growth rate of the plasma oscillations in 1D unmagnetized "CIC plasma" versus N_D. The case $\Delta x = 10\lambda_D$. The analytic curve is obtained from (97).

For completeness we note that in electromagnetic codes sometimes temporal filtering is also used [32].

8 Particle Collisions

8.1 Coulomb Collisions

The forces acting on the particles in a classical PIC scheme correspond to macro fields, so that the simulated plasma is assumed to be collisionless. In order to simulate a collisional plasma it is necessary to implement corresponding routines. Moreover, the field solver is organized in such a way that self-forces are excluded, hence, the field generated by a particle inside the grid cell decreases with decreasing distance from this particle. As a result, inter-particle forces inside grid cells are underestimated (see Fig. 8). Hence, they can be (at least partially) compensated by introducing the Coulomb collision operator.

The first codes simulating Coulomb collisions were the particle–particle codes simulating the exact interaction between each particle pair. Of course this method, which scales as N^2 can not be used in modern PIC simulations. Later different MC models have been developed.

The simplest linear model assumes that the particle distribution is near to a Maxwellian and calculates an average force acting on particles due to collisions [33]. Although this is the fastest operator it probably can

not be used for most of kinetic plasma simulations, when particle distributions are far from the Maxwellian. A nonlinear analogue of this model has been introduced in [34]. Here, the exact collision force is obtained from the particle velocity distribution function. Unfortunately, the number of particles required for building up a sufficiently accurate velocity distribution is extremely large (see [35]), which makes it practically impossible to simulate large systems.

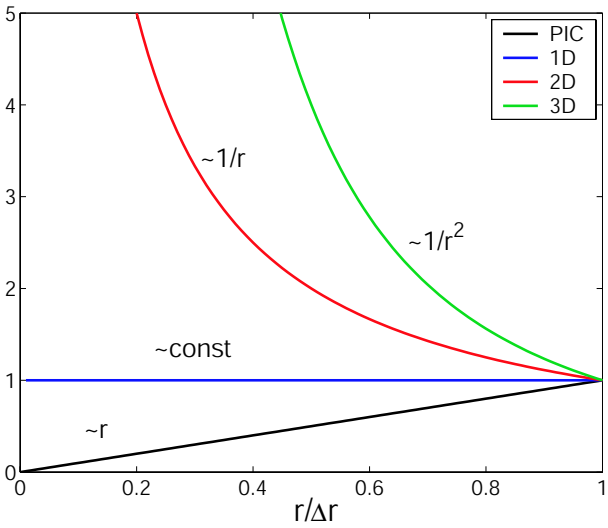

Fig. 8 Inter-particle forces inside grid cell. (Online colour: www.cpp-journal.org).

Most of nonlinear Coulomb collision operators used in our day PIC codes are based on the *binary collision* model introduced in [36]. In this model each particle inside a cell is collided with one particle from the same cell. This collision operator conserves energy and momentum and it is sufficiently accurate. The main idea is based on the fact that there is no need to consider Coulomb interaction between two particles separated by a distance larger than the Debye radius λ_D (e.g., see [37]). Since a typical size of the PIC cell is of the order of λ_D, the interaction between the particles in different cells can be neglected. This method consists of the following three steps (see Fig. 8):

1. First, all particles are grouped according to the cells where they are located.
2. Then these particles are paired in a random way, so that one particle has only one partner and
3. the paired particles are (statistically) collided.

The later is not trivial and we consider it in some detail.

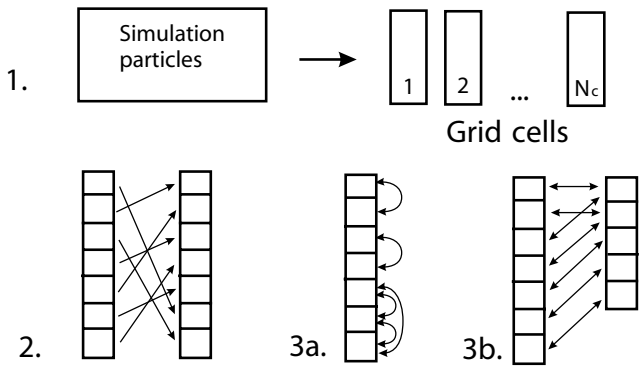

Fig. 9 Binary collision model from [36]. 1. Grouping of particles in the cells; 2. Randomly changing the particle order inside the cells; 3a. Colliding particles of the same type; 3b. Colliding particles of different types.

According to the momentum and energy conservation constrains, we can express the after-collision velocities of particles $\vec{V}_1'$, $\vec{V}_2'$via their before-collision values $\vec{V}_1$and $\vec{V}_2$ (see e.g. [38]):

$$\vec{V}_1' = \vec{V}_1 + \frac{m_2}{m_1+m_2}\Delta\vec{V} \qquad \text{and} \qquad \vec{V}_2' = \vec{V}_2 - \frac{m_1}{m_1+m_2}\Delta\vec{V} \tag{104}$$

with $\Delta\vec{V} = \vec{V}' - \vec{V}$, $\vec{V} = \vec{V}_2 - \vec{V}_1$, $\vec{V}' = \vec{V}_2' - \vec{V}_1'$and $V'^2 = V^2$. As we see the calculation can be reduced to the *scattering* of the relative velocity $\vec{V}$

$$\Delta\vec{V} = \left(\widehat{O}\left(\chi,\psi\right) - 1\right)\vec{V}\ , \tag{105}$$

where $\widehat{O}\left(\chi,\psi\right)$ is the matrix corresponding to the rotation on scattering, χ, and azimuthal, ψ, angles (see [36]):

$$\widehat{O}\left(\chi,\psi\right) = \begin{pmatrix} \cos\left(\chi\right) & a\sin\left(\chi\right)\sin\left(\psi\right) & b_x\sin\left(\chi\right)\cos\left(\psi\right) \\ -a\sin\left(\chi\right)\sin\left(\psi\right) & \cos\left(\chi\right) & b_y\sin\left(\chi\right)\cos\left(\psi\right) \\ -b_x\sin\left(\chi\right)\cos\left(\psi\right) & -b_y\sin\left(\chi\right)\cos\left(\psi\right) & \cos\left(\chi\right) \end{pmatrix} \tag{106}$$
$$b_{x,y} = \frac{V_{x,y}}{V_\perp}, \quad a = \frac{V}{V_\perp}, \quad V_\perp = \sqrt{V_x^2 + V_y^2} > 0$$

The scattering angle χ is calculated from a corresponding statistical distribution. By using the Fokker–Plank collision operator one can show (see [39]) that during the time Δt_c the scattering angle has the following Gaussian distribution:

$$P\left(\chi\right) = \frac{\chi}{\left\langle\chi^2\right\rangle_{\Delta t_c}}\exp\left(-\frac{\chi^2}{2\left\langle\chi^2\right\rangle_{\Delta t_c}}\right) \quad \text{and} \quad \left\langle\chi^2\right\rangle_{\Delta t_c} \equiv \frac{e_1^2e_2^2}{2\pi\varepsilon_0^2}\,\frac{n\Delta t_c\Lambda}{\mu^2V^3}\ . \tag{107}$$

Here $e_{1,2}$and $\mu = m_1m_2/\left(m_{1+}m_2\right)$ denote the charge and reduced mass of the collided particles, respectively. n and Λ are the density and the Landau logarithm [37], respectively. The distribution (107) can be inverted to get

$$\chi = \sqrt{-2\left\langle\chi^2\right\rangle_t\ln R_1}\ . \tag{108}$$

Correspondingly, the azimuthal angle ψ is chosen randomly between 0 and 2π:

$$\psi = 2\pi R_2\ . \tag{109}$$

$R_1$and $R_2$are random numbers between 0 and 1.

Finally, the routine for two-particle collision is reduced to the calculation of expressions (104), (105), (108) and (109). As a result each particle in a grid cell is collided just with one another particle from the same cell saving a lot of CPU time. Interesting to note that the operator introduced in [40] collides all particles inside the cells producing just slightly more accurate relaxation times.

The Coulomb interaction is a long range interaction, when a cumulative effect of many *small scattering* collisions represents the main contribution to the collisionality. Accordingly, the time step for the Coulomb collisions Δt_c should be sufficiently small: $\left\langle\chi^2\right\rangle_{\Delta t}\left(V = V_T\right) \ll 1$. It is more convenient to formulate this condition in the equivalent following form:

$$\nu_c\Delta t_c \ll 1 \qquad \text{and} \qquad \nu_c = \frac{e_1^2e_2^2}{2\pi\varepsilon_0^2}\,\frac{n\Lambda}{\mu^2V_T^3}\ , \tag{110}$$

where ν_c is the characteristic relaxation time for the given Coulomb collisions [41] and V_T is the thermal velocity of the fastest collided particle species. Although usually $\Delta t_c \gg \Delta t$, the binary collision operator is the most time consuming part of the PIC code. Recently, in order to speed up the collisional plasma simulations a number of updated versions of this operator have been developed.

In [7] a new optimization method has been introduced, when all particles are intrinsically sorted according the cell where they are (see Sec. 10). Hence, the first step of the binary collision operator, i.e. grouping of particles in the cells, can be omitted. This gives possibility to easily generalize collision operator to multiple ion species and effectively reduce the number of required random numbers. This saves up to 50% of the CPU time.

If plasma is nearly uniform one can introduce a *cumulative binary collision* operator (e.g., see [42] and [43]). In this model in spite of making a large number of light collisions, particles suffer a single large angle collision. This allows to avoid the condition (110) and significantly reduce the corresponding CPU time.

8.2 Charged-Neutral Particle Collisions

Under realistic conditions the plasma contains different neutral particles, which can collide with the plasma particles. The corresponding collision models used in PIC codes can be divided in two different schemes: *Direct Monte-Carlo* and *Null collision models*. The difference between these models is the way how are the collided particles chosen.

The direct Monte-Carlo model is a common MC scheme when all particles carry information about their collision probability. It is a well studied technique (e.g. see [44], [45]) and frequently used in PIC simulations of nonlinear collision processes in low temperature plasma ([13], [14], [46]- [50]).

The classical direct MC simulation method is based on a simple expression for the collision probability

$$P\left(t\right) = 1 - \exp(-\nu t), \tag{111}$$

where P is the probability that particle will collide at time t, $\nu = \sum_{i=1}^{I} \nu_i$ is the sum of all possible collision frequencies. ν_i depends on local plasma parameters and energy of the given particle: $\nu_i = \sigma_i\left(V\right) n_i$, where V, σ_i and n and are the relative velocity of collided particles, collision cross-section and the density of target particles. For simplicity we consider a linear model with cold target particles, generalization to nonlinear models can be found e.g. in [45]. According to (111) the average time between collisions can be given as

$$t_c = -\frac{\ln R}{\nu}, \tag{112}$$

with R to be a uniform random number between 0 and 1. During the simulation t_c is calculated for each particle and the corresponding counter $\delta t = 0$ is activated. Particles follow a collisionless trajectories until $\delta t = t_c$ when a collision takes place. During the collision it is decided which kind of collision it should take place. For this a random number R_1 (between 0 and 1) is compared to the corresponding relative collision probabilities: if

$$R_1 \leq \frac{P_1\left(t_c\right)}{P\left(t_c\right)} = \frac{1 - exp\left(-\nu_1 t_c\right)}{P\left(t_c\right)} \approx \frac{\nu_1}{\nu}\ , \tag{113}$$

a type 1 collision takes place; else if

$$R_1 \leq \frac{P_1\left(t_c\right) + P_2\left(t_c\right)}{P\left(t_c\right)} \approx \frac{\nu_1 + \nu_2}{\nu}\ , \tag{114}$$

a type 2 collision takes place, and so on until

$$R_1 \leq \frac{\sum_{j=1}^{S} \nu_j}{\nu}, \quad S \leq I, \tag{115}$$

and the collision of a type "S" takes place. After the collision a new t_c is calculated and the counter is initialized ($\delta t = 0$).

It appears that this (classical) method can not be directly applied for plasma applications: during the collisionless motion charged particles can accelerate, so that the actual collision frequency can strongly deviate from one used for calculation of t_c. An updated model has been proposed in [51], which is used in present day PIC-Direct MC codes. In this model in spite of actual collision frequency in (112) a maximum possible one is used:

$$t_c^{\min} = -\frac{\ln R}{\nu_{\max}}. \tag{116}$$

Correspondingly in (113 - 115) ν is substituted by $\nu_{\max}$. As a results, particles are analyzed at minimum possible time intervals increasing accuracy of collision operator. The difference with the expression (112) is that effectively a "null collision" frequency, ν_0, is introduced $\nu_{\max} = \nu + \nu_0$ and if

$$R_1 > \frac{\nu}{\nu_{\max}}, \tag{117}$$

no collision at all takes place.

$$P_{\max}(t) = (1 - \exp(-\nu t))_{\max} = 1 - \exp(-\nu_{\max}\Delta t). \tag{118}$$

The direct MC requires that all particles have to be analyzed for a collision probability. As a result it requires some additional memory storage and sufficiently large amount of the CPU time.

The null collision method (see [52] and [53]) requires a smaller number of particles to be sampled and it is relatively faster. It uses the fact that in each simulation time step only a small fraction of charged particles suffer collisions with the neutrals. Hence, there is no necessity to analyze all particles. It also uses a null collision constrain and as a first step calculates the maximum collision probability during the PIC simulation time step Δt:

$$P_{\max} = 1 - \exp(-\nu_{\max}\Delta t)\ , \tag{119}$$

The maximum number of particles which can suffer a collision per Δt time is given as $N_{nc} = P_{\max} N \ll N$. As a result only N_{nc} particle per time step have to be analyzed. These N_{nc} particles are randomly chosen, e.g., by using the expression $i = R_j N$ with $j = 1, \ldots, N_{nc}$, where i is the index of the particle to be sampled and R_j are the random numbers between 0 and 1. The sampling procedure itself includes the calculation of the collision probability of a sampled particle and choosing which kind of collision it should suffer, which is similar to (113 - 115): if

$$R \le \frac{P_1}{P_{\max}} \approx \frac{\nu_1}{\nu_{\max}}\ , \tag{120}$$

a type 1 collision takes place; else if

$$R \le \frac{\nu_1 + \nu_2}{\nu_{\max}}\ , \tag{121}$$

a type 2 collision takes place, and so on. If

$$R > \frac{\nu}{\nu_{\max}} \tag{122}$$

no collision takes place.

The difference between the nonlinear and linear null collision operators is the way how the collided neutral particles are treated. In the linear model the neutral velocity is *picked up* from the prescribed distribution (usually the Maxwellian distribution with the given density and temperature profiles). Contrary to this, in the nonlinear case the motion of neutral particles is resolved in the simulation, and the collided ones are randomly chosen from the same cells, where the colliding charged-particles are located (see [15]).

When the collision partners and corresponding collision types are chosen, the collision itself takes place. We note that the collision models described below are equally applicable to both the direct MC and null collision methods. Each collision type needs a separate consideration, so that here we discuss the general principle.

The easiest collisions are the ion–neutral charge-exchange collisions. In this case the collision is reduced to an exchange of velocities:

$$\vec{V}_1' = \vec{V}_2 \qquad \text{and} \qquad \vec{V}_2' = \vec{V}_1\ . \tag{123}$$

The recombination collisions are also easy to implement. In this case the collided particles are removed from the simulation and the newly born particle, i.e. the recombination product, has the velocity derived from the momentum conservation:

$$\vec{V}_{new} = \frac{m_1\vec{V}_1 + m_2\vec{V}_2}{m_{new}}\ . \tag{124}$$

The elastic collisions are treated in a similar way as the Coulomb collisions using (104). The scattering angle depends on the given atomic data. E.g., often it is assumed that the scattering is isotropic:

$$\cos\chi = 1 - 2R\ . \tag{125}$$

In order to save computational time during the electron–neutral elastic collisions the neutrals are assumed to be at rest. Accordingly, in spite of resolving (104) a simplified expression is used for the calculation of the after-collision electron velocity:

$$V_e' \approx V_e \sqrt{1 - \frac{2m_e}{M_n}\left(1 - \cos\chi\right)} \,. \tag{126}$$

Excitation collisions are done in a similar way as the elastic ones, just before the scattering the threshold energy E_{th} is subtracted from the charged particle energy:

$$\vec{V} \;\Rightarrow\; \vec{V}' = \vec{V}\sqrt{1 - \frac{E_{th}}{E}} \;\Rightarrow\; \text{scattering} \;\Rightarrow\; \vec{V}'' \,. \tag{127}$$

Important to note is that one has to take care on the proper coordinate system, e.g., in (127) the first transform should be done in a reference system, where the collided neutral is at rest.

Implementation of inelastic collisions when secondary particles are produced is case dependent. E.g., in electron–neutral ionization collisions, first the neutral particle is removed from the simulation and a secondary electron–ion pair is born. The velocity of this ion is equal to the neutral particle velocity. The velocity of electrons is calculated in the following way. First, the ionization energy is subtracted from the primary electron energy and then the rest is divided between the primary and secondary electrons. This division is done according to given atomic data. After finding these energies the electrons are scattered on the angles χ_{prim} and $\chi_{\rm sec}$. For further details see [15].

In a similar way are treated the neutral–neutral and inelastic charged–charged particle collisions.

Short review of collision models including collision of particles with different weight can be found in [54].

9 Dust-plasma interactions

Recent years dusty plasma research has become a rapidly developing part of plasma physics (see [55] and references therein). Accordingly, number of PIC codes including plasma-dust interaction have been developed. Dust particles are much heavier than the plasma ions ($M_{Dust}/M_i \gg 10^3$), so that fully self-consistent PIC simulation including dust and plasma dynamics is an impossible task for our day computers. An interesting exception is the paper [56] reporting on PIC simulations of dusty plasma with artificially reduced mass ratio $M_{Dust}/M_i = 640$. Typically in PIC simulations the dust particles are assumed to be immobile and the plasma-dust interaction is modeled via electrostatic interaction of charged particles (i.e. dust represents an additional ion species) and plasma particle absorption at the dust. These codes can be divided into two parts according to the dust charging model they use.

The simplified models (e.g. see [57]- [60]) use a usual PIC-MC particle collision operator for describing ion (electron) absorption at the dust. The corresponding collision cross-section is obtained from analytic Orbit Motion Limited (OML) theory [61]:

$$\sigma_{OML} = \pi R_d^2 \left(1 - \frac{q_d q_i}{4\pi\varepsilon_0 R_d E_i}\right), \tag{128}$$

where R_d, q_d, q_i and E_p are the dust radius and charge, and ion charge and kinetic energy, respectively. This model is sufficiently fast, but can fail when plasma is strongly nonuniform and/or nonmaxwellian.

Other codes use a self-consistent model when ion trajectories are followed precisely until they are absorbed (or scattered) at the dust. In [62] - [64] the authors consider plasma interaction with a "large" size ($R_d > \lambda_D$) spherical dust. Dust shape and nonuniform surface charge are properly reproduced in these models ensuring high accuracy of simulation results. One has to note, that this approach is applicable only for $R_d > \Delta x$, so that the electric field at the dust surface can be directly calculated from the surface charge $E_d = \sigma_d/\varepsilon_0$. Otherwise the dust-ion interaction inside the cell and the ion motion near the dust can strongly deviate from realistic one (see Fig. (6)). In order to simulate small size dust ($R_d < \lambda_D \sim \Delta x$) in [65] a combined 3D PIC - molecular dynamic simulation has been performed. The simulation domain was divided into two parts. Plasma particles inside the cell with the dust have beet treated by molecular dynamic approach, i.e. interparticle forces where calculated according to the exact analytical expressions. Outside this cell a usual PIC was employed (see Fig. 10).

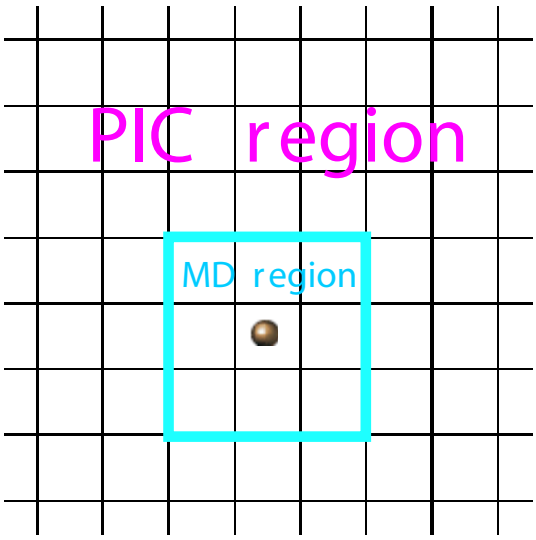

Fig. 10 Computational domain for PIC-MD simulations of a dust in plasma [65]

10 Optimization of PIC codes

The PIC codes represent one of the most time consuming numerical codes used in physics. Some of heavy simulations employ up to 10^{10} particles, run 10^7 time steps and require 10^4 hours of CPU. Hence, optimization of PIC code has become a necessary task. Some of optimization tools can reduce the required CPU time by few tens of percents saving weeks of simulation time for a single processor (serial) codes. Brief reviews of PIC optimization methods can be found in [66]- [68]. Most effective ones are parallelization of the code and so called *sorting*. Parallelization of numerical codes is a common technique use in all branches of numerical science. The idea is to distribute the job between different processors. This method is case dependent, i.e. it depends on the code to be paralleled and the machine where it will run, so that we will leave this subject out of the scope of this paper. The interested reader can find description of this technique for PIC codes e.g. in [69], [70].

In order to understand sorting method one has to realize that the position of particle in simulation area and computer memory are decoupled. The neighboring (in computer memory) particles can be far away from each other in the simulation area suffering different forces and different physics (see. Fig. 8a). Hence, running over particle index the routines like particle weighting, particle mover and collision operators have to operate with different external variables (fields, densities) at different grid points. As a result, the probability that this values will be found in a fast (cache) memory can be low. In order to increase the probability of cache hit number of authors [66], [71]- [73] introduce particle sorting according to the grid cells. The sorting routine are applied before calling the particle mover and weighting operations, so that operations on neighboring particles require the same external variables (see Fig. 11b). As it was demonstrated in [73] by introducing a fast sorting operator the simulation speed can increase up to 70%. In [74] and [7] the authors went further developing a new methods of particle representation in memory based on their position. In [74] particle array has a strict structure, which is ordered according to spatial cells: e.g. particles from first cell are first particles in the array, followed by the particles from the second cell and so on. In order to distinguish different cells an additional array is introduced where indexes of the last particles of each cell are saved. In [7] particle array has an additional index corresponding to the cell index. E.g. for a 1D code the particle coordinate is represented as the array $x[i][j][k]$, where i, j and k indexes correspond to the particle type, cell index and particle index inside the given cell. It appears that the memory representation of this type can be successfully used also in collision operators reducing corresponding CPU time up to 50%.

Although PIC codes represent attractive tools for plasma simulations, they have number of disadvantages, which restrict applicability of these codes. This drives an interest for developing of new tools, which are able to substitute PIC codes in some applications. One of this approaches is the grid-free model based on the *treecode* algorithm. Originally the treecode method was developed for astrophysical applications [75], but later it was successfully used for plasma simulations too (see [76]- [80]). The idea of the treecode method is to divide particle-particle interaction into two groups:

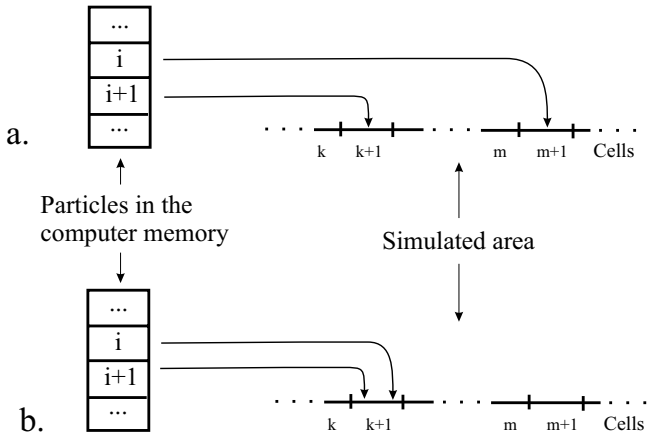

Fig. 11 Representation of two neighboring particles in a conventional PIC code (a) and in a code with sorting (b) ([7]).

1. Particles which are sufficiently far from a given particle are grouped in clusters. Interaction of this given particle with the particles from different clusters is represented by particle-cluster interaction. This operator scales as $O(N \ln N)$ and significantly faster than usual particle-particle interaction scaling as $O(N^2)$.
2. Interaction of neighboring particles is treated by usual particle-particle one. The criteria for division of particle interaction on particle-cluster and particle-particle interactions can be the distance from the given particle.

The treecode algorithm starts with division of simulation area on clusters: first the largest clusters are defined, then each of them is divided on sub-clusters, and so on until the smallest clusters are is defined. Accordingly the force acting on the particle "i" is divided on particle-cluster interaction

$$\overrightarrow{F}_i = \sum_{j \neq i} \overrightarrow{F}\left(\overrightarrow{x}_i - \overrightarrow{x}_j\right) = \sum_{k=1}^{N_C} \overrightarrow{F}\left(\overrightarrow{x}_i, C_k\right), \tag{129}$$

where C_k, denote clusters $k = 1$,...,N_C and

$$\overrightarrow{F}\left(\overrightarrow{x}_i, C_k\right) = \sum_{j \in C_k} \overrightarrow{F}\left(\overrightarrow{x}_i - \overrightarrow{x}_j\right) \tag{130}$$

represents the force acting on i particle from k cluster. Expanding $\overrightarrow{F}\left(\overrightarrow{x}_i, C_k\right)$ in Taylor series about the cluster center $\overrightarrow{x}_c$ we obtain (for simplicity we consider 1D case)

$$F\left(x_i, C_k\right) = \sum_{p=1_k}^{\infty} \frac{1}{p!} \frac{\partial^p}{\partial x_c^p} F\left(x_i - x_c\right) \sum_{j \in C_k} \left(x_c - x_j\right)^p = \sum_{p=1_k}^{\infty} T^p\left(x_i - x_c\right) M_{C_k}^p. \tag{131}$$

Here, T^p and M_C^p represent the pth Taylor coefficient of the force and the pth moment of the C cluster. Note that cluster moments are independent of x_i. Hence, they can be calculated once (each time step) and used for different particles. In practice only finite Taylor series are used with $p < 10$, so that calculation of M_C^p can be performed in a sufficiently effective way.

During the simulation calculation of the force acting on particles starts with particle-largest cluster interaction. If the distance between the particle and cluster is sufficiently large

$$|x_i - x_c| > \alpha R_C \tag{132}$$

where R_C is the size of the cluster and $\alpha > 1$ is the coefficient specifying the accuracy of the calculation, then the interaction with the next large cluster is considered and so on. If the condition (132) is not satisfied then the corresponding sub-clusters are considered and so on until either the condition is satisfied, or the smallest sub-cluster is reached. In the later the force is calculated by direct summation (130) over all particles from the given sub-cluster.

The advantage of the grid-free approach is the ability to simulate complex nonuniform geometries, which in case of PIC would require too tiny gridding and consequently too long simulation times. Moreover, grid-free codes do not suffer problems connected with the interaction at short distances, as it is case for PIC (see Fig. 8). For comparison in Fig. 12 are plotted potential profiles in the plasma sheath obtained from the treecode and PIC simulations [80]. As we see in a strongly nonuniform plasma PIC can reproduce the same results as treecode only when a very tiny gridding is used.

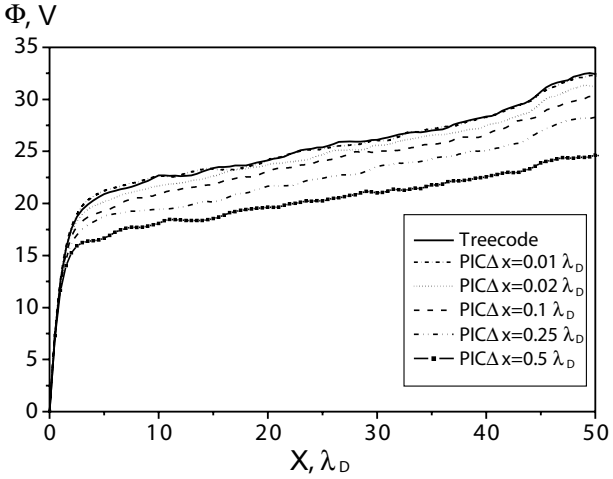

Fig. 12 Potential calculated with the treecode and with the PIC using different cell sizes [80].

Other interesting alternative to PIC codes is the Complex Particle Kinetic (CPK) concept presented in [81]. CPK is kind of hybrid of the fluid and PIC codes, when fluid-like macro-particles with Gaussian shapes in space and Maxwellian velocity profiles are used. It allows to simulate kinetic dynamics for significantly short time, as it required for PIC codes. The necessary condition to explore this advantage is that the simulated system has to be almost in the fluid regime.

Acknowledgement: This work, supported by the European Communities under the Contract of Association between EURATOM and AW, was carried out within the framework of the European Fusion Development Agreement (EFDA). The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] O. Buneman, Phys. Rev., **115**, 503-517 (1959).
[2] J.M. Dawson, Phys. Fluids, **5**, 445-459 (1962).
[3] D. Tskhakaya, Paragraph 6 in Computational Many-Particle Physics, Eds.: H. Fehske, R. Schneider and A. Weiße, Springer Verlag, 2007.
[4] C.K. Birdsall, A.B. Langdon, Plasma Physics Via Computer Simulation, edited by S. Rao and M. Eichenberg, McGraw-Hill, New York, 1985.
[5] R.W. Hockney and J.W. Eastwood, Computer Simulation Using Particles, edited by A. Hilger, IOP, Bristol and New York, 1989.
[6] J.P. Verboncoeur, A.B. Langdon and N.T. Gladd, Comp. Phys. Comm., **87**, 199-211 (1995).
[7] D. Tskhakaya and R. Schneider, J. of Comp. Phys., **225** (1), 829-839 (2007).
[8] D.C. Barnes, T. Kamimura, J.-N. Le Boeuf, and T. Tajima, J. Comput. Phys., **52**, 480-502 (1983).
[9] J.N. Brooks and D. Naujoks, Phys. Plasmas, **7** (6), 2565-2570 (2000).
[10] F. Taccogna, S. Longo, M. Capitelli, Phys. Plasmas, **11** (3), 1-7 (2004).
[11] F. Taccogna, S. Longo, M. Capitelli, Vacuum, **73**, 89-92 (2004).
[12] F. Taccogna, S. Longo, M. Capitelli, Phys. Plasmas, **12**, 093506 (2005).
[13] F. Taccogna, R. Schneider, K. Matyash, S. Longo, M. Capitelli and D. Tskhakaya, J. Nucl. Mater., **363-365**, 437-442 (2007).
[14] F. Taccogna, R. Schneider, K. Matyash, S. Longo, M. Capitelli and D. Tskhakaya, Contrib. Plasma Phys., (2007) accepted for publication.
[15] D. Tskhakaya, S. Kuhn, Y. Tomita, K. Matyas, R. Schneider and F. Taccogna, Contrib. Plasma Phys., (2007) accepted for publication.

[16] J.P. Verboncoeur, M.V. Alves, V. Vahedi, and C.K. Birdsall, J. Comput. Phys., **104** (2), 321-328 (1993).
[17] D. Tskhakaya and S. Kuhn, Contrib. Plasma Phys., **42** (2-4), 302-308 (2002).
[18] K.L. Cartwright, J.P. Verboncoeur and C.K. Birdsall, J. Comput. Phys., **162** (2), 483-513 (2000).
[19] H.C. Kim, Y. Feng and J.P. Verboncoeur, J. Comput. Phys., **223** (2), 629-642 (2007).
[20] D. Tskhakaya and S. Kuhn, Plasma Phys. Control. Fusion, **47**, A327-A337 (2005).
[21] J.P. Verboncoeur, J. Comput. Phys., **174**, 421-427 (2001).
[22] W.M. Ruyten, J. Comput. Phys., **105**, 224-234 (1993).
[23] D.J. Larson, D.W. Hewett, and A.B. Langdon, Comput. Phys. Comm., **90** (2-3), 260-266 (1995).
[24] F. Collino, T. Fouquet, P. Joly, J. Comput. Phys., **211**, 9-35 (2006).
[25] P.N. Swarztrauber, SIAM J. Numer. Anal., **11** (6), 1136-1150 (1974).
See also http://www.cisl.ucar.edu/css/software/fishpack/.
[26] V. Vahedi, G. DiPeso, J. Comp. Phys., **13**1, 149 (1997).
[27] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, Cambridge University Press, Cambridge, New York Port Chester, Melbourne, Sydney, 2002.
[28] N.A. Krall and A.W. Trivelpiece, Principles of Plasma Physics, San Francisco Press, 1986.
[29] L.D. Landau, J. Phys. (USSR), **10**, 25-34 (1946).
[30] B.D. Fried and S.D. Conte, The Plasma Dispersion Function, Academic Press, New York 1961.
[31] L.D. Landau, E.M. Lifshits, Statistical physics, .Elsevier Science & Technology (UK), 1996.
[32] P.W. Rambo, J. Ambrosiano, A. Friednam, and D.E. Nielsen Jr., Proceedings of 13th Conference on the Numerical Simulation of Plasmas, Santa Fe, New Mexico 1989.
[33] A. Bergmann, Contrib. Plasma Phys., **38**, 231 (1998).
[34] O.V. Batishchev, X.Q. Xu, J.A. Byers, R.H. Cohen, S.I. Krasheninnikov, T.D. Rognlien, and D.J. Sigmar, Phys. Plasmas, **3** (9), 3386-3396 (1996).
[35] O.V. Batishchev, S.I. Krasheninnikov, P.J. Catto, et al, Phys. Plasmas, **4** (5), 1672-1680 (1997).
[36] T. Takizuka and H. Abe, J. Comput. Phys., **25** (3), 205-219 (1977).
[37] N.A. Krall and A.W. Trivelpiece, Principles of Plasma Physics, San Francisco Press, Inc., Box 6800, San Francisco, 1986.
[38] L.D. Landau and E.M. Lifshitz, Course of Theoretical Physics, Vol. 1, Mechanics, Pergamon Press, Oxford-London-Paris, 1960.
[39] R. Shanny, J.M. Dawson, and J.M. Greene, Phys. Fluids, **10** (6), 1281-1287 (1967).
[40] W.X. Wang, M. Okamoto, N. Nakajima, and S. Murakami, J. Comput. Phys., **128** (1), 209-222 (1996).
[41] D.L. Book, NRL Plasma formulary, Naval Research Laboratory, Washington D.C., 1978.
[42] K. Nanbu, Phys. Rev., E **55** (4), 4642-4652 (1997).
[43] A.V. Bobylev and K. Nanbu, Phys. Rev., E **61** (4), 4576-4586 (2000).
[44] G.A. Bird, Molecular Gas Dynamics and the Direct Simulation of Gas Flows, Oxford: Oxford Science, 1994.
[45] S. Longo, Plasma Sources Sci. Technol. **15**, 181-188 (2006).
[46] V.V. Serikov, S. Kawamoto, K. Nambu, IEEE Trans. Plasma Sc., **27** (5), 1389-1398 (1999).
[47] K. Matyash, R. Schneider, A. Bergmann, W. Jacobb, U. Fantz, P. Pecher, J. Nucl. Mater., **313-316**, 434-438 (2003).
[48] F. Taccogna, R. Schneider, S. Longo, M. Capitelli, Phys. Plasmas, **14**, 073503 (2007).
[49] F. Taccogna, R. Schneider, S. Longo, and M. Capitelli, Rev. Sci. Instrum., (2007) accepted for publication.
[50] F. Taccogna, S. Longo, M. Capitelli, and R. Schneider, IEEE Trans Plasma Sci., (2007) accepted for publication.
[51] H.R. Skullerud, J. Phys. D: Appl. Phys., **1,** 1567-1569 (1968).
[52] C.K. Birdsall, IEEE Trans. Plasma Sc., **19** (2), 65-85 (1991).
[53] V. Vahedi, M. Surendra, Comp. Phys. Comm., **87**, 179-198 (1995).
[54] K. Nambu, IEEE Trans. Plasma Sc., **28** (3), 971-990 (2000).
[55] P.K. Shukla and A.A. Mamun, Introduction to Dusty Plasma Physics, IoPP, Bristol and Phyladelphia, 2002.
[56] K. Matyash, R. Schneider, Contrib. Plasma Phys., **44** (1-3), 157-161 (2004).
[57] O. Yu. Kravchenko, Yu.I. Chutov, W.J. Goedheer, R.D. Smirnov and S. Takamura, J. Nucl. Mater., **313-316**, 1109-1113 (2003).
[58] Yu.I. Chutov, O.Yu. Kravchenko, S. Masuzaki, A. Sagara, R.D. Smirnov, Yu. Tomita, Contrib. Plasma Phys., **44** (1-3), 138-143 (2004).
[59] R. Smirnov, Y. Tomita, T. Takizuka, A. Takayama, Yu. Chutov, Contrib. Plasma Phys., **44** (1-3), 150-156 (2004).
[60] W.J. Goedheer, M.R. Akdim, Yu.I. Chutov, Contrib. Plasma Phys., **44** (1-3), 395-404 (2004).
[61] J.E. Allen, Phys. Scripta, **45**, 497-503 (1992).
[62] I.H. Hutchinson, Plasma Phys. Control. Fusion, **45**(8), 1477-1500 (2003).
[63] I.H. Hutchinson, Plasma Phys. Control. Fusion, **47**(1), 71-87 (2005).
[64] R. Smirnov, Y. Tomita, D. Tskhakaya, and T. Takizuka, Contrib. Plasma Phys., **46** (7-9), 623-627 (2006).
[65] K. Matyash, R. Schneider, F. Taccogna and D. Tskhakaya, J. Nucl. Mater., **363-365**, 458-461 (2007).
[66] V.K. Decyk, S.R. Karmesin, A. de Boer, P.C. Liewer, Comput. Phys., **10**, 290-298 (1996).
[67] E. Kawamura, C.K. Birdsall and V. Vahedi, Plasma Sources Sci. Technol., **9** (3), 413-428 (2000).

[68] J.D. Blahovec, Jr., L.A. Bowers, J.W. Luginsland, Member, IEEE, G.E. S. Sasser, and J.J.Watrous, IEEE Trans. Plasma Sc., **28** (3), 821-829 (2000).
[69] V.K. Decyk, Comp. Phys. Comm., **87**, 87-94 (1995).
[70] V.K. Decyk, Ch.D. Norton, Comp. Phys. Comm., **164**, 80-85 (2004).
[71] D.V. Anderson, D.E. Shumaker, Comp. Phys. Comm., **87** 16-34 (1995).
[72] T. MacFarland, H.M.P. Couchman, R.F. Pearce, J. Pichlmeier, New Astron., **3**, 678-705 (1998).
[73] K.J. Bowers, J. Comput. Phys., **173**, 393-411 (2001).
[74] R.J. Thacker, H.M.P. Couchman, Comp. Phys. Comm., **174**, 540-554 (2006).
[75] J. Barnes and P. Hut, Nature, **324**, 446-449 (1986).
[76] S. Pfalzner and P. Gibbon, Many-Body Tree Methods in Physics, Cambridge, U.K.: Cambridge Univ. Press, 1996.
[77] S. Pfalzner and P. Gibbon, Phys. Rev. E, **57**, 4698-4705 (1998).
[78] A.J. Christlieb, R. Krasny, and J.P. Verboncoeur, IEEE Trans Plasma Sci, **32**, 384-389 (2004).
[79] A.J. Christlieb, R. Krasny and J.P. Verboncoeur, Comput. Phys. Comm., **164**, 306-310 (2004).
[80] K. Matyash , R. Schneider, R. Sydora and F. Taccogna, Contrib. Plasma Phys., to be published (2007).
[81] D.W. Hewett, J. Comput. Phys., **189**, 390-426 (2003).

