2D Arc-PIC Code Description:

Methods and Documentation

Helga Timkó

2010/2011

Helsinki Institute of Physics

and

CERN
INTRODUCTION

Vacuum discharges are one of the main limiting factors for future linear collider designs such as that of the Compact Linear Collider. To optimise machine efficiency, maintaining the highest feasible accelerating gradient below a certain breakdown rate is desirable; understanding breakdowns can therefore help us to achieve this goal. As a part of ongoing theoretical research on vacuum discharges at the Helsinki Institute of Physics, the build-up of plasma can be investigated through the particle-in-cell method. For this purpose, we have developed the 2D Arc-PIC code introduced here.

We present an exhaustive description of the 2D Arc-PIC code in two parts. In the first part, we introduce the particle-in-cell method in general and detail the techniques used in the code. In the second part, we provide a documentation and derivation of the key equations occurring in the code.

The code is original work of the author, written in 2010, and is therefore under the copyright of the author.

ACKNOWLEDGMENTS

The development of the code has been carried out in frame of a collaboration between the Helsinki Institute of Physics and CERN. The professional guidance of Prof. R. Schneider and Ph.D. K. Matyash from the Max-Planck-Institut für Plasmaphysik is gratefully acknowledged.

The research has received funding from the European Commission under the FP7 Research Infrastructures project EuCARD, grant agreement no. 227579. For further details, see [http://cern.ch/eucard](http://cern.ch/eucard).
## Contents

### 1 Methods

1.1 The particle-in-cell method .................................................. 1
  1.1.1 Superparticles .............................................................. 1
  1.1.2 Field solver ................................................................. 2
  1.1.3 Particle mover ............................................................... 2
  1.1.4 Particle and field weighting ............................................ 3
  1.1.5 Collisions ................................................................. 4
  1.1.6 Accuracy and stability conditions .................................... 5

1.2 The 2D Arc-PIC model ......................................................... 5

### 2 Documentation

2.1 Scaling in 2D PIC ................................................................. 7
2.2 Electric field calculation ....................................................... 13
2.3 Interpolation scheme ............................................................ 13
2.4 Particle pusher equations ..................................................... 14
  2.4.1 Electrostatic case .......................................................... 14
  2.4.2 Magnetic case .............................................................. 15
2.5 Generating a uniform density in 2D PIC ................................. 17
2.6 Particle injection ............................................................... 18
  2.6.1 Generating velocities ..................................................... 18
  2.6.2 Fowler-Nordheim field emission ...................................... 19
2.7 The discharge gap as part of an electric circuit ......................... 19
2.8 Density calculation ............................................................. 21
2.9 Monitoring energy conservation ............................................ 22
2.10 Outputting using `moms.cpp` and `outputz.cpp` ......................... 24
2.11 Potential and electric field boundary conditions ....................... 27
2.12 Particle boundary conditions .............................................. 31
2.13 Particle injection schemes ................................................ 32
2.14 Collision routines ........................................................... 33
  2.14.1 Coulomb collisions for e⁻-e⁻ and Cu⁺-Cu⁺ pairs ................... 33
  2.14.2 Other collisions ......................................................... 34
2.15 Control routines, stability of PIC ......................................... 35
2.16 Miscellaneous useful formulae ............................................ 36
Chapter 1

Methods

1.1 The particle-in-cell method

As an introduction to the code, we shall first briefly review the particle-in-cell (PIC) method\footnote{Part of this introduction is based on my contribution to the wikipedia article ‘Particle-in-cell’, http://en.wikipedia.org/wiki/Particle-in-cell} a method that has been developed in the 1950’s by Buneman \footnote{1}, Dawson \footnote{2}, and others. A more thorough description of it can be found in the classic textbooks of Hockney & Eastwood \footnote{3} or of Birdsall & Langdon \footnote{4}.

The PIC method is a technique widely applied to all sorts of plasma simulations, since it combines the description of individual ‘components’ with the solution of partial differential equations (PDE). In plasma physics, these individual components translate to particles or fluid elements, while the set of PDEs translates to Maxwell’s equations; and thus the PIC method allows for a kinetic description of a plasma system. Why the method is called ‘particle-in-cell’ will become clear somewhat later in conjunction with the weighting scheme.

In PIC, particles and plasma macro-quantities, such as density, current density, or potential are treated in a different way. Particles are followed in continuous phase space, in a Lagrangian (moving) frame, while macro-quantities are discretised onto mesh points and described in a Eulerian (stationary) frame of reference. The temporal description is the same for both particles and macro-quantities, discretised into time steps; time steps for different particle species may vary though. In that case, the fastest species and the macro-quantities should have the same temporal resolution.

The whole simulation procedure is summarised schematically in Fig. 1.1. First, the simulation is set up by choosing simulation parameters (grid size, time step, superparticle weighting, etc.) adequate for the physical problem. By ‘adequate’ we mean that parameters should be ‘well-guessed’ in order to ensure the stability and accuracy of the code; we will come back to the criteria at the end of this section. Once the simulation is set up, the problem solving consists of a loop over following steps:

1. Solving Maxwell’s equations
2. Obtaining fields and forces acting on the particles
3. Moving particles by solving the equations of motion (EOM)
4. Apply collisions, boundary conditions for particles (injection, absorption, etc.)
5. Update macro-quantities that are sources to Maxwell’s equations

We shall not describe all of these points from a general point of view. Instead, below we will focus on some important details of a PIC simulation. Methods specific to the 2D Arc-PIC code will be detailed in Sec. 1.2.

1.1.1 Superparticles

Since usually the number of particles that would have to be simulated to model a real system exceeds our computational capacities, the PIC method makes use of so-called ‘superparticles’. These superparticles represent many real particles at the same time, that is, the real number of particles in the system is ‘rescaled’ to a smaller number of computational particles in the model. We may do so, because the particles move according to the Lorentz force, which depends only on the charge-to-mass ratio, and the charge-to-mass ratio is the same for both real and superparticles.

1.1.2 Field solver

Maxwell’s equations are solved in the field solver part of PIC. The most common approaches for solving PDEs use one of the following three methods:

**Finite difference method (FDM)** that replaces the continuous domain with a discrete grid of points; electromagnetic fields are calculated then on this grid. Derivatives are approximated with differences between neighbouring grid-point values and thus the FDM turns PDEs into algebraic equations.
1.1. THE PARTICLE-IN-CELL METHOD

Finite element method (FEM) that divides the continuous domain into a discrete mesh of elements and the PDEs are then treated as an eigenvalue problem. Initially, a trial solution is calculated using basis functions that are localised in each element. The final solution is obtained by optimising the trial solution until a certain required accuracy is reached.

Spectral methods such as the fast Fourier transform. Also in this case, the PDEs are transformed into an eigenvalue problem, however, this time the basis functions are high order and defined globally over the whole domain. The domain itself is therefore not discretised, it remains continuous. Again, a trial solution is found by inserting the basis functions into the eigenvalue equation and is then optimised to determine the best values of the initial trial parameters.

1.1.3 Particle mover

The EOM of particles is solved in the so-called particle mover or pusher, which is often the most time consuming part of PIC, and is therefore required to be of high accuracy and speed.

The schemes used for the particle mover can be divided into implicit and explicit solvers. While implicit solvers calculate the particle velocity from the already updated fields, explicit solvers use only the old force from the previous time step, and are therefore simpler and faster, but require a smaller time step. Two frequently used schemes are the leapfrog method [4], which is an explicit solver, and the Boris scheme [5], which is implicit.

For plasma applications, the leapfrog method takes the following form:

\[ \frac{x_{k+1} - x_k}{\Delta t} = v_{k+1/2}, \]  
\[ \frac{v_{k+1/2} - v_{k-1/2}}{\Delta t} = \frac{q}{m} \left( E_k + \frac{v_{k+1/2} + v_{k-1/2}}{2} \times B_k \right), \]

where the subscript \( k \) refers to ‘old’ quantities from the previous time step, \( k + 1 \) to updated quantities from the next time step (i.e. \( t_{k+1} = t_k + \Delta t \)), and velocities are calculated in-between the usual time steps \( t_k \).

In comparison, the equations of the Boris scheme are:

\[ x_{k+1} = x_k + \Delta t v_{k+1/2}; \]  
\[ v_{k+1/2} = u' + q'E_k, \]

with

\[ u' = u + (u \times (u \times h)) \times s, \]  
\[ u = v_{k-1/2} + q'E_k, \]  
\[ h = q'B_k, \]  
\[ s = 2h/(1 + h^2), \]  
\[ q' = \Delta t/(2q/m). \]
1.1.4 Particle and field weighting

The origin of the name ‘particle-in-cell’ is in the way how plasma macro-quantities are assigned to simulation particles, that is, the particle weighting. Particles can be situated anywhere in the continuous domain, but macro-quantities are calculated only on the mesh points, just as the fields are. To obtain the macro-quantities, one assumes that the particles have a given ‘shape’ determined by the shape function

$$S(x - X),$$

(1.10)

where $x$ is the coordinate of the particle and $X$ the observation point (usually the grid point $x_i$). Perhaps the easiest and most used choice for the shape function is the so-called cloud-in-cell (CIC) scheme, which is a first order (linear) weighting scheme. In any case, whatever the scheme is, the shape function has to satisfy the following conditions [6]: Space isotropy, charge conservation, and increasing accuracy (convergence) for higher-order terms.

The fields obtained from the field solver are determined only on the grid points and can’t be used directly in the particle mover to calculate the force acting on particles, but have to be interpolated via the field weighting:

$$E(x) = \sum_i E_i S(x_i - x),$$

(1.11)

where the subscript $i$ labels the grid point. To ensure that the forces acting on particles are self-consistently obtained, the way of calculating macro-quantities from particle positions on the grid points and interpolating fields from grid points to particle positions has to be consistent, too, since they both appear in Maxwell’s equations. Above all, the field interpolation scheme should conserve momentum. This can be achieved by choosing the same weighting scheme for particles and fields and by ensuring the appropriate space symmetry (i.e. no self-force and fulfilling the action-reaction law) of the field solver at the same time [6].

1.1.5 Collisions

As the field solver is required to be free of self-forces, inside a cell the field generated by a particle must decrease with decreasing distance from the particle, and hence inter-particle forces inside the cells are underestimated. This can be balanced with the aid of Coulomb collisions between charged particles. Simulating the interaction for every pair in a large system would be computationally too expensive, so several Monte Carlo methods have been developed instead. A widely used method is the binary collision model [7], in which particles are grouped according to their cell, then these particles are paired randomly, and finally the pairs are collided.

In a real plasma, many other reactions may play a role, ranging from elastic collisions, such as collisions between charged and neutral particles, over inelastic collisions, such as electron-neutral ionisation collisions, to chemical reactions; each of them requiring separate treatment. Most of the collision models handling charged-neutral collisions use either the direct Monte-Carlo scheme, in which all particles carry information about their collision probability, or the null-collision scheme [8] [9], which does not analyse all particles but uses the maximum collision probability for each charged species instead.
1.1.6 Accuracy and stability conditions

As in every simulation method, also in PIC, the time step and the grid size must be well chosen, so that the shortest time- and length scale phenomena are properly resolved in the problem. In addition, time step and grid size have also an impact on the speed and accuracy of the code.

As a rule of thumb, two important conditions regarding the grid size $\Delta x$ and the time step $\Delta t$ should be fulfilled in order to ensure the stability of the solution:

\begin{align}
\Delta x &< 3.4 \lambda_{DB}, \\
\Delta t &\leq 0.2 \omega_{pe}^{-1},
\end{align}

which can be derived considering the harmonic oscillations of a one-dimensional unmagnetised plasma [10]. Perhaps not surprisingly, the natural length scale in the plasma is given by the Debye length $\lambda_{DB}$ (Eq. 2.198) and time scale by the inverse plasma frequency $\omega_{pe}^{-1}$ (Eq. 2.199).

1.2 The 2D Arc-PIC model

The 2D Arc-PIC code can be classified as a 2d3v PIC-MCC code. The ‘MCC’ stands for the code being equipped with Monte-Carlo collision routines. The 2d3v means that particles are described in a five-dimensional phase space with two coordinate and three velocity components. The geometry of the coordinate space is (r,z), that is, cylindrical symmetry of the system simulated is assumed. This two-dimensional (r,z)-space can be pictured as an ‘infinitesimal’ slice of a full three-dimensional cylindrically symmetric system (see Fig. 1.2). Indeed, in the pusher of the code (Sec. 2.14), particles are moved according to their three velocity components in three dimensions, and are then rotated back into the ‘plane of reference’, the simulation space. A uniform grid

![Figure 1.2: Illustration of the (r,z)-geometry of the code, which can be interpreted as an infinitesimal slice of a three-dimensional cylindrically symmetric system.](image)

is used throughout all the simulation system, with equal grid size in both coordinates $\Delta z = \Delta r$. 


Phenomena are described electrostatically only, that is, instead of solving the full set of Maxwell’s equations, only Poisson’s equation is taken into account (Sec. 2.1). The field solver transforms with the FDM the Poisson equation into a matrix equation and uses the external package SuperLU [11] for inverting the matrix with the so-called ‘lower-upper factorisation method’. The particle pusher (Sec. 2.4) uses the Boris method. Although the field solver is purely electrostatic, the code supports an optional external magnetic field, too (also in this case, the EOMs are solved with the Boris method). For inter- or extrapolation between particle positions and grid points the CIC weighting scheme is used (Secs. 2.3 & 2.8).

Some special features of the code related to vacuum discharges are certain boundary conditions, such as Fowler-Nordheim field emission of electrons (Sec. 2.6.2), an external circuit model (Sec. 2.7), and collisions (Sec. 2.14), which play a key role in the process.

For optimisation reasons, all the physical variables appearing in the code are rescaled to dimensionless units, so that unnecessary multiplications with constants are avoided. In Chapter 2.1 we will introduce the method of rescaling and derive all key equations in the form they appear in the code. However, may it be pointed out here that since only dimensionless quantities are used in the code, in fact, results could apply to ‘any’ time- and length scale. Only in two places in the code, namely in the collision (Sec. 2.14) and particle boundary condition (Sec. 2.12) routines, are we bound to certain dimensional values, which determines what the ‘physical’ scale of the end result is.

The code is written in C++ language.
Chapter 2
Documentation

2.1 Scaling in 2D PIC

To build-up and derive a fully dimensionless set of variables, we only need to rescale hierarchically the main equations that determine the evolution of the system: The first and second equation of motion together with the electric field calculation and the Poisson equation.

Here is how we shall proceed. In each case we start from the dimensional equation (in SI-units) and multiply then by a rescaling factor. This rescaling factor allows us to define dimensionless quantities (denoted by a tilde) that are then used in the code and enable us to establish the connection between dimensionless and dimensional variables.

The first equation of motion determines the displacement of electrons (or ions, that is, non-electron species) during one electron (ion) time step, shortly: $dz = v dt$, where $z$ denotes the particle’s coordinate, $v$ its velocity and $t$ time, which in the Boris scheme takes the form (2.1)

$$dz = z_i+1 - z_i-1 = (v_{i+\frac{1}{2}}^{(ion)} + v_{i-\frac{1}{2}}^{(ion)}) \Delta t_{(ion)} \cdot / \cdot \frac{1}{\Delta z}$$

Define dimensionless coordinates and velocities through the dimensional grid size and time step as rescaling factors,

$$\tilde{z} \equiv \frac{z}{\Delta z} \quad \tilde{v}_z \equiv \frac{v_z}{\Delta t} \quad \tilde{v}_{z,ion} \equiv \frac{v_{z,ion}}{\Delta t_{ion}}$$

and define also the dimensionless time step and grid size,

$$\tilde{\Delta t} = \omega_{pe} \Delta t = 0.2 (= \tilde{\omega}_{pe})$$
$$\tilde{\Delta z} = \frac{\Delta z}{\lambda_{Db}} = 0.5 (= \frac{1}{\lambda_{Db}})$$

As a shorthand notation, the operations and factors that rescale dimensional equations to dimensionless ones are written down on the right hand side of the equations, marked by a slash.
to arrive at the following dimensionless equation:

\[ d\tilde{z} = \tilde{z}_{i+1} - \tilde{z}_{i-1} = \frac{\Delta t}{\Delta z} (v_{z,i+\frac{1}{2}} + v_{z,i-\frac{1}{2}}) = \tilde{v}_{z,i+\frac{1}{2}} + \tilde{v}_{z,i-\frac{1}{2}}. \tag{2.2} \]

The above equations apply also for \( r \) and \( v_r \) in a similar fashion; the code uses a homogeneous grid with \( \Delta z = \Delta r \). Dimensional variables can be converted back from dimensionless variables as follows:

\[ z_{\text{REAL}} = \Delta z \cdot \tilde{z} \implies z_{\text{REAL}} = \check{\Delta z} \cdot \tilde{z} \tag{2.3} \]

\[ v_{\text{REAL}} = \frac{\Delta z}{\Delta t} \tilde{v} \implies v_{\text{REAL}} = \check{\Delta z} \frac{\tilde{v}}{\Delta t} \tag{2.4} \]

\[ v_{\text{ion,REAL}} = \frac{\Delta z}{\Delta t_{\text{ion}}} \tilde{v} \implies v_{\text{ion,REAL}} = \check{\Delta z} \frac{\tilde{v}}{\Delta t \Delta t_{\text{ion}}} \tag{2.5} \]

where \( \Delta t_{\text{ion}} = \Delta t_{\text{ion}} / \Delta t \) is an integer. In the special case of the thermal velocity (Eq. 2.202) we obtain

\[ \tilde{v}_{\text{te}} = \frac{v_{\text{te}} \Delta t}{\Delta z} = \frac{v_{\text{te}} \check{\Delta t}}{\check{\Delta z}} \frac{1}{\omega_{pe} \lambda_{Db}} = \frac{\check{\Delta t}}{\check{\Delta z}}. \tag{2.6} \]

**The second equation of motion** describes particle acceleration. Firstly, for **electrons** the dimensional equation is:

\[ v_{z,i+\frac{1}{2}} - v_{z,i-\frac{1}{2}} = -\frac{e}{m_e} E_{z,i} \Delta t, \quad / \cdot \frac{\Delta t}{\Delta z} \tag{2.7} \]

with \( e \) being the elementary charge, \( m_e \) the electron mass, and \( E \) the electric field. Define the dimensionless electric field as

\[ \tilde{E}_z \text{ or } r = \frac{e}{2m_e} \frac{\Delta t^2}{\Delta z} E_z \text{ or } r \]

to obtain the dimensionless equation

\[ \tilde{v}_{z,i+\frac{1}{2}} - \tilde{v}_{z,i-\frac{1}{2}} = -2 \left( \frac{e}{2m_e} \frac{\Delta t^2}{\Delta z} \right) E_{z,i} = -2\tilde{E}_z,i. \tag{2.8} \]

Note the convention of dividing the electric field contribution into two parts, hence the factor of two, which is convenient in the Boris scheme. So, in the pusher we update velocities as follows:

\[ \tilde{v} = 2\tilde{E}; \tag{2.9} \]

For this to remain true also for **ions**, the electric field is rescaled for ions in **e_ion.cpp** to an ‘ion electric field’

\[ \text{Throughout the whole document, a ‘variable’ typesetted as } \text{variable} \text{ shall always denote a certain variable appearing in the code.} \]
2.1. SCALING IN 2D PIC

\[ \tilde{E}_{\text{ion}} = -dt_{\text{ion}} \frac{m_e}{M[\text{ion}]} \sum_{i=1}^{dt_{\text{ion}}} \tilde{E}_{i_{\text{electron}}}. \]

The summation is due to the fact that for ions the pusher is called only every ion step \( \Delta t_{\text{ion}} \) instead of every time step \( \Delta t \) and the ions velocity has to be rescaled with \( \Delta t_{\text{ion}} \) as well,

\[ \tilde{v}_{z,i+\frac{1}{2}}^{\text{ion}} - \tilde{v}_{z,i-\frac{1}{2}}^{\text{ion}} = \frac{e}{M[\text{ion}]} E_{z,i} \Delta t_{\text{ion}} \left/ \frac{\Delta t_{\text{ion}}}{\Delta z} \right. \quad (2.10) \]

to obtain the same form of dimensionless equation as for electrons

\[ \tilde{\varphi}_{z,i+\frac{1}{2}}^{\text{ion}} - \tilde{\varphi}_{z,i-\frac{1}{2}}^{\text{ion}} = -2 \left( -dt_{\text{ion}}^2 \frac{m_e}{M[\text{ion}]} \right) \left( \frac{e}{2m_e} \frac{\Delta t^2}{\Delta z} \right) E_{z,i}. \quad (2.11) \]

To average out numerical fluctuations, \( \tilde{E}_{\text{ion}} \) is (i) first integrated over \( dt_{\text{ion}} \) steps,

\[ \tilde{E}_{\text{ion}} = \sum_{i=1}^{dt_{\text{ion}}} \tilde{E}_{i_{\text{electron}}} \quad (2.12) \]

and then rescaled with \(-dt_{\text{ion}} \frac{m_e}{M[\text{ion}]}\) to obtain

\[ \tilde{E}_{\text{ion}} = -dt_{\text{ion}} \frac{m_e}{M[\text{ion}]} \sum_{i=1}^{dt_{\text{ion}}} \tilde{E}_{i_{\text{electron}}} = -dt_{\text{ion}}^2 \frac{m_e}{M[\text{ion}]} \langle E_{\text{ion}} \rangle_{dt_{\text{ion}}}. \quad (2.13) \]

Hereinafter, we denote \( \vec{E} = \vec{E}_{\text{electron}} \), unless stated otherwise.

**Deriving the electric field** from the potential \( \varphi \): \( \vec{E} = -\vec{\nabla} \varphi \) takes the dimensional form

\[ E_{z,i} = \frac{\varphi_{i-1} - \varphi_{i+1}}{2\Delta z} \left/ \frac{e}{2m_e} \frac{\Delta t^2}{\Delta z} \right. \quad (2.14) \]

Defining then the dimensionless potential as

\[ \tilde{\varphi} \equiv \frac{e}{m_e} \frac{\Delta t^2}{\Delta z^2} \varphi \]

results in the dimensionless equation

\[ \tilde{E}_{z,i} = \frac{1}{4} \left( \frac{e}{m_e} \frac{\Delta t^2}{\Delta z^2} \right) (\varphi_{i-1} - \varphi_{i+1}) = \frac{1}{4} (\tilde{\varphi}_{i-1} - \tilde{\varphi}_{i+1}). \quad (2.15) \]

Dimensional variables are obtained by following rescaling factors:

\[ E_{\text{REAL}} = \frac{2m_e}{e} \frac{\Delta z}{\Delta t^2} \tilde{E} = \frac{2m_e}{e} \frac{\Delta z}{\Delta t^2} \lambda_{Dh} \omega_{pe}^2 \tilde{E} \quad (2.16) \]

\[ \varphi_{\text{REAL}} = \frac{m_e}{e} \frac{\Delta z^2}{\Delta t^2} \tilde{\varphi} = \frac{m_e}{e} \frac{\Delta z^2}{\Delta t^2} \lambda_{Dh} \omega_{pe}^2 \tilde{\varphi} \quad (2.17) \]
The latter equation can be rewritten in terms of $T_e$, since

$$
\lambda_{Dh}^2 \omega_{pe}^2 = \frac{\varepsilon_0 T_{ref} e^2 n_{ref}}{\varepsilon_0 m_e} = \frac{T_{ref}}{m_e} v_{te}^2
$$

(2.18)

and therefore

$$
m_e \frac{\Delta z^2}{\Delta t^2} \lambda_{Dh}^2 \omega_{pe}^2 = \frac{\Delta z^2}{\Delta t^2} \left( \frac{T_{ref}}{e} \right)
$$

(2.19)

$$
\Rightarrow \frac{\varphi_{REAL}[V]}{T_{ref}[eV]} = \frac{\Delta z^2}{\Delta t^2} \tilde{\varphi}
$$

(2.20)

**Poisson’s equation** is in its original form is

$$
\nabla^2 \varphi = \frac{-e}{\varepsilon_0} (n_e + n_i),
$$

with $\varepsilon_0$ being the vacuum permittivity and using the convention

$$
\begin{align*}
    n_e &< 0 \\
n_i &> 0
\end{align*}
$$

for the electron and ion number densities $n_e$ and $n_i$, respectively. First, we shall derive the scaling that applies to $n_e$. Written in cylindrical coordinates,

$$
\left( \frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{\partial^2}{\partial z^2} \right) \varphi = \frac{-e}{\varepsilon_0} n_e.
$$

(2.21)

With the finite differences scheme, for *inner* mesh points, for instance, we get the so-called ‘five-point formula’:

$$
\sum_{5-p} c_i \varphi_{i,j,k} = a \varphi_{j-1,k} + b \varphi_{j+1,k} + c \varphi_{j,k-1} + d \varphi_{j,k+1} + e \varphi_{j,k} = \frac{-e}{\varepsilon_0} n_e(j,k)
$$

(2.22)

where $j \in [0, nr]$ and $k \in [0, nz]$ enumerate mesh points, and where factors $c_i$ are rescaled as follows:

$$
a = \frac{1}{r_{j+1} - r_{j-1}} \left( -\frac{1}{r_j} + \frac{2}{r_j - r_{j-1}} \right) = \frac{1}{\Delta r^2} \frac{1}{\tilde{r}_{j+1} - \tilde{r}_{j-1}} \left( -\frac{1}{\tilde{r}_j} + \frac{2}{\tilde{r}_j - \tilde{r}_{j-1}} \right) = \frac{\Delta z^2}{\Delta r^2} \tilde{a}
$$

(2.23)

So for a uniform grid $\tilde{r}_j = j$ and for inner mesh points we have:

$$
\tilde{a} = \tilde{a}(j) = \frac{\Delta z^2}{\Delta r^2} \left( 1 - \frac{1}{2j} \right)
$$

(2.24)

$$
\tilde{b} = \tilde{b}(j) = \frac{\Delta z^2}{\Delta r^2} \left( 1 + \frac{1}{2j} \right)
$$

(2.25)

$$
\tilde{c} = \tilde{d} = 1
$$

(2.26)

$$
\tilde{e} = -2 \frac{\Delta z^2}{\Delta r^2} - 2
$$

(2.27)
where now of course $\Delta r^2 = \Delta z^2$. With the aid of the above formulae, we can then rescale Poisson’s equation

$$\sum_{5-p.} c_i \varphi_i(j, k) = -\frac{e}{\varepsilon_0 m_e} \cdot \Delta z^2 \frac{e}{m_e} \Delta l^2 \frac{1}{\Delta z^2} \sum_{5-p.} c_i \varphi_i(j, k) = -\frac{e}{\varepsilon_0 m_e} \cdot \Delta z^2 \frac{e}{m_e} \Delta l^2 \frac{1}{\Delta z^2}$$

(2.28)

Here we chose a scheme in which the volume of the cell is explicitly taken into account when density is calculated. That is, a particle in an inner cell will correspond to a higher density than a particle in an outer cell, since the outer cell has bigger volume.

In the code, two routines are coupled for the calculation of potential and density: density($n_e$) and potential($n_e$). For outputting, another two functions are coupled that re-calculate the density based on particle position (and so they have to be consequently matched, too): These are out$_n$(mom$_el$) and aver$_n$(mom$_el$). Therefore we start with the calculation of density($n_e$).

We are searching for a dimensionless charge $\tilde{q}$ such that the dimensionless density $dens(j, k)$ at a grid point $(j, k)$ would fulfil

$$\sum_{5-p.} \tilde{c}_i \tilde{\varphi}_i(j, k) = -\frac{\epsilon^2}{\varepsilon_0 m_e} \Delta t^2 n_e(j, k) = -\frac{\hat{N}_{e,cell}(j, k)}{V_{cell}(j)} \tilde{q} \equiv -\tilde{n}_e(0, k) \equiv -dens(j, k),$$

(2.29)

where $\hat{N}_{e,cell}(j, k)$ is the number of particles ‘projected’ onto the grid point $(j, k)$ (see Eq. 2.31) and $\hat{V}_{cell}(j)$ is the dimensionless volume of the cell (Eq. 2.36). Note the convention of the minus sign on the right-hand side. We can simplify this equation to

$$\frac{\epsilon^2}{\varepsilon_0 m_e} \Delta t^2 n_e(j, k) = \frac{\omega_{pe}^2 \Delta t^2 n_e(j, k)}{n_{ref}} \frac{1}{\Delta t^2} \frac{\hat{N}_{e,cell}(j, k)}{V_{cell}(j, k)}$$

(2.30)

since $\omega_{pe}^2 = \frac{e^2 n_{cell}}{\varepsilon_0 m_e}$, with $n_{ref}$ being a reference density chosen initially. In all these equations, $\hat{N}_{e,cell}(j, k)$ is obtained as follows: It is the number of particles calculated on the grid point $(j, k)$ by summing over the four neighbouring cells’ particle distribution $f(\vec{r}_p)$ and weighting it with the particle weighting function $W(\vec{r}_p - \vec{r}_{cell})$, that is

$$\sum_{5-p.} \tilde{c}_i \tilde{\varphi}_i(j, k) = -\Delta t^2 n_e(j, k) \frac{1}{n_{ref}} = -dens(j, k) = -\frac{q}{\hat{N}_{e,cell}(j, k)} \frac{\hat{V}_{cell}(j, k)}{f(\vec{r}_p)}$$

(2.31)

In general, the volume of the cell calculated with the Verboncoeur method [12] on the grid point is:

$$V_{cell}^{(real)}(j) = \begin{cases} \frac{r_0}{2}(r_0 - r_j)(2r_0 + r_j) \Delta z, & j = 0 \\ \frac{r_j + 1}{2}(r_j + 1 + r_j) - r_{j-1}(r_{j-1} + r_{j-1}) \Delta z, & j \in (0, nr) \\ \frac{r_{nr} - r_{nr-1}}{2}(r_{nr-1} + 2r_{nr}) \Delta z, & j = nr \end{cases}$$

(2.32)

independently of $k$. For a homogeneous grid with $r_j = j \Delta r$ and $\Delta r = \Delta z$,

$$V_{cell}^{(real)}(j) = \begin{cases} \frac{r_0}{2} \cdot 1 \Delta r^2 \Delta z, & j = 0 \\ \frac{r_j + 1}{2}(2j + 1) - (j - 1)(2j - 1) \Delta r^2 \Delta z, & j \in (0, nr) \\ \frac{r_{nr} - r_{nr-1}}{2}(3nr - 1) \Delta r^2 \Delta z, & j = nr \end{cases}$$

(2.33)
CHAPTER 2. DOCUMENTATION

\[
V_{\text{cell}}(j) = \begin{cases} 
\frac{\pi}{3} \Delta r^2 \Delta z, & j = 0 \\
2\pi j \Delta r^2 \Delta z, & j \in (0, nr) \\
\pi(nr - \frac{1}{3}) \Delta r^2 \Delta z, & j = nr
\end{cases}
\]

(2.34)

More generally, for \( r_{\text{min}} \neq 0 \) we would have

\[
V_{\text{cell}}^{(\text{real})}(j) = \begin{cases} 
\frac{\pi}{3} \Delta r^2 \Delta z \left( \frac{3r_{\text{min}}}{\Delta r} + 1 \right), & j = 0 \\
2\pi \Delta r^2 \Delta z \left( \frac{r_{\text{min}}}{\Delta r} + j \right), & j \in (0, nr) \\
\pi \Delta r^2 \Delta z \left( \frac{r_{\text{min}}}{\Delta r} + (nr - \frac{1}{3}) \right), & j = nr
\end{cases}
\]

(2.35)

In its dimensionless form \( \tilde{V}_{\text{cell}}(j) = \Delta z^3 \tilde{V}_{\text{cell}}(j) \)

\[
\tilde{V}_{\text{cell}}(j) = \begin{cases} 
\frac{\pi}{3}, & j = 0 \\
2\pi j, & j \in (0, nr) \\
\pi(nr - \frac{1}{3}), & j = nr
\end{cases}
\]

(2.36)

To determine now the dimensionless charge \( \tilde{q} \), we shall consider the simple case of a uniform density in the whole system (see Sec. 2.5). Choosing \( n_{\text{ref}} \) as the uniform density, \( \pi \Delta z^3 nr^2 n_z N_{\text{Db}} \) particles have to be distributed like \( \sim r \) over the whole system, that is \( nr \times n_z \) cells. In \( z \)-direction, the particle distribution is homogeneous, so in \( r \)-direction \( \pi \Delta z^3 nr^2 N_{\text{Db}} \) particles have to be distributed like \( \sim r \), over an integrated volume of \( \int_0^{nr} drr = \frac{nr^2}{2} \).

Let's assume we place all particles directly onto grid points 1, 2, ..., \( (nr - 1) \). In this case we do not need to care about the weighting and \( \tilde{N}_{e,\text{cell}}(j,k) = 2\pi \tilde{\Delta}z^3 j N_{\text{Db}} \), with \( j = 1, 2, \ldots, (nr - 1) \) and \( \int dj j = nr^2/2 \) (N.B. why we calculate here the integral and not the sum becomes clear in Sec. 2.5). For the first cell, for instance, \( \tilde{N}_{e,\text{cell}}(1,k) = 2\pi \tilde{\Delta}z^3 N_{\text{Db}}, \forall k \), while \( V_{\text{cell}}(1) = 2\pi \). Substituting then back to Eq. 2.31 we obtain:

\[
\tilde{\Delta}t^2 = \tilde{q} \frac{\tilde{N}_{e,\text{cell}}(1,k)}{V_{\text{cell}}(1)} = \tilde{q} \tilde{\Delta}z^3 N_{\text{Db}},
\]

(2.37)

from which we can extract

\[
\therefore \tilde{q} = \frac{\tilde{\Delta}t^2}{\tilde{\Delta}z^3 N_{\text{Db}}}. \]

Dimensionless charges defined in \texttt{init.cpp} and used in \texttt{density.cpp} are therefore:

\[
\tilde{q}_e = -\frac{\tilde{\Delta}t^2}{\tilde{\Delta}z^3 N_{\text{Db}}}, \quad \Rightarrow n_e < 0 \quad (2.38)
\]

\[
\tilde{q}_i = -\tilde{q}_e, \quad \Rightarrow n_i > 0 \quad (2.39)
\]
2.2 Electric field calculation

According to Eq. (2.15), for all inner cells we calculate the electric field from the potential as follows,

\[
\tilde{E}_r(j, k) = \frac{1}{4} (\tilde{\varphi}(j - 1, k) - \tilde{\varphi}(j + 1, k))
\]

\[
\tilde{E}_z(j, k) = \frac{1}{4} (\tilde{\varphi}(j, k - 1) - \tilde{\varphi}(j, k + 1)).
\]

At the boundaries, these equations have to be modified such that they are in harmony with the boundary conditions for the potential. For more details, see Sec. 2.11.

2.3 Interpolation scheme

Since in PIC macro-quantities are calculated on the grid but particles move inside the cells, it is often necessary to interpolate a macro-quantity from the grid points to a particle position. As an example, we consider here the interpolation of the electric field, however, for consistency in the code, always the same scheme has been used whenever interpolation was required for whatever physical quantity (e.g. for the potential).

Let a particle number \(n\) have the position \((\text{pa}[n].p.r, \text{pa}[n].p.z)\). Define the integer and fractional parts of this position as (see Fig. 2.1):

\[
j = \text{int} (\text{pa}[n].p.r)
\]

\[
k = \text{int} (\text{pa}[n].p.z)
\]

\[
h_r = \frac{\text{frac} (\text{pa}[n].p.r)}{}
\]

\[
h_z = \frac{\text{frac} (\text{pa}[n].p.z)}{}
\]

The \(i\) component of the electric field at the particle position is then given by the linear interpolation of the four surrounding grid point contributions:

\[
\tilde{E}_i(r, z) = (1 - h_r) \cdot (1 - h_z) \cdot \tilde{E}_i(j, k) + (1 - h_r) \cdot h_z \cdot \tilde{E}_i(j, k + 1) + h_r \cdot (1 - h_z) \cdot \tilde{E}_i(j + 1, k) + h_r \cdot h_z \cdot \tilde{E}_i(j + 1, k + 1). \tag{2.46}
\]

\[
+ hr \cdot 1 \cdot \tilde{E}_i(j + 1, k) + hr \cdot 1 \cdot \tilde{E}_i(j + 1, k + 1).
\]

Figure 2.1: Interpolation from grid points to particle position schematically.
2.4 Particle pusher equations

2.4.1 Electrostatic case

To push a particle, the electric field is first interpolated as above to the particle position. The particle is then accelerated due to the electric part of the Lorentz force,

\[ \ddot{v}_r = 2\dot{E}_r, \]
\[ \ddot{v}_z = 2\dot{E}_z, \]

and particles are moved in the direction of the velocity they have \((\Delta r_i = v_i \Delta t)\),

\[ \ddot{\bar{z}}' = \bar{z} + \Delta \bar{z} = \bar{z} + \ddot{v}_z, \]
\[ \ddot{\bar{r}}' = \sqrt{(\bar{r} + \Delta \bar{r}_x)^2 + \Delta \bar{r}_y^2} = \sqrt{(\bar{r} + \ddot{v}_r)^2 + \ddot{v}_t^2}. \]

where \(z'\) and \(r'\) stand for the updated quantities. The change in \(r\)-coordinate is illustrated in Fig. 2.2(a).

![Figure 2.2](image)

Figure 2.2: Illustration of the particle pusher, where particles are treated in a Lagrangian frame. The coordinates change in the direction of the velocities, that are calculated from the electric field. The new reference frame is given by the updated coordinates \((z', r')\), into which velocities that are calculated in the old frame have to be transformed.

Finally, since the code is only two-dimensional, it means that we are simulating in \(\vartheta\)-direction an infinitely thin ‘slice’ of a cylinder. Therefore, after pushing the particles we must rotate them back into this slice (to a given constant \(\vartheta\), chosen initially, for instance \(\vartheta = 0\), see Fig. 2.2(b)). Defining now \(\sin \alpha = \frac{\Delta r_y}{r'}\) and \(\cos \alpha = \frac{\bar{r} + \Delta \bar{r}_x}{r'}\) we may perform the rotation by transforming velocities to

\[ \ddot{v}_r' = \cos \alpha \cdot \ddot{v}_r + \sin \alpha \cdot \ddot{v}_t \]
\[ \ddot{v}_t' = -\sin \alpha \cdot \ddot{v}_r + \cos \alpha \cdot \ddot{v}_t. \]
2.4.2 Magnetic case

In the code, it is possible to apply an external magnetic field along the z- and/or r-direction \( \mathbf{B}_{ext} = (B_z, B_r, B_t = 0) \) in the right-handed coordinate system \((z, r, t)\). The non-zero components of \( \mathbf{v} \times \mathbf{B} \) are

\[
\mathbf{v} \times \mathbf{B} = \begin{pmatrix} v_z \\ v_r \\ v_t \end{pmatrix} \times \begin{pmatrix} B_z \\ B_r \\ 0 \end{pmatrix} = \begin{pmatrix} -v_t B_r \\ v_t B_z \\ v_z B_r - v_r B_z \end{pmatrix}
\] (2.54)

Denoting the ‘old’ velocity vector from the previous time step as \( \mathbf{v}_0 = (v^0_z, v^0_r, v^0_t) \) and the updated velocity vector from the next time step as \( \mathbf{v}' = (v'_z, v'_r, v'_t) \), we can update the dimensional velocity vector according to the Boris method [5] in the following four steps (treating first electrons):

1. Electric push (I)

\[
v^a_z = v^0_z - \frac{e\Delta t}{2m_e} E_z
\] (2.55)
\[
v^a_r = v^0_r - \frac{e\Delta t}{2m_e} E_r
\] (2.56)

2. Magnetic push (I)

\[
v^b_z = v^a_z + \frac{e\Delta t}{2m_e} v^0_t B_r
\] (2.57)
\[
v^b_r = v^a_r - \frac{e\Delta t}{2m_e} v^0_t B_z
\] (2.58)
\[
v^b_t = v^0_t - \frac{e\Delta t}{2m_e} \left\{ v^a_z B_r - v^a_r B_z \right\}
\] (2.59)

3. Magnetic push (II)

\[
v^c_z = v^a_z + \frac{e\Delta t}{m_e} \frac{v^0_t B_r}{1 + \left( \frac{e\Delta t}{2m_e} \right)^2 \left\{ B_z^2 + B_r^2 \right\}}
\] (2.60)
\[
v^c_r = v^a_r - \frac{e\Delta t}{m_e} \frac{v^0_t B_z}{1 + \left( \frac{e\Delta t}{2m_e} \right)^2 \left\{ B_z^2 + B_r^2 \right\}}
\] (2.61)
\[
v^c_t = v^0_t - \frac{e\Delta t}{m_e} \frac{v^b_r B_r - v^b_t B_z}{1 + \left( \frac{e\Delta t}{2m_e} \right)^2 \left\{ B_z^2 + B_r^2 \right\}}
\] (2.62)

4. Electric push (II)

\[
v'_z = v^c_z - \frac{e\Delta t}{2m_e} E_z
\] (2.63)
\[
v'_r = v^c_r - \frac{e\Delta t}{2m_e} E_r
\] (2.64)
Once the velocities are updated, particle positions are updated and velocities rotated in the same way as in the electrostatic case, see Eqs. 2.50, 2.51, 2.52, and 2.53.

To obtain the dimensionless equations, we rescale $\tilde{E} = \frac{e}{2m_e} \frac{\Delta t}{\Delta z^2} E$ as before, and analogously $\tilde{v}\tilde{B} = \frac{e}{2m_e} \frac{\Delta t}{\Delta z^2} vB$. Since $\tilde{v} = \frac{\Delta t}{\Delta z} v$, the dimensionless magnetic field has to be defined as

$$\tilde{B} = \frac{e\Delta t}{2m_e} B$$

The *dimensionless* velocity update equations are then:

1. Electric push (I)

   $$\tilde{v}^a_z = \tilde{v}_z^0 - \tilde{E}_z \quad (2.65)$$
   $$\tilde{v}^a_r = \tilde{v}_r^0 - \tilde{E}_r \quad (2.66)$$

2. Magnetic push (I)

   $$\tilde{v}^b_z = \tilde{v}_z^a + \tilde{v}_r^0 \tilde{B}_r \quad (2.67)$$
   $$\tilde{v}^b_r = \tilde{v}_r^a - \tilde{v}_t^0 \tilde{B}_z \quad (2.68)$$
   $$\tilde{v}^b_t = \tilde{v}_t^0 - \{\tilde{v}_z^a \tilde{B}_r - \tilde{v}_r^a \tilde{B}_z\} \quad (2.69)$$

3. Magnetic push (II)

   $$\tilde{v}^c_z = \tilde{v}_z^a + \frac{2\tilde{v}_r^b \tilde{B}_r}{1 + \{B_z^2 + B_r^2\}} \quad (2.70)$$
   $$\tilde{v}^c_r = \tilde{v}_r^a - \frac{2\tilde{v}_t^b \tilde{B}_z}{1 + \{B_z^2 + B_r^2\}} \quad (2.71)$$
   $$\tilde{v}^c_t = \tilde{v}_t^0 - \frac{2\tilde{v}_r^b \tilde{B}_r - 2\tilde{v}_t^b \tilde{B}_z}{1 + \{B_z^2 + B_r^2\}} \quad (2.72)$$

4. Electric push (II)

   $$\tilde{v}_t^l = \tilde{v}_t^c - \tilde{E}_z \quad (2.73)$$
   $$\tilde{v}_r^l = \tilde{v}_r^c - \tilde{E}_r \quad (2.74)$$

To apply the same equations also for ions, replace $\tilde{E}$ by $\tilde{E}_{\text{ion}}$ and multiply $\tilde{B}$ by the factor $-\frac{m_e}{M_{\text{ion}}}$ (one factor of $dt_{\text{ion}}$ is contained already in the ion velocity).
2.5. Generating a uniform density in 2D PIC

Over the whole domain \((r \in [0, R_{\text{max}}], \ z \in [0, Z_{\text{max}}])\), we would like to distribute
\[
N_{\text{REAL}} = \int dV \ n_{\text{ref}} = \pi R_{\text{max}}^2 Z_{\text{max}} n_{\text{ref}} \tag{2.75}
\]
real particles, corresponding to
\[
N_{\text{SIM}} = \frac{N_{\text{REAL}}}{N_{\text{SP}}} = \pi R_{\text{max}}^2 Z_{\text{max}} \frac{n_{\text{ref}}}{N_{\text{SP}}} \tag{2.76}
\]
\[
= \pi \frac{(nr)^2}{\lambda_{Db}^3} \Delta z \frac{nz}{\Delta z} N_{Db} = \pi \Delta z \frac{nr^2}{\Delta z} n_{\text{ref}} N_{Db} \tag{2.77}
\]
simulation particles distributed in a uniform density, where \(N_{Db}\) is defined as the number of superparticles in a Debye cube corresponding to the density \(n_{\text{ref}}\). To obtain a uniform density, the particle distribution has to have the same dependency on the coordinates as the volume element:
\[
f(r, z) = c \cdot r \tag{2.78}
\]
where \(c\) is a constant that is determined through the normalisation requirement:
\[
\int_0^{nz} dz \int_0^{nr} dr \ c \cdot r = \frac{c}{2} nr^2 nz = \pi \Delta z \frac{nr^2}{\Delta z} n_{\text{ref}} N_{Db} \tag{2.79}
\]
\[
\therefore \ c = 2\pi \Delta z \frac{nr^2}{\Delta z} N_{Db}. \tag{2.80}
\]
The probability distribution function of one simulation particle is then
\[
p(r, z) = \frac{2}{nr^2 nz} r = \left( \frac{2}{nr^2} \right) \left( \frac{1}{nz} \right) \equiv p_f(r)p_g(z) \tag{2.81}
\]
To generate particles in such a distribution, we calculate the cumulative distribution functions
\[
[0, 1] \ni F(r) = \int_0^{r'} \frac{2}{nr^2} r' \rightarrow F^{-1}(\text{RAND}) = nr \cdot \sqrt{\text{RAND}} \tag{2.82}
\]
\[
[0, 1] \ni G(z) = \int_0^{z'} \frac{1}{nz} dz' = \frac{z}{nz} \rightarrow G^{-1}(\text{RAND}) = nz \cdot \text{RAND} \tag{2.83}
\]
In conclusion, we need to generate \(\pi \Delta z ^3 nr^2 nz N_{Db}\) particles with following positions:
\[
\therefore \ \hat{z} = \text{RAND} \cdot nz \tag{2.84}
\]
\[
\hat{r} = \sqrt{\text{RAND}} \cdot nr \tag{2.85}
\]
2.6 Particle injection

2.6.1 Generating velocities

Particles injected from the wall should be directed away from the wall, otherwise in a Gaussian distribution around an injection temperature \( v_{th} \) (different for different species). For instance, for an injection in \(+z\)-direction the velocity distribution is

\[
f(v_z, v_r, v_t) \sim v_z e^{-\frac{v_z^2 + v_r^2 + v_t^2}{2v_{th}^2}},
\]

(2.86)

or, in terms of rescaled variables (N.B. these are not the same as the dimensionless variables used in the code!):

\[
f(z \equiv \frac{v_z}{v_{th}}, r \equiv \frac{v_r}{v_{th}}, t \equiv \frac{v_t}{v_{th}}) \sim ze^{-\frac{z^2 + r^2 + t^2}{2}}.
\]

(2.87)

From the normalisation criterion \( \int d\vec{x} f(\vec{x}) = 1 \) and using \( \int_{-\infty}^{\infty} e^{-r^2/2} dr = \sqrt{2\pi} \) as well as \( \int_{0}^{\infty} z e^{-z^2/2} dz = 1 \) we obtain:

\[
f(z, r, t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{r^2}{2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}
\]

(2.88)

For the \((r,t)\)-part we use a transformation to polar coordinates \((R, \phi)\):

\[
f_r f_t dr dt = \frac{1}{2\pi} e^{-\frac{R^2}{2}} R R dR d\phi = f_R f_{\phi}.
\]

(2.89)

Using now the cumulative distribution functions

\[
F_{\phi} = \int_{0}^{\phi} \frac{1}{2\pi} d\phi \in [0, 1] \text{ for } \phi \in [0, 2\pi)
\]

(2.90)

\[
F_R = \int_{0}^{R} Re^{-\frac{R^2}{2}} dR = 1 - e^{-\frac{R^2}{2}} \in [0, 1] \text{ for } R \in [0, \infty)
\]

(2.91)

we can generate

\[
\phi = 2\pi \text{RAND}_1
\]

(2.92)

\[
r = \sqrt{-2 \ln(\text{RAND}_2)}
\]

(2.93)

where \( \text{RAND}_i \) are random numbers between \((0,1)\). The remaining \( f_z \) distribution has just the same form as \( f_R \), so

\[
z = \sqrt{-2 \ln(\text{RAND}_3)}
\]

(2.94)

In terms of the original velocity components, we end up with

\[
v_z = \sqrt{-2 \ln(\text{RAND}_3)} v_{th} > 0
\]

(2.95)

\[
v_r = \sqrt{-2 \ln(\text{RAND}_1)} \cos(2\pi \text{RAND}_2) v_{th}
\]

(2.96)

\[
v_t = \sqrt{-2 \ln(\text{RAND}_1)} \sin(2\pi \text{RAND}_2) v_{th}
\]

(2.97)
2.6.2 Fowler-Nordheim field emission

In ‘discharge’ mode, that is, choosing boundary conditions \( \text{PBC} = 0 \) (see Sec. 2.12) and \( \text{PIS} = 0 \) (see Sec. 2.13), electrons are injected according to the Fowler-Nordheim formula [13], with the Wang and Loew approximation [14]:

\[
 j_{\text{FN}} = 1.5414 \cdot 10^{-6} E_{\text{LOC}}^2 \phi \cdot t^2(s) \exp\left( \frac{6.8309 \cdot 10^9 \phi^{3/2} v(s)}{E_{\text{LOC}}} \right),
\]

(2.98)

where the field emission current density is measured in \( [j_{\text{FN}}] = \text{A/m}^2 \), the local surface field in \( [E_{\text{LOC}}] = \text{V/m} \), and the work function in \( [\phi] = \text{eV} \); the elliptical functions are approximated with \( t(s) = 1 \) and \( v(s) = 0.956 - 1.062 \cdot s^2 \), where \( s = 3.7947 \cdot 10^{-5} \sqrt{\frac{E_{\text{LOC}}}{\phi}} \).

Inserting the value \( \phi = 4.5 \text{ eV} \), we arrive at the form used in the code:

\[
 j_{\text{FN}} = 4.7133 \cdot 10^9 E_{\text{LOC}}^2 \cdot \exp\left( -\frac{62.338}{E_{\text{LOC}}} \right),
\]

(2.99)

here \( [j_{\text{FN}}] = \text{A/cm}^2 \) and \( [E_{\text{LOC}}] = \text{GV/m} \).

2.7 The discharge gap as part of an electric circuit

In ‘discharge’ mode, it is also desirable to know what is the current-voltage characteristic and the energy consumption of the discharge gap. To determine these, a simple RC-circuit has been assumed (Fig. 2.3).

![Figure 2.3: The discharge gap as part of an RC-circuit. The energy available for the discharge is stored in the external capacitor.](Image)

Initially, the capacitor \( C_{\text{ext}} \) is assumed to be fully charged with the initial voltage \( U_{Nz} - U_0 \) given by the user in input.txt, \( Q_0 = (U_{Nz} - U_0)C_{\text{ext}} \). The capacitor charge \( Q(t) \) and the voltage over the discharge gap \( U(t) \) are then updated every time step as follows:

\[
 Q(t_{i+1}) = Q(t_i) - I(t_i) \Delta t, \quad \text{(2.100)}
\]

\[
 U(t_{i+1}) = \frac{Q(t_{i+1})}{C_{\text{ext}}} - R_{\text{ext}} I(t_i), \quad \text{(2.101)}
\]

with the current \( I(t_i) \) preferably calculated at the cathode (side losses will be more significant in the anode current).
We shall now rescale the above equations to

\[ \bar{Q}(t_{i+1}) = \bar{Q}(t_i) - \bar{I}(t_i), \quad (2.102) \]

\[ \bar{U}(t_{i+1}) = \frac{\bar{Q}(t_{i+1})}{C_{ext}} - \bar{R}_{ext} \bar{I}(t_i), \quad (2.103) \]

where \( \bar{U} \) is the dimensionless potential defined in Eq. 2.20. Defining the dimensionless current as the sum of superparticles crossing the boundary during a time step (summed up with the proper sign depending on species and direction), \( \bar{I} \equiv \sum_{\text{species}} (\text{no. of particles}) \), the total current is rescaled by the factor

\[ \frac{\bar{I}}{I[A]} = \frac{\Delta t}{eN_{SP}} = \frac{1}{\varepsilon_0} \sqrt{\frac{m_e}{e}} \frac{\Delta t \cdot N_{Db}}{(T_{ref}/e)^{3/2}} = 2.693026 \cdot 10^5 \frac{\Delta t \cdot N_{Db}}{(T_{ref}/e)^{3/2}}, \quad (2.104) \]

where the reference temperature is given in units of \([T_{ref}/e] = V\). Charge, capacitance, and resistance have to scale then, with the reference density given in units of \([n_{ref}] = 1/cm^3\), as follows:

\[ \frac{\bar{Q}}{Q[C]} = \frac{\bar{I}}{I} = \frac{1}{eN_{SP}} = \frac{\sqrt{e}}{\varepsilon_0^{3/2}} \frac{N_{Db} \cdot \sqrt{n_{ref}}}{(T_{ref}/e)^{3/2}}, \quad (2.105) \]

\[ \frac{\bar{C}}{C[F]} = \frac{\bar{Q}U}{Q U} = \frac{1}{eN_{SP}} \left( \frac{\Delta z}{\Delta t} \right)^2 \frac{T_{ref}}{e} = \frac{\sqrt{e}}{\varepsilon_0^{3/2}} \left( \frac{\Delta z}{\Delta t} \right)^2 \frac{N_{Db} \cdot \sqrt{n_{ref}}}{\sqrt{T_{ref}/e}}, \quad (2.106) \]

\[ \frac{\bar{R}}{R[\Omega]} = \frac{\bar{U} I}{U I} = \frac{1}{eN_{SP}} \left( \frac{T_{ref}/e}{\Delta t} \right)^2 = \frac{\varepsilon_0}{m_e} \frac{\Delta t}{\Delta z^2} \frac{\sqrt{T_{ref}/e}}{N_{Db}}, \quad (2.107) \]
2.8 Density calculation

As it was mentioned in the introduction (Sec. 1.1), in order to ensure self-consistency, extrapolation of quantities from particle position to grid points has to be carried out in analogy with the interpolation from grid points to the particle position.

Therefore we define $h_r = \frac{p[n].p.r}{V_{\text{cell}(j)}}$ and $h_z = \frac{p[n].p.z}{V_{\text{cell}(j)}}$ as we did in Sec. 2.3. Then one particle contributes the following density to the four closest grid points (see Fig. 2.4):

\[
dens(j, k) = (1 - h_r)(1 - h_z)
\]
\[
dens(j, k + 1) = (1 - h_r)h_z
\]
\[
dens(j + 1, k) = h_r(1 - h_z)
\]
\[
dens(j + 1, k + 1) = h_rh_z
\]

In the end, according to Eq. (2.31) this is to be multiplied by the species’ charge:

\[
dens(j, k)* = q_p
\]


2.9 Monitoring energy conservation

At any instant, the total energy in the system can be followed via \texttt{energy\_total\_2D()} in \texttt{engy.cpp}. The total energy is the sum of kinetic and potential energy,

\[
W = T + V = \sum_{i,n} m_i v_{i,n}^2 + \sum_{i,n} q_i \varphi_{i,n}(\vec{r}_{i,n}),
\]

(2.113)

where we sum over species \((i)\) and number of particles within that species \((n)\). In the following, we will split the total energy into particle \((e, n, i)\) and electrostatic potential energy \((p)\) contributions:

\[
W_{tot} = W_e + W_n + W_i + W_p.
\]

(2.114)

By following the values of \(W_{tot}\), energy conservation may be monitored directly in non-collisional systems, if boundary conditions are appropriate (e.g. reflection is energy conserving; for other boundary conditions, the energy lost at the walls has to be calculated). In systems with external particle and/or energy sources, \(W_{tot}\) serves as an indicator for reaching steady-state, for instance; while changes in the ratio of kinetic and potential energy can indicate e.g. the build-up of a sheath.

Note that since every simulation particle represents \(N_{SP}\) real particles, the kinetic energy contributions will scale as \(\sim m_i \sim N_{SP}\), whereas the electrostatic potential energy scales \(\sim q_i q_j \sim N_{SP}^2\), and the electric field energy density scales as \(\sim 1\). To compare these quantities, we have to make sure that they all scale in the same way with respect to \(N_{SP}\). In the following, \(W_\ast\) \((\ast = e, n, i, p)\) stands for the physical energy, a sum of energies over real particles.

1. Electrons (all velocities dimensionless)

\[
\frac{W_e}{N_{SP}T_{ref}} = \frac{\vec{v}_e^2}{2v_{te}^2} = \frac{pa[] \cdot p \cdot vz^2 + pa[] \cdot p \cdot vr^2 + pa[] \cdot p \cdot vt^2}{2v_{te}^2}
\]

(2.115)

2. Neutrals (all velocities dimensionless)

\[
\frac{W_n}{N_{SP}T_{ref}} = \frac{\vec{v}_n^2}{2c_{s,Cu}^2} = \frac{pa[] \cdot p \cdot vz^2 + pa[] \cdot p \cdot vr^2 + pa[] \cdot p \cdot vt^2}{2\frac{m_e}{M_{Cu}}v_{te}^2}
\]

(2.116)

3. Ions (all velocities dimensionless)

\[
\frac{W_i}{N_{SP}T_{ref}} = \frac{\vec{v}_i^2}{2c_{s,Cu}^2} = \frac{pa[] \cdot p \cdot vz^2 + pa[] \cdot p \cdot vr^2 + pa[] \cdot p \cdot vt^2}{2\frac{m_e}{M_{Cu}^+}v_{te}^2}
\]

(2.117)

4. The electrostatic potential energy of a super-point-charge is \(V(\vec{r}) = q\varphi(\vec{r})\):

\[
\frac{W_p}{N_{SP}T_{ref}} = \frac{q\varphi(\vec{r})}{T_{ref}} = \frac{q}{\Delta z^2} \sum_{\pm 1} \frac{\Delta z^2}{\Delta t^2} \varphi(\vec{r}),
\]

(2.118)

where the sign is + for ions and − for electrons, and the potential is interpolated from the grid to the position of the charge, since that will give us the maximum of the potential energy.
5. Alternatively, the energy contained in the electric field \( W_f \) may be calculated, too. However, the electric field is a secondary quantity, since it is derived from the potential, therefore, the error will be bigger in this case. To follow the same scaling as above, this term needs to be rescaled also to \( N_{SP} \)

\[
\frac{W_f}{N_{SP}T_{ref}} = \frac{1}{2} \varepsilon_0 \sum_{j,k} (\vec{E}(j,k)^2) \Delta V(j),
\]

where \( E(j,k) \) is the electric field on the grid and \( V(j) \) is the corresponding volume of the cell, calculated on the grid. In terms of dimensionless variables:

\[
\frac{W_f}{N_{SP}T_{ref}} = \frac{1}{2} \varepsilon_0 \Delta z^3 \sum_{i,j,k} (\vec{E}(j,k)^2 \tilde{\nabla}_{cell}(j))
\]

\[
= \frac{1}{2} \varepsilon_0 \Delta z^3 \sum_{i,j,k} \left( \frac{2m_e}{e^2} \frac{\Delta z}{\Delta t} \lambda_{Db} \omega_{pe}^2 \tilde{E}_i^2(j,k) \right) \tilde{\nabla}_{cell}(j)
\]

\[
= \frac{1}{2} N_{Db} \frac{\varepsilon_0}{n_{ref} T_{ref}} \frac{4m_e^2}{e^2} \frac{\Delta z^5}{\Delta t} \lambda_{Db} \omega_{pe}^2 \omega_{pe} \sum_{i,j,k} (\tilde{E}_i^2(j,k) \tilde{\nabla}_{cell}(j))
\]

\[
= \frac{1}{2} N_{Db} \frac{\varepsilon_0}{n_{ref} T_{ref}} \frac{4m_e^2}{e^2} \frac{\Delta z^5}{\Delta t^2} \frac{\varepsilon_0 n_{ref}}{m_e} \sum_{i,j,k} (\tilde{E}_i^2(j,k) \tilde{\nabla}_{cell}(j))
\]

\[
= \frac{4N_{Db} \Delta z^5}{2 \Delta t} \sum_{i,j,k} (\tilde{E}_i^2(j,k) \tilde{\nabla}_{cell}(j)).
\]

Given that the dimensionless thermal velocity \( \tilde{v}_{te} = \Delta t/\Delta z \), we finally obtain

\[
\frac{W_f}{N_{SP}T_{ref}} = \frac{4N_{Db} \Delta z^3}{2 \tilde{v}_{te}} \sum_{i,j,k} (\tilde{E}_i^2(j,k) \tilde{\nabla}_{cell}(j)) = \frac{4}{2 \tilde{q} \tilde{v}_{te}} \sum_{i,j,k} (\tilde{E}_i^2 \tilde{V}).
\]

Note once more that energy conservation can only be demanded, if there are no sinks, no sources, no collisions, no externally pumped-in energy and so forth.
2.10 Outputting using \texttt{moms.cpp} and \texttt{outputz.cpp}

All quantities in the output files are given in reference quantities \((T_e, n_{ref}, \text{etc.})\). Therefore prior to outputting, variables used in the code have to be properly rescaled, which we discuss below.

**Outputting currents** Total currents originating from charged particles injected from or absorbed at the boundaries can be obtained from superparticle counts as described in Sec. 2.7. Alternatively, current density can be integrated over the area of impact, \(I = \int dA \cdot j\). This integral can be rescaled to dimensionless variables similarly as current density in Eq. 2.20 is rescaled with the factor \(j\):

\[
I = \sum_{r_j} (j_{\text{removed}}(r_j) \Delta A_{e11}(r_j)) = j \lambda_{Db}^2 \sum_{r_j} \left(N_{p,\text{removed}} \frac{\Delta A(r_j)}{\lambda_{Db}^2}\right) \tag{2.126}
\]

\[
\equiv \sum_{r_j} \left(N_{p,\text{removed}}(2j + 1)\right), \tag{2.127}
\]

where \(N_{p,\text{removed}}\) is the number of particles removed (or injected) at the boundary and in the last equation \(j\) stands for the grid number along \(r\). Since the area of a cell is

\[
\Delta A_{e11}(r_j) = \int_{r_j}^{r_{j+1}} 2\pi r = \pi (2j + 1) \Delta z \lambda_{Db}^2, \tag{2.128}
\]

we can express the rescaling factor as:

\[
\mathcal{J} = \pi \Delta z^2 e \frac{N_{SP} \omega_{pe}}{p2inj_{\text{step}}}, \tag{2.129}
\]

where \(p2inj_{\text{step}}\) is given in \(\omega_{pe}^{-1}\)s. Eliminating \(N_{SP}\) and \(\omega_{pe}\), and taking into account that \(T_e\) is measured in \(eV\), we obtain the following final form for the rescaling factor:

\[
\mathcal{J} = \pi e \varepsilon_0 \sqrt{\frac{\Delta z^2 T_{ref}^{3/2}}{m_e N_{Db} p2inj_{\text{step}}}} = 1.1665658 \cdot 10^{-5} \frac{\Delta z^2 T_{ref}^{3/2}}{N_{Db} p2inj_{\text{step}}}. \tag{2.130}
\]

**Outputting density** We sum up the density on each grid point over \(n_{\text{aver}}\) steps, and store it in the variable \(\text{dens}_\text{av}[j*\text{NR}+k]\). The rescaled density in units of the reference density is then:

\[
\frac{n_e^{\text{REAL}}(r,z)}{n_{ref}} = \text{sign} \frac{\text{dens}_\text{av}[j*\text{NR}+k]}{n_{\text{aver}} \Delta t^2} \tag{2.131}
\]

where \text{sign} is +1 for ions and −1 for electrons.

**Outputting potential** Similarly here we use directly the potential calculated in \texttt{phi.cpp}, it is just averaged over \(n_{\text{aver}}\) steps. Therefore Eq. 2.20 still applies:

\[
\frac{\varphi^{\text{REAL}}[V]}{T_{ref}[eV]} = \frac{\Delta z^2}{n_{\text{aver}} \Delta t^2} \varphi_{av} \tag{2.132}
\]
Outputting velocity In \texttt{moms.cpp} in the function \texttt{aver\_moments\_2D()}, velocities are summed over (i) averaging time steps and (ii) all particles in the given cell: 
\[
\text{mom[]\_\_u} + = \text{pa[]\_\_p\_v}.
\]
This is then rescaled to units of \(c_{s,Cu^+}\) in \texttt{outputz.cpp}, \texttt{out\_velz\_2D()} by calculating 
\[
\text{mom[]\_\_u} / \left( u_0 \text{mom[\_\_n]} \right),
\]
where (i) for electrons
\[
u_0 = c_s \sqrt{M\_ions[0] / M\_ions[1]} = c_{s,Cu^+},
\]
and (ii) for ions
\[
u_0 = c_s \, dt\_ion \sqrt{M\_ions[0] / M\_ions[1]} = \sqrt{T_{ref} / m_H^+ \, dt\_ion} \sqrt{M\_ions[0] / M\_ions[1]} = c_{s,Cu^+} \, dt\_ion.
\]
Finally we get exactly the average velocity in units of \(c_{s,Cu^+}\). For electrons, for instance (and similarly for ions)
\[
\frac{<u>}{c_{s,Cu^+}} = \frac{\text{mom[]\_\_u}}{u_0 \text{mom[\_\_n]}} = \frac{\sum_i \sum_n v_i}{c_{s,Cu^+} \sum_i \sum_n 1}.
\]
Here the sums go over averaging time steps (\(\sum_i\)) and particles in the cell (\(\sum_n\)).

Outputting temperature Correspondingly, in \texttt{aver\_moments\_2D()}, temperatures are summed over (i) averaging time steps and (ii) all particles in the given cell: 
\[
\text{mom[]\_\_t} + = \text{pa[]\_\_p\_v}^2.
\]
This is then rescaled to units of \(T_e\) in \texttt{outputz.cpp}, \texttt{out\_tempz\_2D()}.
Let’s assume a Maxwell-Boltzmann distribution
\[
\sim e^{-m(v-v_0)^2 / 2T^2}
\]
in each of the directions separately, where, in general, there might be some stream velocity \(v_0\) in that direction. Then the temperature is nothing else but the standard deviation squared of this distribution:
\[
T = \sigma^2 = m(<v^2> - <v>^2) = m \left( \frac{\sum_{i=1}^N v_i^2}{N} - \left[ \frac{\sum_{i=1}^N v_i}{N} \right]^2 \right)
\]
To output temperature in units of \(T_e\), we have to calculate:
(i) for electrons:
\[
\frac{T}{T_{ref}} = \frac{v_e^2}{v_{te}^2} = \frac{\sum_i \sum_n v_i^2}{\sum_i \sum_n 1} - \left[ \frac{\sum_i \sum_n v_i}{\sum_i \sum_n 1} \right]^2
\]
so, in principle,
\[
\frac{T}{T_{ref}} = \frac{\text{mom\_el[]\_\_t}}{\text{mom\_el[]\_\_n}} - \left( \frac{\text{mom\_el[]\_\_u}}{\text{mom\_el[]\_\_n}} \right)^2
\]
However, if \texttt{out\_velz\_2D()} has been called already before, the second term is already given in units of \(c_{s,Cu^+}\) and we get

\[
\frac{T}{T_{\text{ref}}} = \left( \frac{\text{mom\_el[]}\cdot t}{u_0 \cdot \text{mom\_el[]}\cdot n} - (\text{mom\_el[]}\cdot u)^2 \right) \cdot f_{\text{norm}}, \tag{2.140}
\]

with \(u_0 = c_{s,Cu^+}\) and \(f_{\text{norm}} = \frac{m_e}{m_{Cu^+}}\).

(ii) for ions:

\[
\frac{T}{T_{\text{ref}}} = \frac{m_{Cu^+}}{m_e} \frac{v_{\text{ion}}^2}{(dt_{\text{ion}})^2 v_{Te}^2} = \frac{v_{\text{ion}}^2}{(dt_{\text{ion}})^2 c_{s,Cu^+}^2}, \tag{2.141}
\]

which implies

\[
\frac{T}{T_{\text{ref}}} = \frac{\text{mom\_ion[]}\cdot t}{\text{mom\_ion[]}\cdot n} - \left( \frac{\text{mom\_ion[]}\cdot u}{\text{mom\_ion[]}\cdot n} \right)^2 \frac{(dt_{\text{ion}})^2 c_{s,Cu^+}^2}{(dt_{\text{ion}})^2 c_{s,Cu^+}^2}. \tag{2.142}
\]

However, in combination with \texttt{out\_velz\_2D()} we have to use the following expression in \texttt{out\_tempz\_2D()}:

\[
\frac{T}{T_{\text{ref}}} = \left( \frac{\text{mom\_ion[]}\cdot t}{u_0 \cdot \text{mom\_ion[]}\cdot n} - (\text{mom\_ion[]}\cdot u)^2 \right) \cdot f_{\text{norm}}, \tag{2.143}
\]

with \(u_0 = c_{s,Cu^+} \cdot dt_{\text{ion}}\) and \(f_{\text{norm}} = 1\).

**Outputting velocity distribution** Velocity distribution is outputted with a spatial resolution of \(2 \times 2\) cells and a resolution of \(N_{\text{vdst}}\) equal parts in velocity space, centred around zero velocity. A given range of velocity can be sampled like this: For electrons, \(|v| \leq \frac{N_{\text{vdst}}}{\text{const}} v_{Te}\), where we chose \(\text{const} = 200/3\) and typically \(N_{\text{vdst}} = 401\), resulting in a resolution of \(|v| \leq 3v_{Te}\). For ions, \(v_{Te}\) is replaced by \(c_{s,i}\).
2.11 Potential and electric field boundary conditions

In this section, we shall consequently omit the tildes, but we still mean dimensionless quantities.

The switch between different boundary condition options for the field solver is the variable ‘BC’ in \texttt{h/dim.h}. To which grid points the boundary conditions are applied is illustrated in Fig. 2.5. The options are the following:

**BC 0** for $\varphi|_{r=nr}=0$ (rather unphysical)

- **Cathode**: $k=0$, $j \in [0, nr]$
  \[
  \begin{align*}
  \varphi &= \varphi_C \\
  E_r(j, 0) &= 0 \\
  E_z(j, 0) &= \frac{3}{4}\varphi(j, 0) - \varphi(j, 1) + \frac{1}{4}\varphi(j, 2)
  \end{align*}
  \]  \hspace{1cm} (2.144)

- **Anode**: $k=nz$, $j \in [0, nr]$
  \[
  \begin{align*}
  \varphi &= \varphi_A \\
  E_r(j, nz) &= 0 \\
  E_z(j, nz) &= -\frac{3}{4}\varphi(j, nz) + \varphi(j, nz - 1) - \frac{1}{4}\varphi(j, nz - 2)
  \end{align*}
  \]  \hspace{1cm} (2.147)

- **Symmetry axis**: $j = 0$, $k \in (0, nz)$
  \[
  \begin{align*}
  \varphi(0, k-1) - 6\varphi(0, k) + \varphi(0, k+1) + 4\varphi(1, k) &= -n_e(0, k) - n_i(0, k) \\
  \frac{\partial \varphi}{\partial r} \bigg|_{r=0} &= E_r(0, 0) = 0 \text{ (symmetry)} \\
  E_z(0, k) &= \frac{1}{4}(\varphi(0, k-1) - \varphi(0, k+1))
  \end{align*}
  \]  \hspace{1cm} (2.148)

- **‘Infinity’**: $j = nr$, $k \in (0, nz)$
  Note that in this special case $E_z(nr, k) = 0 \forall k$, so it should be made sure the boundary conditions altogether make sense (for instance, this BC can not be applied for a vacuum boundary in $j = nr$, since in case of the vacuum solution for $\varphi_C \neq \varphi_A$ we would get $E_z(nr, k) = \text{const}$). This is just a special case of a fixed potential at the wall $j = nr$, and therefore we can not require either that far away from the centre $\frac{\partial \varphi}{\partial r} \bigg|_{r=\infty} = 0$.

  \[
  \begin{align*}
  \varphi(nr, k) &= 0 \\
  E_r(nr, k) &= -\frac{3}{4}\varphi(nr, k) + \varphi(nr - 1, k) - \frac{1}{4}\varphi(nr - 2, k) \\
  E_z(nr, k) &= \frac{1}{4}(\varphi(nr, k - 1) - \varphi(nr, k + 1)) = 0
  \end{align*}
  \]  \hspace{1cm} (2.149)
Figure 2.5: Grid points handled with different boundary conditions. Red markers in Fig. b) correspond to BC 3.
2.11. POTENTIAL AND ELECTRIC FIELD BOUNDARY CONDITIONS

BC 1 for $\frac{\partial \varphi}{\partial r} \bigg|_{r=nr} = 0$ (suitable for breakdown simulation)

Here we apply exactly the same BCs as for BC 0, save for $j = nr$:

- ‘Infinity’: $j = nr$, $k \in (0, nz)$
  \[
  \left. \frac{\partial \varphi}{\partial r} \right|_{r=nr} = -\frac{3}{4} \varphi(nr,k) + \varphi(nr-1,k) - \frac{1}{4} \varphi(nr-2,k) = 0
  \]
  \[
  E_r(nr,k) = 0
  \]
  \[
  E_z(nr,k) = \frac{1}{4} (\varphi(nr,k-1) - \varphi(nr,k+1))
  \]

BC 2 for $\varphi = \text{const}$ at $r$-boundaries, $\partial \varphi / \partial z = 0$ at $z$-boundaries

- Cathode: $k = 0$, $j \in (0, nr)$
  \[
  \left. \frac{\partial \varphi}{\partial z} \right|_{k=0} = \frac{3}{4} \varphi(j,0) - \varphi(j,1) + \frac{1}{4} \varphi(j,2) = 0
  \]
  \[
  E_r(j,0) = \frac{1}{4} (\varphi(j-1,0) + \varphi(j+1,0))
  \]
  \[
  E_z(j,0) = 0
  \]

- Anode: $k = nz$, $j \in (0, nr)$
  \[
  \left. \frac{\partial \varphi}{\partial z} \right|_{k=nz} = -\frac{3}{4} \varphi(j,nz) + \varphi(j,nz-1) - \frac{1}{4} \varphi(j,nz-2) = 0
  \]
  \[
  E_r(j,nz) = \frac{1}{4} (\varphi(j-1,nz) + \varphi(j+1,nz))
  \]
  \[
  E_z(j,nz) = 0
  \]

- Rmin: $j = 0$, $k \in [0, nz]$
  \[
  \varphi(0,k) = \varphi_{\text{min}}
  \]
  \[
  E_r(0,k) = \frac{3}{4} \varphi(0,k) - \varphi(1,k) + \frac{1}{4} \varphi(2,k)
  \]
  \[
  E_z(0,k) = 0 \text{ (no gradient along z)}
  \]

- Rmax: $j = nr$, $k \in [0, nz]$
  \[
  \varphi(nr,k) = \varphi_{\text{max}}
  \]
  \[
  E_r(nr,k) = -\frac{3}{4} \varphi(nr,k) + \varphi(nr-1,k) - \frac{1}{4} \varphi(nr-2,k)
  \]
  \[
  E_z(nr,k) = 0
  \]

BC 3 for a periodic $\varphi$ along $z$: Boundary grid points are treated as inner grid points, implicit periodicity is used.
CHAPTER 2. DOCUMENTATION

- Cathode: \( k = 0, \ j \in (0, nr) \)
  \[
  \left(1 - \frac{1}{2j}\right) \phi(j - 1,0) + \phi(j, nz - 1) - 4\phi(j,0) + \phi(j,1) + \right.
  \left(1 + \frac{1}{2j}\right) \phi(j + 1,0) = \{n_e(j,0) + n_e(j, nz)\} - \{n_i(j,0) + n_i(j, nz)\}
  
  \]
  \[
  E_r(j,0) = \frac{1}{4}(\phi(j - 1,0) - \phi(j + 1,0)) \tag{2.172}
  
  E_z(j,0) = \frac{1}{4}(\phi(j, nz - 1) - \phi(j,1)) \tag{2.173}
  
- Anode: \( k = nz, \ j \in (0, nr) \)
  \[
  \left(1 - \frac{1}{2j}\right) \phi(j - 1,nz) + \phi(j, nz - 1) - 4\phi(j,nz) + \phi(j,1) + \right.
  \left(1 + \frac{1}{2j}\right) \phi(j + 1,nz) = \{n_e(j,0) + n_e(j, nz)\} - \{n_i(j,0) + n_i(j, nz)\}
  
  \]
  \[
  E_r(j,nz) = \frac{1}{4}(\phi(j - 1,nz) - \phi(j + 1,nz)) \tag{2.175}
  
  E_z(j,nz) = \frac{1}{4}(\phi(j, nz - 1) - \phi(j,1)) \tag{2.176}
  
- Symmetry axis: \( j = 0, \ k \in [0, nz] \)
  \[
  \phi(0, k - 1) - 6\phi(0, k) + \phi(0, k + 1) + 4\phi(1, k) = -n_e(0,k) - n_i(0,k)
  \tag{2.177}
  
  E_r(0,k) = 0 \text{ (symmetry)} \tag{2.178}
  
  E_z(0,k) = \frac{1}{4}(\phi(0, k - 1) - \phi(0, k + 1)) \tag{2.179}
  
In the special case of \( k = 0 \) and \( k = nz \) we replace \( k - 1 \) with \( nz - 1 \) and \( k + 1 \) with 1 whenever \( k - 1 \) and \( k + 1 \) is not defined and modify the right-hand side of Eq. 2.177 as follows: \(-\{n_e(j,0) + n_e(j, nz)\} - \{n_i(j,0) + n_i(j, nz)\}\).

- ‘Infinity’: \( j = nr, \ k \in [0, nz] \)
  \[
  \left.\frac{\partial\phi}{\partial r}\right|_{j=nr} = -\frac{3}{4}\phi(nr,k) + \phi(nr - 1,k) - \frac{1}{4}\phi(nr - 2,k) = 0
  \tag{2.180}
  
  E_r(nr,k) = 0 \tag{2.181}
  
  E_z(nr,k) = \frac{1}{4}(\phi(nr, k - 1) - \phi(nr, k + 1)) \tag{2.182}
  
Also here, in the special case of \( k = 0 \) and \( k = nz \) we replace \( k - 1 \) with \( nz - 1 \) and \( k + 1 \) with 1 whenever \( k - 1 \) and \( k + 1 \) is not defined.
2.12 Particle boundary conditions

The switch between different boundary condition options for the particles is the variable ‘PBC’ in h/dim.h. The options are the following:

**PBC 0** is used for ‘discharge’ mode, incl. sputtering and secondary electron yield calculations, to be combined with PIS 0;

**PBC 1** to remove particles at all the boundaries: $k = 0$, $k = nz$, $j = nr$

**PBC 2** to reflect particles at the boundaries

- Cathode ($z < z_{\text{min}}$):
  \[ z' = z_{\text{min}} + (z_{\text{min}} - z), \quad v'_z = -v_z \]
- Anode ($z > z_{\text{max}}$):
  \[ z' = z_{\text{max}} - (z - z_{\text{max}}), \quad v'_z = -v_z \]
- ‘Infinity’ ($r > r_{\text{max}}$):
  \[ r' = r_{\text{max}} - (r - r_{\text{max}}), \quad v'_r = -v_r \]

**PBC 3** for periodicity in z, reflection in ‘infinity’

- Cathode ($z < z_{\text{min}}$):
  \[ z' = z + (z_{\text{max}} - z_{\text{min}}) \]
- Anode ($z > z_{\text{max}}$):
  \[ z' = z - (z_{\text{max}} - z_{\text{min}}) \]
- ‘Infinity’ ($r > r_{\text{max}}$):
  \[ r' = r_{\text{max}} - (r - r_{\text{max}}), \quad v'_r = -v_r \]

**PBC 4** is used for a simplified ‘discharge’ model: At the anode and cathode, Cu is reflected back, a Cu$^+$ sputters Cu with 100 % and an e$^-$ sputters Cu with 1 % probability. At the radial boundary, particles are removed. Applicable in combination with PIS 5.
2.13 Particle injection schemes

The switch between different injection scheme options for the particles is the variable ‘PIS’ in h/dim.h. For constant injection cases, injection fluxes are determined by the ratio $p2inj/p2inj\_step$. The options are the following:

**PIS 0** is used for ‘discharge’ mode, incl. electron field emission and neutral evaporation from a field emitter of a radius $Rem$ (chosen in input.txt), to be combined with PIS 0;

**PIS 1** inject electrons and ions with constant flux, in uniform density within $R_{\text{max} \_ \text{inj} \_ \text{UD}}$ inner cells in $r$, all along $z$;

**PIS 2** inject electrons from the cathode and ions from the anode, both with a constant flux;

**PIS 3** inject electrons and ions with constant flux, at the centre of the system (by centre we mean both in $r$ and $z$);

**PIS 4** inject electrons and neutrals with constant flux, both from the cathode.

**PIS 5** is used for a simplified ‘discharge’ model, in which electrons and neutrals are injected from the cathode with a constant flux. To be combined with the simple sputtering boundary conditions of PBC 4.
2.14 Collision routines

2.14.1 Coulomb collisions for e\(^{-}\)-e\(^{-}\) and Cu\(^{+}\)-Cu\(^{+}\) pairs

Coulomb collisions between the same species are treated with the Takizuka-Abe binary collision model (Ref. [7]) in the function \texttt{coll\_el\_knm\_2D()} with the implementation described in the PhD thesis of K. Matyash [15].

In the model, the changes in velocity due to Coulomb collisions are characterised by the angle \(\vartheta\), and the variable \(\delta \equiv \tan \vartheta/2\) is chosen randomly from a Gaussian distribution such that the variance \(<\delta^2>\) is given by

\[
<\delta^2> = \frac{\alpha_{\alpha} q_{\alpha} n_L \ln \lambda}{8 \pi \varepsilon_0^2 m_s^2 v_{rel}^3} \Delta t_{coll}\]

where \(\alpha\) and \(\beta\) denotes the species, \(n_L\) is the lowest of different species’ densities, \(\ln \lambda\) is the Coulomb logarithm, \(m_s\) is the reduced mass, and \(v_{rel}\) the relative velocity. In our case \(\alpha = \beta = e^{-}\) or Cu\(^{+}\), therefore \(n_L = n_{\alpha} = n_{\beta}\), and \(m_s = M_p/2\), where \(M_p\) is the electron or ion mass.

When rescaling the above expression to dimensionless quantities, just as before, \(v_{rel} = \frac{\Delta z}{\Delta t_{\text{ion}}} \tilde{v}_{\text{ion}}\), \(\Delta t_{coll} = \Delta t \cdot n_{\text{coll\_el}}\), masses are expressed in \(m_e\), and the density in a cell can be calculated based on the number of simulation particles in the cylindrical geometry as

\[
n = \frac{N_{SP} N_p}{V_{cell}} = \frac{N_{SP} N_p}{\pi (2j + 1) \Delta z^3},
\]

where \(N_p\) is the number of simulation particles in a given cell determined by points \((j, j+1, k, k+1)\), recorded in the variable \texttt{ordcount}\([j, k]\), and the volume of that cell is \(V_{cell} = \int_{j}^{j+1} dr \int_{k}^{k+1} dz = 2\pi \Delta z \int_{j}^{j+1} r dr = \pi (2j + 1) \Delta r^2 \Delta z\) (note: Not to be confused with the Verboncoeur volumes, which are calculated on the grid points).

Substituting these into the expression for \(<\delta^2>\) we arrive at

\[
\frac{\alpha_{\alpha} q_{\alpha} n_L \ln \lambda}{8 \pi \varepsilon_0^2 m_s^2 v_{rel}^3} \Delta t_{coll} = \text{vel\_scale} \frac{e^4}{2 \pi \varepsilon_0^2 m_e^2} \left( \frac{m_e}{M_p} \right)^2 \ln \Lambda \frac{\Delta t^4}{\Delta z^3} \frac{N_{SP}}{(2j + 1) \Delta z^3} n_{\text{coll\_el}} \frac{N_p}{v_{rel}^3} \] \hspace{1cm} (2.185)

where \text{vel\_scale} is 1 for electrons and \(dt_{\text{ion}}^{3}\) for ions due to the scaling of \(v_{rel}^3\). Note that in order to keep the collision frequency for simulated particles the same as for real particles, we used \(q_{\alpha} = \pm e\) and \(m_{\alpha} = m_{e-(Cu^{+})}\) and did not multiply charges and masses by \(N_{SP}\).

Since \(\frac{e^4}{\varepsilon_0^2 m_e^2} = \frac{\omega_{pe}}{n_{ref}^2}\) and \(\omega_{pe} \Delta t = \tilde{\Delta} t\), we can simplify the above to

\[
<\delta^2> = \text{vel\_scale} \frac{\ln \Lambda}{2\pi^2} \left( \frac{m_e}{M_p} \right)^2 \left( \frac{\tilde{\Delta} t}{\Delta z} \right) \frac{N_{SP}}{(2j + 1) \Delta z^3} n_{\text{coll\_el}} \frac{N_p}{v_{rel}^3} \equiv \text{Acoll}
\]

which are exactly the factors appearing in the code.
2.14.2 Other collisions

In this section, we treat the amplitude scaling in collision routines which apply a known (obtained by measurement or fit) cross-section \( \sigma \). Linear interpolation to cross-section data is carried out in init.cpp. The following routines all belong to this category:

- `coll_ion_neutral_noSP_2D()`, treating ion-neutral elastic collisions with charge exchange, \( \text{Cu}^+ + \text{Cu} \rightarrow \text{Cu} + \text{Cu}^+ \)
- `coll_n_n_2D()`, handling neutral-neutral elastic collisions, \( \text{Cu} + \text{Cu} \rightarrow \text{Cu} + \text{Cu} \)
- `coll_el_all_fake_2D()`, covering electron-neutral elastic collisions, \( e^- + \text{Cu} \rightarrow e^- + \text{Cu} \)
- `coll_el_neutrals_2D()`, treating the impact ionisation process, \( e^- + \text{Cu} \rightarrow e^- + \text{Cu}^+ \).

In all these routines, the probability that a collision occurs is approximated as

\[
P_i = 1 - e^{-\sigma_n \text{Cu} v_{\text{rel}} \Delta t_{\text{coll}}} \approx \sigma_n \text{Cu} v_{\text{rel}} \Delta t_{\text{coll}}.
\]  

(2.187)

The requirement that this probability should remain unaltered when rescaling to dimensionless quantities determines the scaling of \( \sigma \). As before, \( v_{\text{rel}} = \frac{\Delta z}{\Delta t_{\text{ion}}} \), \( \Delta t_{\text{coll}} = \Delta t \cdot n_{\text{coll ion}} \), and the dimensional density in a given cell is \( n = \frac{N_{\text{SP}} N_p}{V_{\text{cell}}} = \frac{N_{\text{SP}} N_p}{\pi (2j+1) \Delta z^2} \).

We then can determine \( \text{CScoeff} \equiv \tilde{\sigma}/\sigma \) through demanding

\[
\tilde{\sigma} N_p \tilde{v} \cdot 1 = \text{CScoeff} \cdot \frac{\pi (2j+1) \Delta z^3}{N_{\text{SP}}} \cdot \frac{\Delta t}{\Delta z} \cdot \frac{1}{n_{\text{coll ion}}} (\sigma_n \text{Cu} v_{\text{rel}} \Delta t_{\text{coll}})
\]  

\[\text{CScoeff} = \frac{N_{\text{SP}} n_{\text{coll ion}}}{\pi (2j+1) \Delta z^2} = \frac{n_{\text{ref}} \lambda^3 D_b n_{\text{coll ion}}}{\pi (2j+1) \Delta z^2 \lambda^2 D_b N_{Db}} = \frac{n_{\text{ref}} \lambda^3 D_b n_{\text{coll ion}}}{\pi \Delta z^2 N_{Db}}
\]

\[\text{CScoeff} = \frac{1}{2j+1}
\]

(2.188)

(2.189)

In the code, the correction with the factor \( 1/(2j+1) \) happens separately when calculating the probability in Eq. 2.187.
2.15 Control routines, stability of PIC

To ensure the stability of the solution calculated with PIC, the smallest time- and length scale phenomena have to be well resolved, and therefore the following conditions have to be fulfilled all the time (cf. Sec. 1.1):

\[ \Delta t \lesssim 0.2 \omega_{pe}^{-1} \]  
\[ \Delta z \lesssim 0.5 \lambda_{Db} \]  

(2.190)  
(2.191)

Since the computation happens in dimensionless quantities, the electron temperature and the electron density corresponding to the \( \omega_{pe}^{-1} \) and \( \lambda_{Db} \) have to be initially guessed, these guessed values are the reference values \( T_{\text{ref}} \) and \( n_{\text{ref}} \), and iteratively adjusted until the guess is sufficiently close to reality. To have a feedback on whether the chosen scaling is suitable or not, a diagnostic routine is built in, which investigates the fulfilment of

\[ \Delta t = 0.2 \left( \omega_{pe}^{-1} \right)^{\text{guessed}} \lesssim 0.2 \left( \omega_{pe}^{-1} \right)^{\text{real}} \]  
\[ \Rightarrow \frac{1}{\sqrt{n_{\text{ref}}}} \lesssim \frac{1}{\sqrt{n_e}} \]  
\[ n_e \lesssim n_{\text{ref}} \]  

(2.192)  
(2.193)  
(2.194)

and similarly the fulfilment of

\[ \Delta z = 0.5 \left( \lambda_{Db} \right)^{\text{guessed}} \lesssim 0.5 \left( \lambda_{Db} \right)^{\text{real}} \]  
\[ \Rightarrow \sqrt{\frac{T_{\text{ref}}}{n_{\text{ref}}}} \lesssim \sqrt{\frac{T_e}{n_e}} \]  
\[ \frac{n_e}{T_e} \lesssim \frac{n_{\text{ref}}}{T_{\text{ref}}} \]  

(2.195)  
(2.196)  
(2.197)

In practice, \( n_e \lesssim 5n_{\text{ref}} \) can be allowed, but the fraction \( \frac{n_e}{T_e} \) should not be exceeded by much. Note that here \( T_e \) is not defined as standard deviation of a Maxwellian distribution, so for instance, when an external electric field is applied, the drift velocity should also be taken into account (contrary to the temperature outputted, cf. Sec. 2.10).
2.16 Miscellaneous useful formulae

Debye length
\[
\lambda_{Db} = \sqrt{\frac{\varepsilon_0 T_e}{e^2 n_e}} = \sqrt{\frac{552635 \cdot T_e}{e^2 n_e}} \text{ [cm]} \quad (2.198)
\]

Plasma frequency
\[
\omega_{pe} = \sqrt{\frac{e^2 n_e}{\varepsilon_0 m_e}} = 56414.6 \sqrt{n_e} \left[ \frac{1}{8} \right] \quad (2.199)
\]

Gyroradius (Larmor radius)
\[
r_g = \frac{m_e v_\perp}{eB} \quad (2.200)
\]
where \( B \) is the magnitude of a constant external magnetic field and \( v_\perp \) is the speed of the electron perpendicular to the magnetic field.

Gyrofrequency
\[
\nu_g = \frac{eB}{2\pi m_e} \quad (2.201)
\]

Thermal velocity
\[
v_{T,e} = \lambda_{Db} \omega_{pe} = \sqrt{\frac{T_e}{m_e}} \quad (2.202)
\]

Sound velocity
\[
c_{s,i} = \sqrt{\frac{m_e}{M_i} v_{T,e}} \quad (2.203)
\]

Number of superparticles
\[
N_{SP} = \frac{n_e \lambda_{Db}^3}{N_{Db}} \quad (2.204)
\]
where \( N_{Db} \) is defined as the number of superparticles in a Debye cube corresponding to the density \( n_{ref} \).

Current density (in A/cm\(^2\))
\[
j_p = j \cdot p2inj, \quad (2.205)
\]

where \( p2inj \) is the number of particles to inject within \( p2inj\_step \) steps. The factor \( j \) is
\[
j = e \frac{N_{SP} \omega_{pe}}{p2inj\_step \lambda_{Db}^2} = 6.7193 \cdot 10^{-12} \frac{n_{ref} \sqrt{T_{ref}}}{N_{Db} p2inj\_step}, \quad (2.206)
\]
in case that \( p2inj\_step \) is given in \( \omega_{pe}^{-1} \)'s. If \( p2inj\_step \) is given in time steps, \( \tilde{\Delta}t \) has to be taken into account in the denominator:
\[
j = 6.7193 \cdot 10^{-12} \frac{n_{ref} \sqrt{T_{ref}}}{N_{Db} p2inj\_step \tilde{\Delta}t} = 3.36 \cdot 10^{-11} \frac{n_{ref} \sqrt{T_{ref}}}{N_{Db} p2inj\_step} \quad (2.207)
\]
Ionisation rate Using uniform and monoenergetic electron and neutral beams in a head-on collision provides an easy way to cross-check the ionisation routine. The theoretically predicted ionisation rate is given by

\[ R = n_e \cdot n_{Cu} \cdot v_{rel} \cdot \sigma(E), \] (2.208)

where \( n_e \) and \( n_{Cu} \) are the electron and neutral number densities, respectively, \( v_{rel} \) is the relative speed of the beams and the total cross-section \( \sigma \) is calculated at the energy \( E = \frac{1}{2} m_s v_{rel}^2 \), with \( m_s \approx m_e \) being the reduced mass. The unit of \( R \) is \([R] = \frac{1}{cm^3 \cdot s}\) (or equally well \([R] = \frac{1}{m^3 \cdot s}\)).

This rate is then to be compared with the simulated rate

\[ R = \frac{\Gamma_{real}}{V \cdot \Delta t} = \frac{NSP \cdot \Gamma_{sim}}{V \cdot \Delta t}, \] (2.209)

where \( \Gamma_{sim} \) is the number of simulated ions yielded in the collisions during a time step of \( \Delta t \) in a volume of \( V \), and where electrons and neutrals were initially uniformly distributed, e.g. \( V = nz \cdot \pi \cdot (2j + 1) \Delta r^2 \Delta z \) for particles distributed in the inner \( j \) cells.

Ion species

<table>
<thead>
<tr>
<th>Sorts</th>
<th>( j )</th>
<th>Ion</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 )</td>
<td>H(^+) (reference species)</td>
<td></td>
</tr>
<tr>
<td>( 1 )</td>
<td>Cu(^+)</td>
<td></td>
</tr>
<tr>
<td>( 2 )</td>
<td>Cu</td>
<td></td>
</tr>
</tbody>
</table>

Masses

\[ M_{ions[0]} = M_{H^+} = 1836 \cdot m_e \] (2.210)
\[ M_{ions[1]} = M_{Cu^+} = (63 \cdot 1836 + 62)m_e \] (2.211)
\[ M_{ions[2]} = M_{Cu} = (63 \cdot 1836 + 63)m_e \] (2.212)
Bibliography


