FENNECS: a novel particle-in-cell code for simulating the formation of magnetized non-neutral plasmas trapped by electrodes of complex geometries

G. Le Bars[a,*], J. Loizu[a], S. Guinchard[a], J.-Ph. Hogge[a], A. Cerfon[b], S. Alberti[a], F. Romano[a], J. Genoud[a], P. Kamiński[a]

[a] *Ecole Polytechnique Fédérale de Lausanne (EPFL), Swiss Plasma Center (SPC), Lausanne, CH-1015, Switzerland*

[b] *Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 USA*

Abstract

This paper presents the new 2D electrostatic particle-in-cell code FENNECS developed to study the formation of magnetized non-neutral plasmas in geometries with azimuthal symmetry. This code has been developed in the domain of gyrotron electron gun design, but solves general equations and can be applied in other domains of plasma physics. FENNECS is capable of simulating electron-neutral collisions using a Monte Carlo approach and considers both elastic and inelastic (ionization) processes. It is also capable of solving the Poisson equation on domains with arbitrary geometries with either Dirichlet or natural boundary conditions. The Poisson solver is based on a meshless Finite Element Method, called web-splines, based on b-splines of any order, and used for the first time in the domain of plasma physics. In addition, the effect of fast ions colliding with the electrodes and causing ion induced electron emission at the electrode surfaces has been implemented in the code. In this paper, the governing equations solved by FENNECS and the numerical methods used to solve them are presented. A number of verification cases are then reported. Finally, the parallelization scheme used in FENNECS and its parallel scalability are presented.

Keywords: particle-in-cell; finite element methods; web-splines; Monte Carlo; non-neutral plasma; gyrotron

*Corresponding author.
E-mail address: guillaume@lebars.ch

1. Introduction

The understanding of the dynamics of non-neutral plasmas is relevant to many fields in physics and engineering, from extremely cold and low density plasmas in elementary particle physics [1], to high-energy and high density plasmas in particle accelerators [2] or microwave sources [3]. These plasmas possess interesting stability properties that allow them to be stored for long periods of time ranging from hours to days [4] and allow the accumulation of antimatter charged particles. They play a central role in the study of antimatter (for example the effect of gravity on anti-Hydrogen [5, 6]), or in the accumulation and storage of positrons necessary to study electron-positron plasmas [7]. In the domain of microwave sources, non-neutral plasmas can be used to generate or amplify RF waves [3], but they can also form in the microwave device at undesired places, hindering the normal operation of the source. For example in some high power gyrotrons used to heat the electrons of magnetically confined fusion plasmas, electron clouds can form in the electron gun region of the device, preventing their nominal operation [8, 9]. These clouds are formed due to the presence of potential wells generated by the combination of externally applied electric and magnetic fields and are similar in nature to the ones characterizing Penning traps [1, 9, 10]. Due to their high kinetic energy, the trapped electrons can ionize the residual neutral gas present in the gun and lead to high density clouds that can cause damaging currents between the accelerating electrodes of the gun and can lead to arcing events. To understand the conditions of formation of these clouds and their evolution, a particle-in-cell code called FENNECS (Finite Element Non-Neutral Electron Cloud Simulator) has been developed and is the subject of this publication.

Current codes considering neutral and non-neutral plasma discharges are either proprietary (LSP [11], MAGIC [12]), or are limited to simple electrode geometries (NINJA [13]) due to the method used to solve Poisson's equation. In the domain of gyrotrons, electron gun simulations are carried out with codes capable of simulating the complex geometry of the gun. However, the most common codes such as EGUN [14], ESRAY [15], DAPHNE [16] or ARIADNE++ [17] do not consider electron neutral collisions and assume a beam-optics framework, which considers the long time scale evolution and neglects the fast electrostatic modes that can arise in high density electron clouds. Another candidate, the WARP [18–20] code, is both capable of simulating electron neutral collisions and the complex electrodes geometries used in gyrotron electron guns. However, due to the finite difference method on staggered grids used to solve Poisson's equation, simulations of complex geometries require potentially costly grid refinements.

The 2D electrostatic axisymmetric particle-in-cell code FENNECS presented in this paper considers a novel Finite Element Method (FEM) that allows the exact definition of the electrodes geometry and somewhat decouples the grid definition and the geometry of interest when solving for the electrostatic potential. This method has been successfully used in several domains of physics, for example to solve elastic deformation problems [21], electromagnetic wave propagation in wave-guides [22], or the stationary Stokes problem [23], and is used here for the first time in plasma physics, to the best of the authors knowledge [24, 25]. The code is also capable of simulating electron neutral collisions, considering elastic and inelastic (ionization) collisions, and resolves the fast time-scale plasma waves and electron cyclotron motion. This is of great importance for simulating high-density trapped electron clouds for which the Brillouin ratio, defined as $f_b = 2\omega_{pe}^2/\Omega_{ce}^2$ is close to one. Here, $\omega_{pe}^2 = e^2 n_e/\epsilon_0 m_e$ is the electron plasma frequency squared and $\Omega_{ce} = eB/m_e$ the electron cyclotron frequency. e and m_e are, respectively, the electron charge and mass, n_e is the electron density, B is the magnetic field amplitude, and ϵ_0 is the vacuum permittivity. The magnetic fields generated by the electron clouds are neglected in front of the strong externally applied magnetic field that is assumed to dominate the dynamics. This code has already been successfully used to study the self-consistent formation of trapped electron clouds in gyrotron electron guns and to derive scaling laws for the electron cloud density and resulting current as a function of external parameters [9]. In the same context, FENNECS was successfully validated against experimental measurements [26]. The code is currently used to study gyrotron electron guns. However, the authors believe that the governing equations are sufficiently general that it could be used in the domain of Penning traps [1], or be easily adapted to study cathodic arcs [27]. In addition, the code and its dependencies are, at the time of writing, under the process of licensing to allow the open-source publication and sharing of the code. This will facilitate its modification, enable further improvements, and simplify the beginning of new collaborations. However, the code is already accessible at `https://c4science.ch/source/fennecs`.

The goal of the present paper is to present the FENNECS code and capabilities, as well as the numerical methods used. After the introduction, Sec.2 describes the physical model implemented in FENNECS. In Sec.3, the numerical methods used to simulate different physical phenomena are described, and special care is taken to describe the novel FEM solver based on weighted extend b-splines (web-splines). The Monte Carlo methods used to simulate electron-neutral collisions and ion induced electron emission on the electrode surfaces are also described. In Sec.4, a set of verification test cases confirming the correct

implementation of the governing equations is also presented. The parallelization schemes are described and the scalability of the code using domain decomposition is presented in Sec.5. Finally, a summary of the paper and its conclusions follow.

2. Physical model

2.1. Governing equations

FENNECS is an axisymmetric 2D3V electrostatic particle-in-cell code that solves the Boltzmann-Poisson system for an electron distribution function $f(\vec{r},\vec{v},t)$ and the electrostatic potential $\phi(\vec{r},t)$ with the addition of electron-neutral collision operators. The neutral gas is considered as a cold background gas of uniform density n_n. Only one neutral gas species is considered, and it is assumed that n_n does not change in time. In the current model, only elastic and single ionization collisions are considered with their respective collision cross-sections σ_{ela} and σ_{io}. This choice is supported by the fact that, due to the large radial electric field in gyrotron electron guns, the newly formed ions are lost on time-scale $\tau_{ion,loss}$ much smaller than the second ionization collision time scale. Similarly, due to the large electron kinetic energies (more than several hundred eV), the collision time-scales for excitation of the neutral gas are at least one order of magnitude larger than both the elastic and single ionization time-scales and are therefore neglected [9, 28]. In elastic and ionization collisions, we assume anisotropic scattering cross-sections using a screened-Coulomb scattering cross-section [29]. For an ionization event, the remaining kinetic energy after collision (initial kinetic energy minus the ionization energy) is split between the freed and the incoming electron using a double differential cross-section $\frac{\partial^2\sigma_{\text{io,sec}}}{\partial\Omega\partial E_p}$ for the energy of the secondary electron, and a second double differential cross-section $\frac{\partial^2\sigma_{\text{io,sca}}}{\partial\Omega\partial E_p}$ for the energy of the scattered electron. These cross-sections depend on the solid scattering angle Ω and the incoming electron energy E_p [30]. The two double differential cross-sections are defined such that, in ionization events, the total energy is conserved $E_p = E_{\text{io}} + E_{\text{sca}} + E_{\text{sec}}$. Here, E_{io} is the first ionization energy of the neutral gas, E_{sca} is the energy of the scattered electron and E_{sec} is the energy of the secondary electron.

The magnetic field $\vec{B}_0^{\text{ext}}$ is imposed externally, and the magnetic field generated by the electron cloud is neglected. Perfectly absorbing boundary conditions for the particles are used at the electrodes, thereby representing a loss term L_{wall}. In addition, a volumetric seed source S_{seed} can be imposed, and electron emission due to ions impacting the electrode surfaces can be simulated, introducing a

surface source S_{IIEE}. In this case, the Boltzmann equation becomes

$$\begin{aligned}
&\left[\frac{\partial}{\partial t}+\vec{v}\cdot\frac{\partial}{\partial \vec{r}}-\frac{e}{m_e}\left(\vec{E}+\vec{v}\times\vec{B}_0^{\mathrm{ext}}(\vec{r})\right)\cdot\frac{\partial}{\partial \vec{v}}\right]f(\vec{r},\vec{v},t)= \\
&+n_n|\vec{v}|\int\frac{\mathrm{d}\sigma_{\mathrm{ela}}(|\vec{v}|)}{\mathrm{d}\Omega}[f(\vec{r},\vec{v}'(\Omega),t)-f(\vec{r},\vec{v},t)]d\Omega \\
&+\frac{n_n|\vec{v}|}{E}\left(\int_{E+E_{\mathrm{io}}}^{2E+E_{\mathrm{io}}}\int E_p\frac{\partial^2\sigma_{\mathrm{io,sec}}}{\partial\Omega\partial E_p}f(\vec{r},\vec{v}'(\Omega),t)d\Omega dE_p\right. \\
&\left.+\int_{2E+E_{\mathrm{io}}}^{\infty}\int E_p\frac{\partial^2\sigma_{\mathrm{io,sca}}}{\partial\Omega\partial E_p}f(\vec{r},\vec{v}'(\Omega),t)d\Omega dE_p\right) \\
&-n_n|\vec{v}|\sigma_{\mathrm{io}}(|\vec{v}|)f(\vec{r},\vec{v},t) \\
&+S_{\mathrm{seed}}+S_{\mathrm{IIEE}}-L_{wall}.
\end{aligned} \tag{1}$$

Here, Ω is the solid scattering angle; $\vec{v}'$ is the electron velocity that is scattered by a solid angle Ω to $\vec{v}$ [31]; $E=m_e|\vec{v}|^2/2$ is the electron energy at velocity $\vec{v}$. On the right hand side, the first term accounts for the scattering by elastic collisions [31], the second term describes the emission of secondary electrons by ionization, the third term accounts for the scattering of incoming electrons during an ionization collision and the fourth term describes the electrons removed from the distribution by ionization [32]. The electric potential ϕ is solved self-consistently using Poisson's equation

$$\nabla^2\phi(\vec{r},t)=-\frac{\rho}{\epsilon_0}=\frac{e}{\epsilon_0}\int f(\vec{r},\vec{v},t)d^3\vec{v} \tag{2}$$

and considering the boundary conditions imposed by the electrodes. At the electrodes, in addition to fixed potentials imposed by ideal power supplies, the resistive and capacitive effects of a non-ideal power-supply can be simulated (see section 3.7).

2.2. Normalizations

To improve the numerical precision of the code, all the physical quantities are normalized by physical constants relevant to the problem. To this end, four reference quantities are used: B_0, an input parameter usually set to the maximum amplitude of the magnetic field in the simulation domain; c, the speed of light in vacuum; e, the electron charge; and m_e, the electron mass. The time is normalized by the inverse of the cyclotron frequency $t_N=1/\Omega_{ce}=m_e/eB_0$, velocities are normalized to $v_N=c$, lengths are normalized to $r_N=v_N/t_N$, the magnetic field

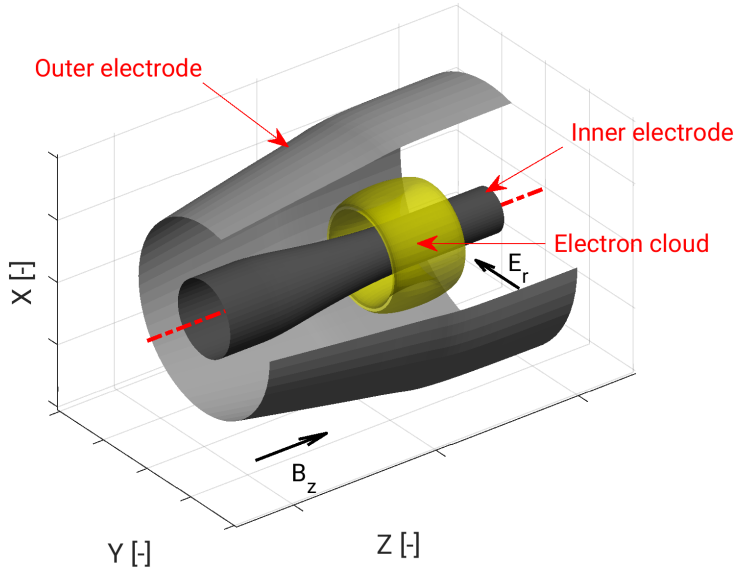

Figure 1: Typical geometry of interest used in FENNECS. The yellow ring represents an example of an electron plasma cloud. The gray/black parts are the electrodes on which a fixed potential can be applied. The red dotted-dashed line highlights the axis of symmetry.

is given in units of $B_N = B_0$, and the electric potential and fields are respectively normalized by $\phi_N = B_N v_N r_N$ and $E_N = v_N B_N$.

2.3. Geometries of interest

As FENNECS is a 2D axisymmetric code, it is capable of simulating geometries with an azimuthal symmetry. Namely, in cylindrical coordinates (r, θ, z) all fields can depend on r and z but not on θ. For gyrotron electron guns, the typical geometry of interest, as represented in Figure 1, is composed of a coaxial cathode and an outer cylindrical anode described by an arbitrary radial profile $r(z)$. However, the code is more flexible in the definition of the electrodes and multiple concentric electrode rings can be defined. Similarly, regions where only the outer electrode is present are also possible. This is important to simulate all types of gyrotron electron guns, but can also be useful to study non-neutral plasmas in other physical settings. In addition independent potentials can be applied to each simulated electrode. Furthermore, the particles are subjected to an axisymmetric external magnetic field with both radial and axial components. In the configurations typical of gyrotron guns, the electron clouds usually have an annular shape (see Figure 1) confined axially by magnetic mirrors, due to the electrons large per-

pendicular velocity, or by electrostatic potential wells imposed externally [9, 10]. However, cylindrical clouds can also be simulated.

2.4. Time scale separation

Typical electron clouds trapped in gyrotron electron guns are subject to physical phenomena happening on various time-scales. These can span up to ten orders of magnitude, between the fast electron cyclotron motion at Ω_{ce}, and the slow ionization collision frequency f_{io} and effective elastic collision frequency for momentum exchange f_d, as illustrated in Figure 2. This large time scale separation prevents the complete simulation of all the time-scales due to the numerical cost of the task. However, as $f_{\mathrm{io}} = n_n < \sigma_{\mathrm{io}} v >$ and $f_d = n_n < \sigma_{\mathrm{ela}} v >$ are linearly proportional to n_n, the collision time-scales can be shortened by considering, in the simulations, an increased neutral gas pressure $n_{n,simu} = \alpha n_{n,phys}$ compared to the physical pressure of interest [9, 26]. This factor $\alpha > 1$ must be selected such that a sufficient time-scale separation is kept between the slow and fast time-scales, namely such that $f_{\mathrm{io}}, f_d \ll f_{\parallel}$. Here, $f_{\parallel} \approx v_{\parallel}/L$ is the electron bounce frequency in the trap of length L along the magnetic field lines, and $v_{\parallel}$ is the electron velocity parallel to the magnetic field line. The simulation characteristic times of particle losses can then be rescaled to the physical time-scales using the same parameter α.

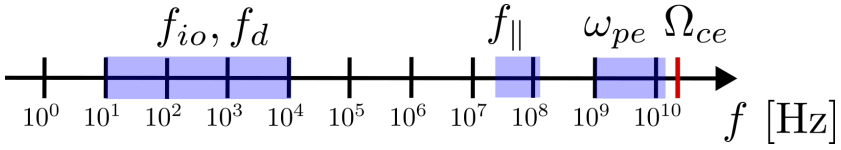

Figure 2: Relevant time scales for electron clouds of interest simulated by FENNECS. f_{io} and f_d are the ionization collision frequency and the effective elastic collision frequency for momentum exchange; $f_{\parallel} \approx v_{\parallel}/L$ is the electron bounce frequency in the trap of length L along the magnetic field lines; ω_{pe} is the electron plasma frequency; Ω_{ce} is the electron cyclotron frequency. The blue shaded area indicates the range of possible time scales for $n_e \approx 10^{14} \rightarrow 10^{17}\mathrm{m}^{-3}$ and $n_n \approx 10^{13} \rightarrow 10^{17}\mathrm{m}^{-3}$. The red line give Ω_{ce} for $B = 0.3\,\mathrm{T}$.

3. Numerical methods

To solve the Boltzmann equation (1) and the Poisson equation (2), the particle-in-cell method is employed. The distribution function f is sampled using a finite

number of macro particles i at position $\vec{r}_i$ with velocity $\vec{v}_i$ and each representing N_i electrons such that

$$f(\vec{r},\vec{v},t) = \sum_i N_i \delta(\vec{r}-\vec{r}_i)\delta(\vec{v}-\vec{v}_i). \tag{3}$$

Here $\delta(\vec{x})$ is the Dirac delta function. This representation of the particles has been chosen to reduce the number of computations necessary to calculate the right-hand side of Poisson's equation, to facilitate the particles removal at the boundaries, and to facilitate the parallelization of the code. On the other hand, this choice forces the use of a relatively large number of macro-particles in order to minimize numerical noise.

Starting from an initial distribution of macro-particles, the code performs the following steps to advance in time the particles and the fields according to the Boltzmann-Poisson system described in equations (1) and (2). As illustrated in the diagram of Figure 3, at each time-step, the code:

1. Localizes each particle in the geometry and calculates its FEM cell index. Removes the particles that are outside the vacuum region, and generates the secondary electrons for the lost ions if the Ion Induced Electron Emission (IIEE) module is activated (Sec. 3.9). In Message Passing Interface (MPI) parallelism, the particles that are leaving or entering the local domain simulated by each process are exchanged between the neighbouring processes.
2. Runs the Monte Carlo collision routine for each particle and scatter/reduce their velocity vector accordingly, and adds the freed electrons due to ionization of the neutral gas (Sec. 3.8).
3. Injects the new particles according to the seed source distribution function (Sec. 3.10).
4. Computes the new applied bias according to the collected current on the electrodes when the non-ideal power supply module is activated (Sec. 3.7).
5. Computes the right-hand side of Poisson's equation by looping on all the macro-particles (Sec. 3.3).
6. Solves Poisson's equation (Sec. 3.3).
7. Computes for each particle the value of $\vec{E}$ and $\vec{B}$ at their position and advances in time their velocity (Sec. 3.4 and 3.2).
8. Saves on file the requested diagnostic quantities (particles positions and velocities, electric field evaluated on the grid, moments of the distribution function evaluated on the grid, etc.) (Sec. 3.11).
9. Advances in time the particles' positions (Sec. 3.1).
10. Restarts the cycle.

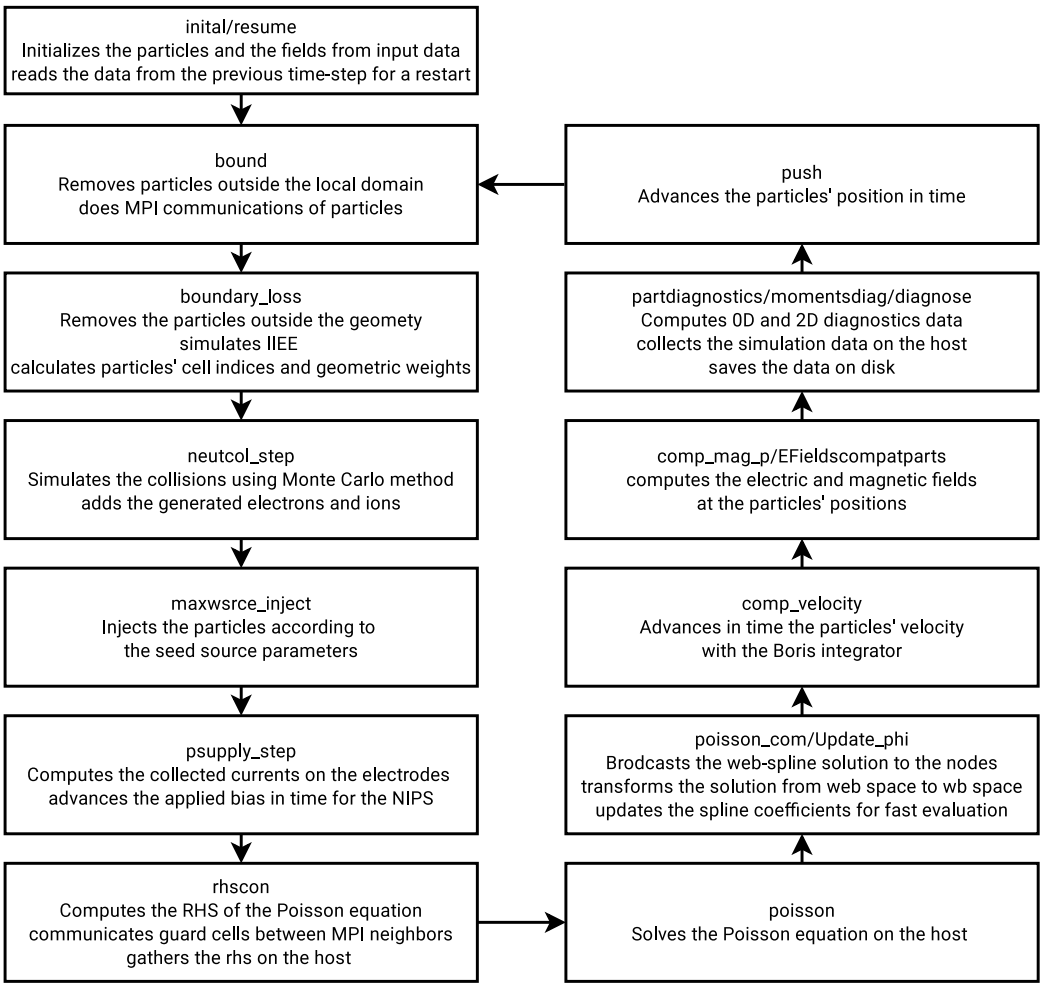

Figure 3: Flow chart of the FENNECS simulations with the relevant subroutines called in the code.

3.1. Particle trajectory: Boris algorithm

To advance in time the macro-particle positions and velocities, according to the left-hand side of Boltzmann equation (1), the Boris algorithm is used [33]. This method was selected for its simplicity and reliability, and for its capability of integrating both classical and relativistic trajectories with very little change in the code. This allows the user of FENNECS to select at run-time if the classical or relativistic Newton's equation is solved during the simulations, by means of an input flag. The Boris algorithm is a second order in time ($O(\Delta t^2)$) explicit integrator based on a leap-frog scheme, meaning that the particles positions and velocities are never known at the same time-step. Instead, the positions are known at times t_i and the velocities are known at times $t_{i+1/2} = t_i + \Delta t/2$. This is important to take into account during the initialization of the particles and when calculating diagnostics quantities as a naive evaluation will lead to a reduced accuracy of $O(\Delta t)$. To advance in time the velocities at position $\vec{r}_{t_i}$ from $\vec{v}_{t_i-\Delta t/2}$ at $t_i - \Delta t/2$ to $\vec{v}_{t_i+\Delta t/2}$ at $t_i + \Delta t/2$, the algorithm

1. advances the velocity by half the electric field to $\vec{v}_-$,
2. rotates the velocity due to the magnetic field force to $\vec{v}_+$,
3. advances the velocity by half the electric field to $\vec{v}_{t_i+\Delta t/2}$.

To solve the relativistic Newton equation, the new variable $\vec{u} = \gamma\vec{v}$ is used, with $\gamma = (1 - v^2/c^2)^{-1/2}$ the Lorentz relativistic factor. This gives in equation form:

$$\vec{u}_- = \vec{u}_{t_i-\Delta t/2} + \frac{q\Delta t}{2m}\vec{E}_{t_i}(\vec{r}_{t_i}), \tag{4}$$

$$\vec{u}' = \vec{u}_- + \vec{u}_- \times \vec{t}, \tag{5}$$

$$\vec{u}_+ = \vec{u}_- + \vec{u}' \times \vec{s}, \tag{6}$$

$$\vec{u}_{t_i+\Delta t/2} = \vec{u}_+ + \frac{q\Delta t}{2m}\vec{E}_{t_i}(\vec{r}_{t_i}). \tag{7}$$

Here used has been made of the two rotation vectors

$$\vec{t} = \frac{q\vec{B}(\vec{r}_{t_i})}{\gamma_{t_i} m}\frac{\Delta t}{2}, \tag{8}$$

and

$$\vec{s} = \frac{2\vec{t}}{1+t^2}, \tag{9}$$

with q the macro-particle charge, m its mass and $\gamma_{t_i} = (1 - v_-^2/c^2)^{-1/2}$. The classical Newton equation is recovered if $\gamma \equiv 1$ is imposed numerically. The particles' position is finally advanced with:

$$\vec{r}_{t_i+\Delta t} = \vec{r}_{t_i-\Delta t} + \Delta t \vec{v}_{t_i+\Delta t/2}. \tag{10}$$

3.2. Magnetic field

The magnetic field is imposed externally assuming azimuthal symmetry. It is also assumed that the external magnetic field amplitude is large enough so that the contribution from the electron cloud current can be neglected. It can be defined either using an analytical magnetic field vector potential that, e.g., approximates a magnetic mirror close to the magnetic axis, as described in Sec.4.2, or it can be calculated on the grid using standard Biot-Savart solvers and be used as an input for the simulations. Finally, at the particle position, the magnetic field is computed using linear interpolation from the FEM grid points values to reduce the computational cost of the evaluation.

3.3. Poisson: Web-spline method

The Poisson equation, for a scalar field ϕ and a source term Q, is solved on the domain D, closed by boundaries ∂D, using a Finite Element Method (FEM) based on bivariate b-splines of any order [25, 34, 35]. Dirichlet boundary conditions are imposed on boundaries ∂D_i and Neumann boundary conditions are imposed on boundaries ∂D_k such that:

$$-\nabla^2 \phi = Q \text{ in } D,\ \phi = g_i \text{ on } \partial D_i,\ \nabla_{\perp k} \phi = 0 \text{ on } \partial D_k, \tag{11}$$

where $\nabla_{\perp k}$ denotes the normal derivative perpendicular to ∂D_k. To define these boundary conditions on curved surfaces, the web-spline method is used for the first time in plasma physics, to the authors knowledge [24, 25]. This paper will be limited to the description of the method and of the points necessary for the implementation. Details regarding the numerical stability and accuracy of the method can be found in references [24, 25].

To derive a variational formulation, the electric potential ϕ is first rewritten to eliminate the inhomogeneous boundary conditions by setting

$$\phi = u + \tilde{g}, \tag{12}$$

with u a function that vanishes on ∂D_i and $\tilde{g}$ an extension of the Dirichlet boundary conditions g to all D. $\tilde{g}$ can be set to any smooth function such that $\tilde{g}(\vec{x}) =$

$g_i(\vec{x})\ \forall \vec{x} \in \partial D_i$. The Poisson equation is then multiplied by a test function ψ and integrated over D leading to the weak formulation:

$$\int_D \nabla u \nabla \psi = \int_D (Q\psi - \nabla \tilde{g} \nabla \psi). \tag{13}$$

To construct the Ritz-Galerkin approximation of the solution, the function ψ is taken to be a set of n_b basis polynomials Ψ_l^m of degree m with compact support on mesh cells of D, and the solution is approximated by a function ϕ_h such that

$$u_h = \sum_{l=1}^{n_b} u_l \Psi_l^m. \tag{14}$$

To ensure by construction that the Dirichlet boundary conditions are respected, the basis functions Ψ_l^m are defined such that they are 0 on ∂D_i. Solving the Ritz-Galerkin approximation of the solution then reduces to solving a system of linear equations for the coefficients u_l:

$$\overleftrightarrow{A} \cdot \vec{u} = \vec{\lambda}, \tag{15}$$

with $\overleftrightarrow{A}$ a matrix with coefficients

$$A_{lk} = \int_D \nabla \Psi_l^m \nabla \Psi_k^m, \tag{16}$$

and $\vec{\lambda}$ a vector with coefficients

$$\lambda_l = \int_D Q\Psi_l^m - \nabla \tilde{g} \nabla \Psi_l^m. \tag{17}$$

One can define a set of basis functions Ψ_l^m using weighted b-splines by defining a smooth geometric weight function w such that $w(x) = 0\ \forall x \in \partial D_i$, and w is positive inside the domain $D/\partial D_i$ and negative outside of D. In this case: $\Psi_l^m \equiv w b_{l,h}^m$, with $b_{l,h}^m$ the n-variate tensor product of b-spline of degree m, with grid width h, and support $(l_1, ..., l_n)h + [0, m+1]^n h$. Since the grid is regular (h is the same for all b-splines of the basis), the index h will be neglected for the rest of the paper. The weighted b-spline method is known to show bad numerical convergence as the grid width is reduced due to a strong increase of the condition number of the Ritz-Galerkin matrix [24]. This problem comes from the effect of boundary b-splines whose intersection between their support and the simulation

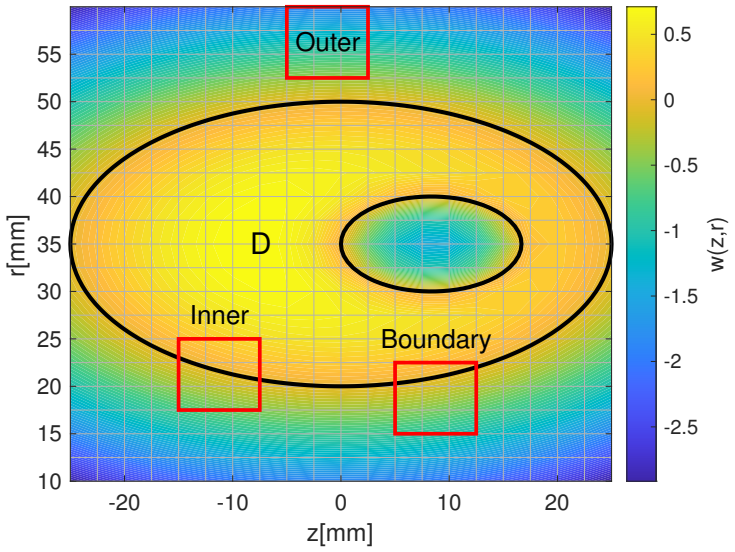

Figure 4: Geometric weight w and domain boundary for the test case presented in section 4.1 (black). The gray lines show the FEM grid, and the red squares show the boundary of the support of 3 types of quadratic b-splines.

domain becomes small. To alleviate this problem, Höllig and co-authors [24] combined boundary and inner b-splines to form a new basis called the web-spline basis.

To understand the web-spline basis, it is necessary to define inner, outer, and boundary b-splines, depending on the intersection between the support of the b-spline and the domain D. Inner b-splines need to have at least one grid-cell in their support that is fully inside D; for outer b-splines, the intersection between their support and D is $\emptyset$; and boundary b-splines are all the other b-splines. An example of this classification is represented in Figure 4, using the geometry defined in subsection 4.1 and quadratic b-splines. For the rest of this section, the ensemble L of inner b-splines are identified as b_l^m using the subscript l and the ensemble K of boundary b-splines are identified as b_k^m using the subscript k.

A web-spline basis $\Psi_{l,web}^m$ is defined by combining boundary and inner b-splines while keeping the correct approximation order of the initial b-spline space. To this end, the approximated solution u_h is separated between the contribution of internal and boundary b-splines

$$u_h = \sum_{l \in L} u_l b_l^m + \sum_{k \in K} u_k b_k^m. \tag{18}$$

According to Marsden's identity [36], if u_h is a polynomial of degree $\leq m$ the coefficients u_l and u_k can be calculated by a polynomial $\tilde{u}(i) = u_i$ of degree $\leq m$ dependent on the spline index i. In this case, the outer coefficients u_k can be interpolated, for the n-variate b-splines in a space of dimension n, from any $(m+1)^n$ inner indices without affecting the approximation power of the spline space. In practice, the interpolant indices are taken as the closest $(m+1)^n$ inner indices l, and the coefficients u_k are interpolated using Lagrange polynomials such that

$$u_k = \sum_{l \in L} u_l e_{lk}. \tag{19}$$

This assumes that the grid width h is sufficiently small for this list to exist. The coefficients e_{lk} are defined as

$$e_{lk} = \prod_{\nu=1}^{n} \prod_{\mu=0}^{m} \frac{k_\nu - p_\nu - \mu}{l_\nu - p_\nu - \mu}, \tag{20}$$

with n the dimension of the b-spline space and p_ν the lower index in each dimension of the inner interpolating splines. In addition to the interpolation of the external coefficients u_k, the web-spline basis is also rescaled by the weight evaluated at x_l, the center of a grid cell belonging to the support of b_l^m and fully inside D. The final web-spline basis is finally defined as

$$\Psi_{l,web}^m = \frac{w}{w(x_l)} \left[b_l^m + \sum_{k \in K(l)} e_{lk} b_k^m \right], \tag{21}$$

with $K(l)$ the ensemble of boundary b-splines for which the inner b-spline l is used to interpolate u_k. The Ritz-Galerkin matrix obtained using this new web-spline basis has a condition number which is reduced and remains stable as the grid width is reduced. This greatly improves the numerical stability of the finite element method and allows for arbitrary small grid width. In addition, this method allows to keep a regular grid and thus greatly simplifies the localization of the particles in the grid and facilitates the code parallelization with domain decomposition.

In FENNECS, since the distribution function is sampled with point-like macro particles defined with δ functions, the evaluation of the first term of λ_l is straight-

forward. This terms becomes, for an electron distribution function,

$$\begin{aligned}\int_D Q\Psi_l^m &= -\frac{e}{\epsilon_0}\int_D\int_v f(\vec{r},\vec{v},t)d\vec{v}\Psi_l^m = -\frac{e}{\epsilon_0}\int_D\sum_i N_i\delta(\vec{r}-\vec{r}_i)\Psi_l^m \\ &= -\frac{e}{\epsilon_0}\sum_i N_i\Psi_l^m(\vec{r}_i).\end{aligned} \tag{22}$$

In practice, since the basis functions Ψ_l^m have a compact support on mesh cells, only $(m+1)^n$ basis functions need to be evaluated for each macro particle i. The source term is therefore calculated by looping through all the macro particles and accumulating their contribution to the $(m+1)^n$ elements λ_l for which $\Psi_l^m(\vec{r}_i)$ is non-zero.

It must be noted that by itself the web-spline method still suffers from some limitations that characterize some meshless FEM. Indeed, it is not able to solve the Poisson equation on domains with discontinuous dielectric constants, such as at the boundary between a dielectric and a vacuum vessel. This can be mitigated by not solving the Poisson equation inside the dielectric, but by imposing a Neumann boundary condition at the dielectric surface. However, several methods have been devised to solve this problem such as the eXtended Finite Element Method (XFEM) [37], the Generalized Finite Element Method (GFEM) [38] or the Stabilized Generalized Finite Element Method (SGFEM) [39], which could be implemented in future revisions of FENNECS, should they be necessary.

3.4. Implementation of the web-splines method

As described in Section 3.3, the Poisson equation is effectively solved by using the linear system of equations

$$\overleftrightarrow{A}\cdot\vec{u} = \vec{\lambda}, \tag{23}$$

where the matrix and vectors are defined with the web-spline basis

$$\Psi_{l,web}^m = \frac{w}{w(x_l)}\left[b_l^m + \sum_{k\in K(l)} e_{lk}b_k^m\right]. \tag{24}$$

This basis is however not easy to manipulate numerically. This is the case both during the integration of the basis to generate the linear system of equations and during the evaluation of the solution. One easier method is to work first in a regular weighted b-spline (wb-spline) basis to calculate a matrix $\overleftrightarrow{A}'$ with coefficients

$$A'_{ij} = \int_D \nabla(wb_i^m)\nabla(wb_j^m), \tag{25}$$

and a right-hand side vector $\vec{\lambda}'$ with coefficients

$$\lambda'_i = \int_D Q w b_i^m - \nabla \tilde{g} \nabla (w b_i^m). \tag{26}$$

This can be calculated directly by looping on the grid cells and the b-spline functions, and using standard numerical integration methods (e.g. Gauss-Legendre quadrature integration). In a second step, all the b-splines are catalogued in the three inner, outer, and boundary splines and a transformation matrix from wb-spline to web-spline space $\overleftrightarrow{E}$ is calculated, with coefficients

$$E_{l,j} = \frac{1}{w(x_l)} \begin{cases} 1 & \text{for } j = l, \\ e_{l,k} & \text{for } j = k \in K(l), \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

With this definition the web-spline basis is equivalent to

$$\Psi_{l,web}^m = \sum_j w b_j^m E_{l,j}. \tag{28}$$

The web-spline linear system is then obtained from the wb-spline system through a basis transformation

$$\overleftrightarrow{A} = \overleftrightarrow{E} \overleftrightarrow{A}' \overleftrightarrow{E}^t, \tag{29}$$

and

$$\vec{\lambda} = \overleftrightarrow{E} \vec{\lambda}'. \tag{30}$$

Here, the superscript t indicates the transposed matrix. The system

$$\overleftrightarrow{A} \cdot \vec{u} = \vec{\lambda}, \tag{31}$$

is then solved and the wb-spline coefficient vector $\vec{u}'$ can be obtained through an inverse transformation

$$\vec{u}' = \overleftrightarrow{E}^t \vec{u}. \tag{32}$$

The final solution

$$u_h = \sum_i u'_i w b_i^m, \tag{33}$$

is also easier to evaluate numerically using standard b-spline libraries in the wb-spline basis than in the web-spline basis.

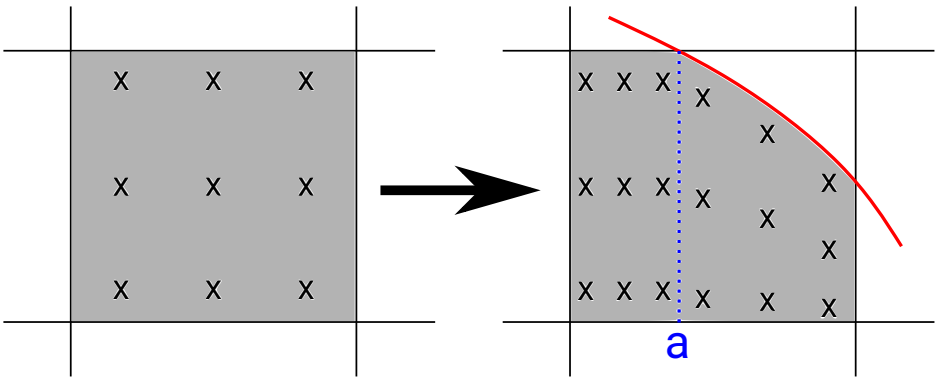

Figure 5: Left: integration on an inner cell using three Gauss-Legendre points in each direction. Right: integration on a boundary cell using three Gauss-Legendre points in each direction for each sub-rectangle. The red line shows the domain boundary and the blue dotted line shows the cell subdivision. The X indicate the integration points and the gray background highlights the intersection between the domain and the grid cell.

3.5. Numerical element integration for generating the Ritz-Galerkin matrix

The integration of the elements on all the inner cells is done using a standard numerical integration algorithm based on Gauss-Legendre quadrature due to its good precision and efficiency. However, in the boundary cells, the integration domain is not rectangular and the integration scheme needs to be modified. In FENNECS, it is assumed that the domain in each cell can be reduced to the union of a finite set of smoothly deformed rectangles, by cuts parallel to the coordinate directions and Gauss points with adapted positions are used in each sub-rectangle. This process is illustrated in Figure 5 where a boundary cell is subdivided by the dotted blue line between two smoothly deformed rectangles. In each deformed sub-rectangle, the Gauss points are first defined normally in the direction perpendicular to the cell subdivision (horizontal direction in Figure 5). In a second step, at each of these Gauss points, the position of the points parallel to the cut are defined by the boundaries of the domain in the sub-rectangle. To this end, a root finding algorithm based on the dichotomy method is used on the weight function w to obtain the boundary points of the domain (the position of the red curve in Figure 5) and set the extent of the integration segment parallel to the cut. This method allows a systematic evaluation of the element integrals while needing only local information on the cell, and allowing for a straightforward parallelisation of the construction of $\overleftrightarrow{A}$ and $\vec{\lambda}$.

3.6. Boundaries and boundary conditions definitions

With the web-spline method, Dirichlet boundary conditions are imposed by defining the weight function $w(x)$ and the value at the boundary is set with the function $\tilde{g}(x)$. There is a freedom in defining $w(x)$ and $\tilde{g}(x)$. Within this section, two main methods will be described to create these functions, that are both systematic and robust. Moreover, a 2D domain is considered, but this method can also be applied to 3D. The first method used to define w, that sets the simulation domain D, is based on analytical geometric functions w_i that define elementary geometric shapes. These functions can define, for example, a half-plane, a disc, or a square. One such geometric function w_1 can be the equation delimiting the inside of an ellipse of minor radius a, major radius b and center (x_0, y_0)

$$w_1(x, y) = 1 - \left(\frac{x - x_0}{a}\right)^2 - \left(\frac{y - y_0}{b}\right)^2. \tag{34}$$

These elementary weight functions can be combined by using Rvachev functions to define the union, intersection, or complementary of these elementary domains [40]. The union (+) and intersection (-) of two weights w_1 and w_2 can be calculated with

$$w(x, y) = w_1(x, y) + w_2(x, y) \pm \sqrt{w_1^2(x, y) + w_2^2(x, y)}. \tag{35}$$

Similarly, the complementary of a domain D is defined by taking the negative of the weight function. This method is used in Sec.4.1 to combine two elliptical domains of different major and minor radii and of different center. For this type of weight, the function $\tilde{g}$ used to impose the Dirichlet boundary conditions can be defined using transfinite interpolation of the potentials g_i imposed on n_i boundaries ∂D_i and defined using weights w_i [41]. In this case:

$$\tilde{g} = \sum_{i=1}^{n_i} g_i \frac{\prod_{j=1;j\neq i}^{n_i} w_j}{\sum_{k=1}^{n_i} \prod_{j=1;j\neq k}^{n_i} w_j}. \tag{36}$$

This function ensures that $\forall \vec{x} \in \partial D_i;\ g(\vec{x}) = g_i$. Furthermore, if all w_i are continuous, then $\tilde{g}$ is also continuous and that is needed for the stability and physicality of the solution.

The second method to define a boundary is to use spline curves defined using a set of control points, called "knots", in 2D or 3D. The total weight function w, induced by the boundaries ∂D_i can be computed, in this case, using a smoothed

distance function to the curves, blended with a plateau of value 1 inside the domain.

$$w = 1 - \sum_{i=1}^{n_i} \max(1 - d(x, y; \partial D_i)/d_0, 0)^3, \tag{37}$$

where $d(x, y; \partial D_i)$ is the shortest distance between the point (x, y) and a point on the boundary ∂D_i, and d_0 is the characteristic fall length from the plateau to 0. This variable d_0 is an input parameter and needs to be chosen such that the shortest distance between two boundaries is always greater than d_0 ($d(\partial D_i; \partial D_j) > d_0$). With this method, $w = 1$ on almost all D and $w < 1$ only at a distance $d < d_0$ of the boundaries. This is useful to ensure that at each position in the domain, only one distance function needs to be calculated. This limits the number of calls to the distance function, as this evaluation is numerically expensive and needs to be computed for each macro-particle. The cubic power in w ensures the C^2-continuity of the weight at the plateau boundary. For this type of boundary conditions, since at most one boundary affects the weight in any position, $\tilde{g}$ can be defined using

$$\tilde{g} = \sum_{i=1}^{n_i} g_i \max(1 - d(x, y; \partial D_i)/d_0, 0)^3. \tag{38}$$

Since by the choice of d_0, only one boundary ∂D_i can be at a distance smaller than d_0 anywhere, the relevant boundary i can be pre-computed in each grid cell during the initialization of the geometry and at most one distance must be computed anywhere. Furthermore, for points sufficiently inside the domain, $w = 1$ and $\tilde{g} = 0$, which is equivalent to the classic b-spline FEM. This greatly reduces the execution time of the code.

To impose natural boundary conditions, only the integration domain D needs to be adapted as the FEM ensures by construction $\nabla_{\perp k}\phi = 0$ on ∂D_k. To define D, the current solution is to consider another weight function $w_N(\vec{x}) = 0\ \forall \vec{x} \in \partial D_k$ which is used to calculate on each grid-cell, and in each direction, the integration boundaries of D, by finding the roots of w_N. In its current form, FENNECS permits only the definition of Neumann boundary conditions with $\nabla_{\perp k}\phi = 0$ on ∂D_k. However, the web-spline method is more general and the code could be modified with limited effort to include more general Neumann boundary conditions. For the case $\nabla_{\perp k}\phi = g_k$ on ∂D_k, this boundary condition can be imposed by adding a term $\sum_k \int_{\partial D_k} g_k \Psi_l^m$ to λ_l in equation (17).

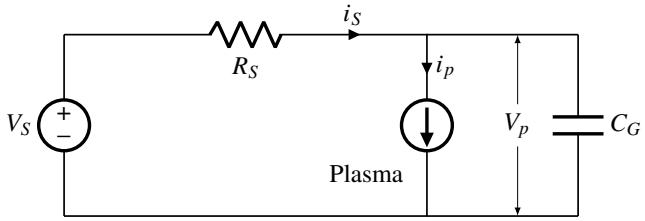

Figure 6: Circuit considered top simulate the effect of a non ideal power supply imposing the external bias between the electrodes.

3.7. Non-Ideal power supply

In addition to a constant bias imposed on each electrode of the domain, the code can simulate the effect of a non-ideal power supply (PS) imposing the confining biases between two selected electrodes. The PS and load circuit in this case is approximated by the circuit of Figure 6. The power supply is configured to impose a set voltage V_S supposed to be constant in time, and has an internal resistance R_S. The plasma cloud is described as a current source generating a current $i_p(t)$. In addition, the capacitive effects of the geometry and of the cables connecting the PS and the electrodes is simulated by a capacitor with capacitance C_G connected in parallel with the plasma cloud. This configuration imposes a bias V_p between the electrodes and is included in the code by changing in time the values g_j corresponding to the boundaries j connected to the PS. The ordinary differential equation of this circuit for the potential V_p is

$$\frac{\mathrm{d}V_p(t)}{\mathrm{d}t} = \frac{V_S - R_S i_p(t) - V_p(t)}{R_S C_G}. \tag{39}$$

The applied bias V_p is advanced in time using a 4th order Runge-Kutta method every N_s time-steps of the particle advance, by assuming that $i_p(t)$ is only function of time. Here, N_s is an input parameter. The plasma current $i_p(t)$ is measured every $N_s/2$ time steps, by accumulating the charge $q(t_i)$ collected at each time step t_i on the electrodes, giving:

$$i_p(t_{i+N_s/2}) = \frac{1}{\Delta t N_s/2} \sum_{j=i}^{i+N_s/2} q(t_j). \tag{40}$$

Here, Δt is the numerical timestep used to evolve the particles position. Since the PS has a fixed characteristic response time, independent on p_n, and due to

the artificial reduction of the time-scale separation defined by the parameter α, the time response of the PS must be rescaled in the simulation. Similarly, as the collected current is linearly proportional to the neutral gas density [9, 26], i_p must be adapted in the simulations. This is particularly important if simulations need to be run for experimental cases in high and ultra high vacuum. To this end the value of R_S is rescaled in the simulations defining $R_{S,\text{simu}} = R_{S,\text{phys}}/\alpha$. This has the effect of rescaling both the power supply time-scales $\tau_{PS} = R_S C_G$, and is equivalent to rescaling the numerical current $i_{p,\text{simu}}(t)$ by $1/\alpha$ due to the term $R_S i_p(t)$ in the differential equation.

3.8. Electron-neutral collisions

The electron-neutral collisions are simulated using a standard Monte Carlo approach [42]. For this process, each macro-particle is temporarily treated as a single particle of the simulated species. At each time step, the collision cross-sections σ_{io} and σ_{ela} are evaluated, from tabulated data [28, 43], for each type of interaction and each macro-particle. For each particle i, a random number $x_{i,1} \in [0, 1]$ is generated according to a uniform distribution function and is used to determine if the particle i of kinetic energy E_i and velocity v_i undergoes a collision event.

1. If $x_{i,1} < 1 - \exp(-n_n(\sigma_{\text{io}}(E_i) + \sigma_{\text{ela}}(E_i))v_i \Delta t)$, with Δt the time step, a collision is triggered.
2. The type of collision is determined using a new random variable $x_{i,2}$. If

$$x_{i,2} < \frac{\sigma_{\text{ela}}(E_i)}{\sigma_{\text{io}}(E_i) + \sigma_{\text{ela}}(E_i)}, \tag{41}$$

   an elastic collision is triggered, otherwise an ionization event takes place.
3. In case of an elastic event, the first scattering angle χ is calculated using a singly differential cross-section for screened Coulomb Collision [29, 44], using a third random number $x_{i,3}$, according to

$$\cos(\chi) = 1 - \frac{2x_{i,3}(1-\xi)}{1+\xi(1-2x_{i,3})}. \tag{42}$$

   Here, $\xi = 4E_i/(E_h + 4E_i)$ and $E_h = \hbar^2/(m_e a_0^2) = 27.21\text{eV}$ is one Hartree, the atomic unit of energy, with $\hbar$ the reduced Planck constant and a_0 the Bohr radius. The second scattering angle $\theta = 2\pi x_{i,4}$ is obtained with a fourth random number. The electron velocity is then rotated using χ and θ.

4. In case of an ionization event, the energy splitting between the two resulting electrons is determined using a normalized differential cross-section obtained from experimental data [30] with a random number $x_{i,5}$. The scattered electron kinetic energy E_{sca} is

$$E_{\mathrm{sca}} = E^* \tan\big(x_{i,5} \arctan((E_i - E_{\mathrm{io}})/(2E^*))\big), \tag{43}$$

   with E^* a fitted scattering factor that depends on the neutral gas [30] and E_{io} the ionization energy of the gas. A new macro-particle is created in the simulation at the position of the particle i and with velocity v_i rescaled such that the kinetic energy of the new particle E_{sec} ensures energy conservation $E_{\mathrm{sec}} = E_i - E_{\mathrm{io}} - E_{\mathrm{sca}}$. The kinetic energy of the incoming particle is then also rescaled to E_{sca}.
5. Both scattered and freed electrons undergo a scattering event using the same differential cross-section as for the elastic collision. The same procedure as in point 3 is used for both electrons.

Depending on the physical system being simulated, the generated ion can either be added to a second species and be tracked in the simulation or not simulated. Ignoring the generated ions is justified in cases where the ions are lost rapidly and where ion induced electron emission happens in regions devoid of trapping mechanism. This rapid loss happen when the ions Larmor radius is larger than the dimensions of the vacuum vessel, which is typically the case in gyrotron electron gun simulations.

3.9. Ion induced emission

In simulations with large electrode bias (above 5kV), the ions generated due to an ionization event are accelerated toward the electrodes and gain large energies, e.g. of the order of several keV. In this regime, their collision with the electrodes can cause ion induced electron emission (IIEE). This process is currently simulated in the code for several types of metallic surfaces and Hydrogen ions. To calculate the electronic yield $\gamma(E_i)$, defined as the average number or electrons released by one impinging ion, two collision regimes are considered depending on the ion kinetic energy E_i. At low kinetic energies, $E_i < 1\,\mathrm{keV}$, the electrons of the metal are extracted due to the potential energy of the incoming ions [45]. In this case, the yield depends on the ionization energy E_{io} necessary to form the incoming ion, the Fermi energy ϵ_F and the work function Φ of the metal, but is independent of the ion kinetic energy,

$$\gamma_{pot} \approx \frac{0.2}{\epsilon_F}(0.8E_{\mathrm{io}} - 2\Phi). \tag{44}$$

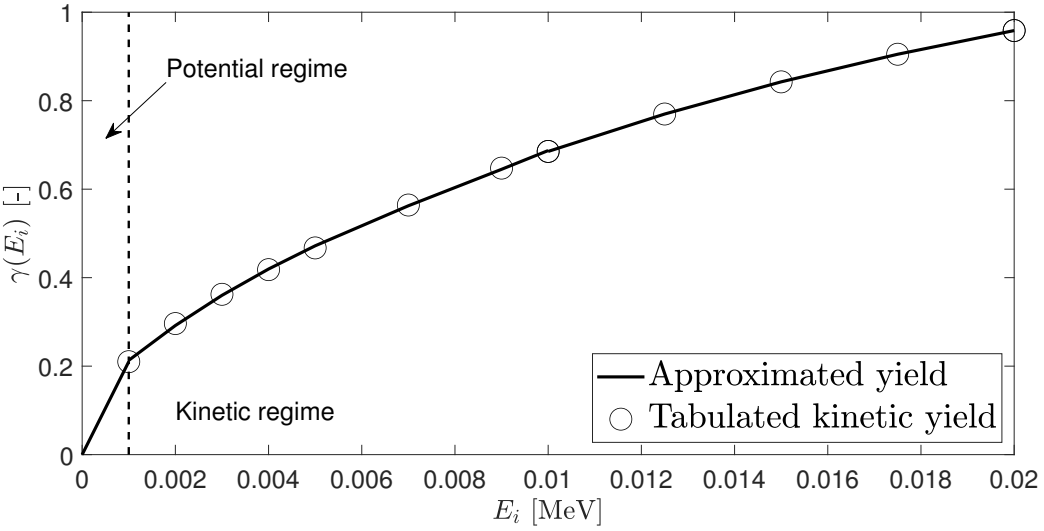

Figure 7: IIEE yield used in FENNECS for Hydrogen atoms hitting aluminum electrodes as a function of the incoming ion kinetic energy. The circles show the effective yield calculated using (45) and tabulated inelastic stopping power [48, 49]. For this electrode material $\gamma_{pot} = 0$.

At large kinetic energies $E_i > 1\,\mathrm{keV}$, the energy used to free the metallic electron comes from the kinetic energy of the ion and the yield is calculated using a model derived by Schou [46]

$$\gamma_{kin}(E_i) = \Lambda\beta \left(\frac{\mathrm{d}E}{\mathrm{d}x}\right)_e . \tag{45}$$

Here, Λ is a material constant, β is a coefficient that accounts for energy transport by recoiling electrons and by backscattered ions, and $\left(\frac{\mathrm{d}E}{\mathrm{d}x}\right)_e$ is the inelastic stopping power of the impacting ion which depends on the ion type and kinetic energy, and on the material type. As the code currently only considers IIEE by Hydrogen atoms, an experimental parameter $\Lambda\beta = \Lambda_{exp} \approx 0.1\,\text{Å}\,\mathrm{eV}^{-1}$ is used, which is independent on the metal type [46, 47]. The inelastic stopping power is taken from tabulated data based on experimental data and theoretical predictions [48, 49]. As both types of collision events can happen in the same simulation, the two yields are blended in the range $0 \leq E_i \leq 1\,\mathrm{keV}$ using a linear interpolation between γ_{pot} and $\gamma_{kin}(1\,\mathrm{keV})$ as seen in Figure 7.

To simulate IIEE in the code, the ions generated by electron-neutral collisions are simulated and tracked. Once an ion reaches an electrode, its kinetic energy E_i and yield $\gamma(E_i)$ are calculated. To determine the number of macro-electrons generated, a random number k is generated according to a Poisson distribution

with mean $< k >= \gamma(E_i)$ such that the probability of freeing k electrons is

$$P(k) = \frac{\exp(-\gamma(E_i))}{k!}. \tag{46}$$

k macro-electrons are then generated at the last known position of the ion inside the domain. The k resulting electrons are given a velocity normal to the electrode surface. To simulate the emitted electron energy spectra [46], the kinetic energy of the generated electrons E_e is given using a gamma distribution function which best fits experimental measurements [50]. We recall here the probability density function of the gamma distribution with parameters κ and θ and mean $< x >= \kappa\theta$:

$$f_\gamma(x) = \frac{1}{\Gamma(\kappa)\theta^\kappa} x^{\kappa-1} \exp\left(-\frac{x}{\theta}\right). \tag{47}$$

Here $\Gamma(\kappa)$ is the gamma function

$$\Gamma(\kappa) = \int_0^\infty t^{\kappa-1} e^{-t} dt. \tag{48}$$

As a first approximation, the parameters $\kappa = 4$ and $\theta = 0.5$ of the distribution function are independent of the electrode material and impose a mean kinetic energy of the generated electrons $< E_e >= 2\,\mathrm{eV}$. This approximation is supported by the fact that, in gyrotron gun simulations, large electric fields are externally imposed, and the initial electron kinetic energy becomes negligible compared to the one gained by electric field acceleration.

3.10. Seed sources

In addition to the electron sources resulting from the ion-induced electron emission and the impact ionization of the neutrals, a volumetric seed source is implemented in the code. This source can generate electrons in a fixed volume according to various types of distribution functions in velocity. For example, a Maxwellian distribution function with temperature T or a mono-velocity beam can be used. The amplitude and spacial distribution of the source is also an input parameter. This source can be used to simulate the effect of neutral ionization due to background radiation, or to simulate the effect of field-emissions on the electrodes. It is sometimes necessary to ensure that some electrons are present at all times in the simulation domain and can start the cloud formation cascade [9]. Indeed, without this source, the initial electron cloud population might be entirely lost due to electron-neutral friction drifts and no new electron cloud could be generated.

3.11. Post-processing

In addition to the Fortran PIC code, FENNECS is also bundled with a set of post-processing routines, implemented in MATLAB, to allow the extraction of several physical quantities relevant to the problem at hand. These routines allow the loading and easy manipulation of the raw simulation data. They provide an abstraction layer that reads HDF5 simulation result files, and present to the user standard MATLAB structures and classes, which facilitates the manipulation of the data and reduces the complexity of figures generation. For example, a set of graphical user interface routines have been created to display dynamically the time evolution of the electron density, the fluid velocities, the electrostatic potential and electric field, or the pressure tensor.

The raw data is saved at user-defined regular intervals. This can either be by saving the individual particles physical quantities for studying the full distribution of the simulated species, or by calculating at each grid-point the 0^{th},

$$n(\vec{r},t) = \int_{\mathbb{R}^3} f(\vec{r},\vec{v},t)d^3v, \tag{49}$$

first,

$$n\vec{u}(\vec{r},t) = \int_{\mathbb{R}^3} \vec{v} f(\vec{r},\vec{v},t)d^3v, \tag{50}$$

and second order,

$$nE_{ij}(\vec{r},t) = \int_{\mathbb{R}^3} \frac{m_e}{2} v_i v_j f(\vec{r},\vec{v},t)d^3v \tag{51}$$

moments of the distribution function in each direction. From these moments, the density and fluid velocity are directly accessible and evaluated at discrete time-steps. The pressure tensor can be calculated with

$$\mathcal{P}_{ij} = \int_{\mathbb{R}^3} m_e(v_i - u_i)(v_j - u_j) f(\vec{r},\vec{v},t)d^3v = n\left(-m_e u_i u_j + 2E_{ij}\right). \tag{52}$$

This information is then accessible in the MATLAB class representing the simulation result.

4. Verifications

To verify the correct implementation of the code, a set of test cases have been run and are presented in this section.

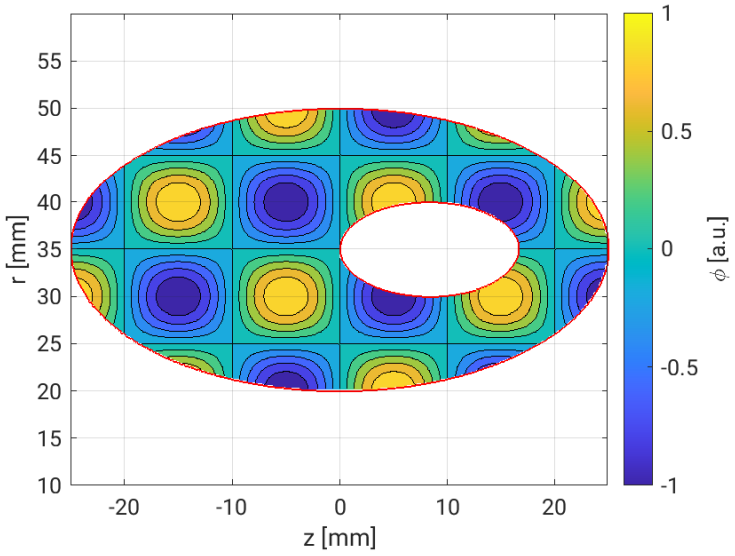

Figure 8: Domain and manufactured solution of the potential used to verify the implementation of the FEM solver.

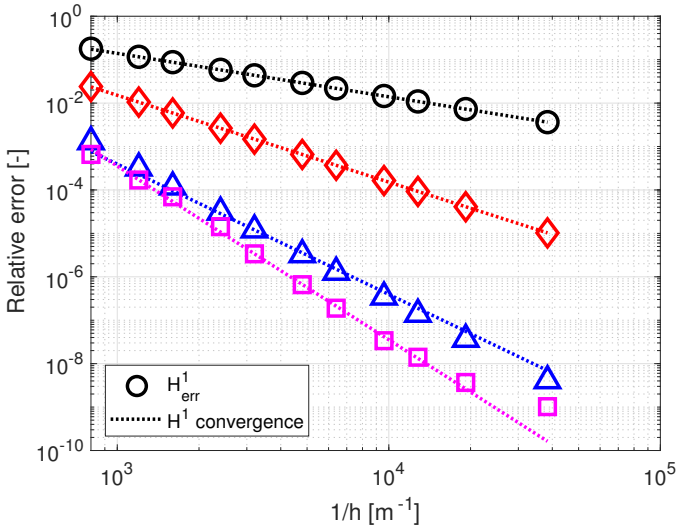

Figure 9: H1 norm of the relative error for spline orders from 2 to 5 ($\circ, \diamond, \triangle, \square$) and varying grid width h. The dotted lines highlight the ideal convergence for each spline order. Each order is color coded (black, red, blue, magenta) for readability.

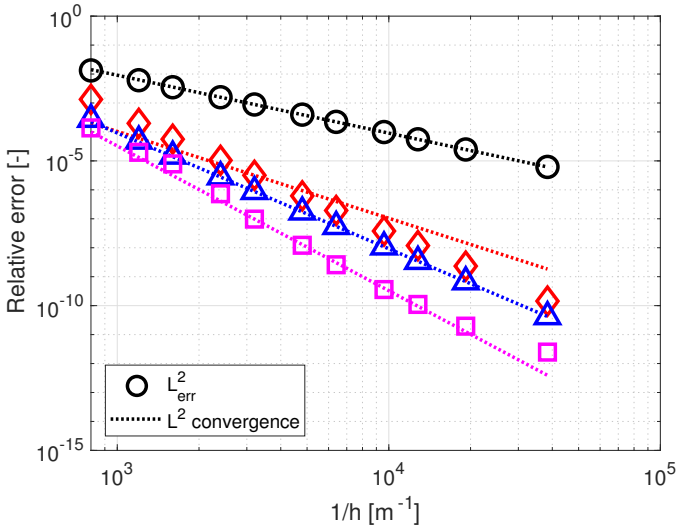

Figure 10: L2 norm of the relative error for spline orders from 2 to 5 ($\circ, \diamond, \triangle, \square$) and varying grid width h. The dotted lines highlight the ideal convergence for each spline order. Each order is color coded (black, red, blue, magenta) for readability.

4.1. General complex geometry Poisson solver

The Poisson solver is verified using a manufactured solution of the form

$$\phi = \sin\left(\pi \frac{z - z_0}{L_z}\right) \sin\left(\pi \frac{r - r_0}{L_r}\right), \tag{53}$$

which satisfies the Poisson equation with the source term:

$$\begin{aligned} Q = &\left(\frac{\pi}{L_z}\right)^2 \sin\left(\pi \frac{z - z_0}{L_z}\right) \sin\left(\pi \frac{r - r_0}{L_r}\right) \\ &+ \frac{\pi}{L_r} \sin\left(\pi \frac{z - z_0}{L_z}\right) \left[-\frac{1}{r} \cos\left(\pi \frac{r - r_0}{L_r}\right)\right. \\ &\left. + \frac{\pi}{L_r} \sin\left(\pi \frac{r - r_0}{L_r}\right)\right], \end{aligned} \tag{54}$$

on a domain defined using the Rvachev intersection of two domains defined with ellipses, as represented in Figure 8. The spline grid is defined with $-25 \leq z \leq 25$ mm and $10 \leq r \leq 60$ mm and for an increasing number of grid cells per dimension, from 20 to 960. The manufactured solution parameters are $r_0 = 35\,\text{mm}$,

$z_0 = 0\,\text{mm}$, $L_r = 10\,\text{mm}$ and $L_z = 10\,\text{mm}$. The inner ellipse is defined with the geometric weight

$$w_1 = \left(\frac{r-35}{8.3}\right)^2 - \left(\frac{z-8.3}{8.3}\right)^2 - 1, \tag{55}$$

and the outer ellipse is defined with

$$w_2 = 1 - \left(\frac{r-35}{15}\right)^2 - \left(\frac{z}{25}\right)^2. \tag{56}$$

Here r and z are defined in mm. The error in the solution is evaluated using both the L^2 and the H^1 norms defined respectively as

$$e_2 = ||\phi_h - \phi||_{L^2} = \left(\int_D (\phi_h - \phi)^2 dV\right)^{1/2}, \tag{57}$$

and

$$e_1 = ||\phi_h - \phi||_{H^1} = \left(\int_D (\phi_h - \phi)^2 + |\nabla(\phi_h - \phi)|^2 dV\right)^{1/2}. \tag{58}$$

It can be shown [35] that $e_1 \sim O(h^{m-1})$ as $h \to 0$ when using web-splines of order m. Similarly, it can be shown that $e_2 \sim O(h^m)$ as $h \to 0$ when using web-splines of order m. Evaluations of the relative error using the H^1 and L^2 norms are represented in Figure 9 and Figure 10 using the manufactured solution previously defined. We observe that the correct convergence rates are recovered for different orders of the web-spline basis.The error calculated using the maximum norm also called L^∞ shows the same convergence order and one order of magnitude higher relative error than obtained with the L^2 norm.

The same manufactured solution is also used in a geometry defined by a set of straight lines forming a 4 branch star pattern to show the robustness of the method in simulating sharp edges (see Figure 11). The weight function w for this geometry, is defined using a smoothed distance function (see equation (37)) with parameter $d_0 = 0.01\text{m}$. In Figure 11, the regions of low and high relative error in this geometry, show that convex edges are well resolved by the method, while concave edges show less accuracy. This is probably due to the reduced number of free parameters at these locations and to the loss of regularity of the weight function w. The L^2 norm of the relative error for this configuration, represented in Figure 12, shows that the convergence order of the method is lost for degrees $m > 2$ for reasons that are not yet clear. However, the method still converges with a reduced order. We suspect that this effect is due to a loss of regularity of the weight function w, but further studies will be necessary to address this effect.

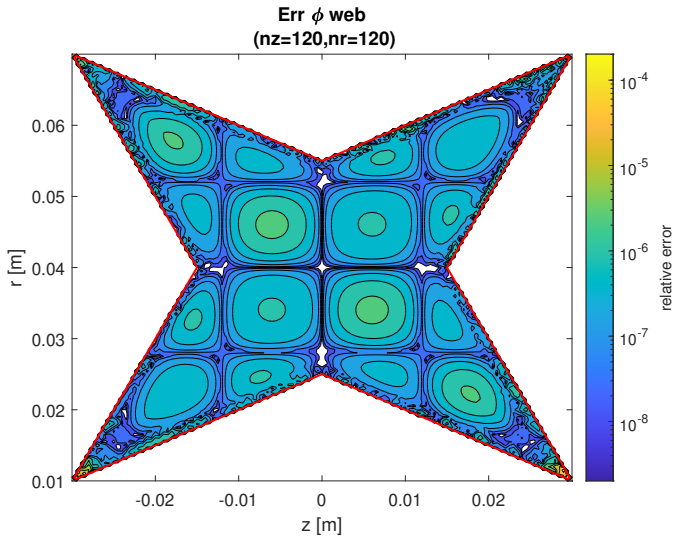

Figure 11: Domain and relative error on the potential for the second geometry used to verify the implementation of the FEM solver. This result was obtained for a resolution $n_r = n_z = 120$ and b-splines of degree $m = 2$. The error is larger at the convex edges and show that the concave edges are well resolved. The white regions inside the domain show a relative error below 10^{-9}.

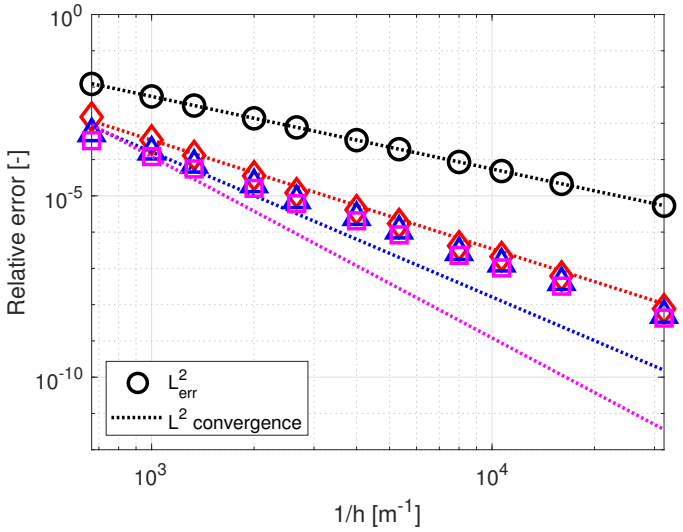

Figure 12: L^2 norm of the relative error for spline orders from 2 to 5 ($\circ, \diamond, \triangle, \square$) and varying grid width h for the star shaped domain. The dotted lines highlight the ideal convergence for each spline order. Each order is color coded (black, red, blue, magenta) for readability.

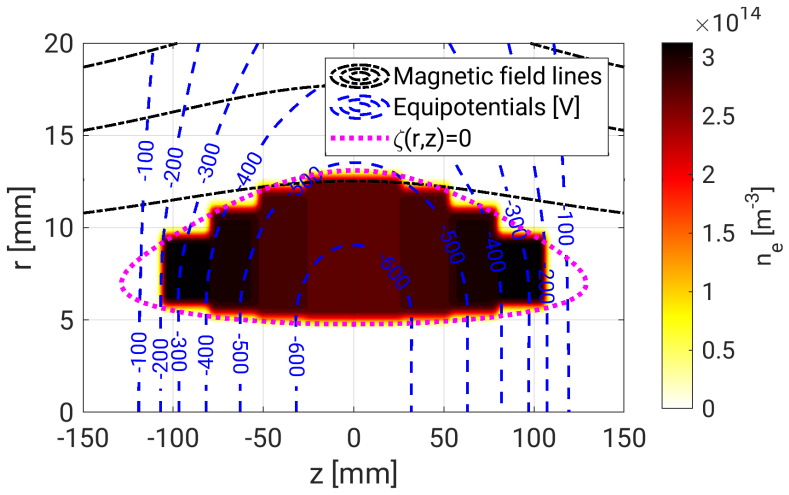

Figure 13: Zoom on the initial density loading of the electron cloud and steady-state $\zeta = 0$ contour (dotted magenta). In addition, the electric equipotential (dashed blue) and magnetic field lines (dash-dotted black) are represented.

4.2. Equilibrium of a ring of charges trapped in a magnetic mirror

In this section we present the results of simulations of a pure electron plasma equilibrium for which an analytical solution exists. This particular Vlasov electrostatic equilibrium considers an annular electron cloud trapped radially by a strong magnetic field and trapped axially by a magnetic mirror of length L [2, 51]. The magnetic vector potential for an externally imposed mirror field is described analytically by

$$A_0^{\text{ext}}(r,z) = \frac{1}{2}B_0\left[r - \left(\frac{L}{\pi}\frac{R-1}{R+1}\right)I_1\left(\frac{2\pi r}{L}\right)\cos\left(\frac{2\pi z}{L}\right)\right], \tag{59}$$

with B_0 the magnetic field amplitude on axis at $z = \pm L/4$, I_1 the modified Bessel function of order one, L the distance between the mirror coils, and R the mirror ratio defined by $R \equiv B_{max}/B_{min} = B_0^{\text{ext}}(r=0, z=\pm L/2)/B_0^{\text{ext}}(r=0, z=0)$. The distribution function is written in terms of conserved quantities, namely the total energy $H = p^2/(2m_e) - e\phi(r,z)$, with $\vec{p} = m_e\vec{v}$, and the canonical angular momentum $P_\theta = r\left[p_\theta - eA_0^{\text{ext}}(r,z)\right]$,

$$f_e(H, P_\theta) = \frac{n_0 R_0}{2\pi m_e}\delta(H - H_0)\delta(P_\theta - P_0). \tag{60}$$

Here, n_0 is the maximum electron density in the cloud, R_0 is the lower radial limit of the cloud at $z = 0$, H_0 and P_0 are positive constants, and p_θ is the electron

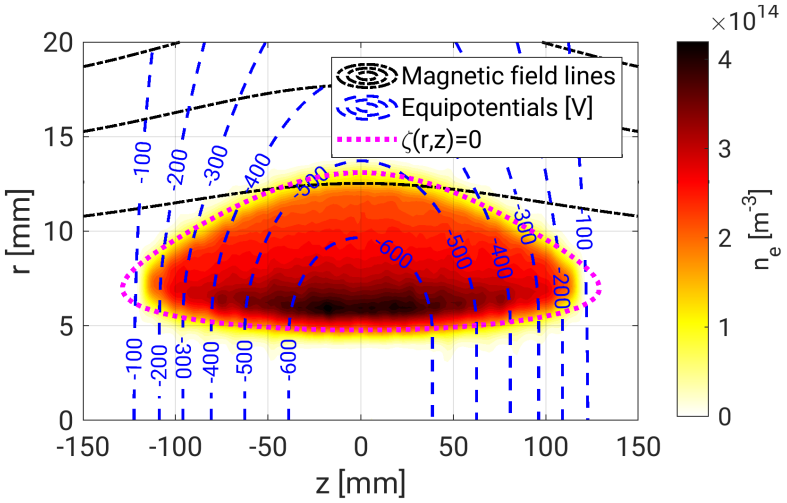

Figure 14: Zoom on the final density of the electron cloud and steady-state $\zeta = 0$ contour (dotted magenta). In addition, the electric equipotential (dashed blue) and magnetic field lines (dash-dotted black) are represented.

momentum in the azimuthal direction. Equation (60) is a solution of the Vlasov equation. For this equilibrium, one can show that an envelope function $\zeta(r,z)$ can be defined [51]:

$$\zeta(r,z) = \frac{p_\perp^2(r,z)}{2m_e H_0} = 1 + \frac{e\phi}{H_0} - \frac{1}{2m_e H_0}\left[\frac{P_0}{r} + eA_0^{\text{ext}}\right]^2, \tag{61}$$

such that the curve where $\zeta(r,z) = 0$ denotes the limit of the electron cloud. Here, $p_\perp$ is the momentum perpendicular to $\hat{e}_\theta$ and ϕ is the self-consistent electric potential. The electron density is therefore:

$$n_e(r,z) = \frac{R_0}{r} n_0 U[\zeta(r,z)], \tag{62}$$

with $U[x]$ the Heaviside step function. To verify the code implementation, a cloud of electrons is loaded in FENNECS with $H_0 = 3.2\times10^{-14}\,\text{J}$ and $P_0 = 8.66\times10^{-26}\,\text{kg m}^2\,\text{s}^{-1}$ in the region where $\zeta_0 = 1 - \frac{1}{2m_e H_0}\left[\frac{P_0}{r} + eA_0^{\text{ext}}\right]^2 > 0$ using a uniform density distribution function as seen in Figure 13. The number of loaded macro particles is 2116800 with a macro particle weight $w_p = 1.018\cdot10^4$ such that the reference density in steady state is $n_0 = 5\times10^{14}\,\text{m}^{-3}$. The simulation domain is defined with a cylindrical volume of radius r_b and length L_z as shown in

B_0	$0.21\,\mathrm{T}$
R	1.5
$L = L_z$	$0.48\,\mathrm{m}$
r_b	$0.06\,\mathrm{m}$
Δz	$1.9\times10^{-3}\,\mathrm{m}$
Δr	$2.8\times10^{-4}\,\mathrm{m}$
Δt	$5\times10^{-12}\,\mathrm{s} \approx 0.2 m_e/eB_0$
n_0	$5\times10^{14}\,\mathrm{m}^{-3}$
R_0	$0.005\,\mathrm{m}$
H_0	$3.2\times10^{-14}\,\mathrm{J}$
P_0	$8.66\times10^{-26}\,\mathrm{kg\,m^2\,s^{-1}}$

Table 1: Physical and numerical parameters used in the simulation of the annular electron cloud trapped in a magnetic mirror.

Figure 13. The volume is enclosed radially by a cylindrical conductor of radius r_b at ground. The system and numerical parameters are given in Table 1.

The simulation is run with a timestep $\Delta t = 5\times10^{-12}\,\mathrm{s} \approx 0.2/\Omega_{ce}$ and the total kinetic and potential energies of the cloud are monitored. As shown in Figure 15, the total energy summed over all particles E_{tot} is conserved on average and the relative error on the energy is kept below 10^{-4} over long time scales as compared to the bounce time of electrons. During the simulation no particles are lost, and after a time $t = 364\,\mathrm{ns} \approx 460/\omega_{pe}$, the system is in a steady state and successfully retrieves the analytical solution of the equilibrium. In particular, the $1/r$ radial dependency of the electron cloud density, as predicted by equation (62), is well reproduced and the cloud remains limited by the envelope function as can be observed in Figure 14 and Figure 16.

4.3. Radial drifts due to collisions

To verify the electron neutral collision implementation, an annular electron cloud is considered in a coaxial configuration of infinite length, subjected to a uniform axial magnetic field, and to elastic collisions with a residual neutral gas. We assume that the distribution remains close to isotropic and study the system using a fluid model [9]. In this model the force balance equation is

$$n_e m_e \frac{\mathrm{d}\vec{u}}{\mathrm{d}t} = -n_e e\left(\vec{E} + \vec{u}\times\vec{B}\right) - n_e m_e \nu_{\mathrm{ela,mom}}\vec{u} - \nabla P. \tag{63}$$

Here, $\vec{u}$ is the fluid velocity of the electrons, n_e is the local electron density, $\nu_{\mathrm{ela,mom}} = n_n < \int \frac{\mathrm{d}\sigma_{\mathrm{ela}}}{\mathrm{d}\Omega} v d\Omega >_f$ is the averaged collision frequency for momentum

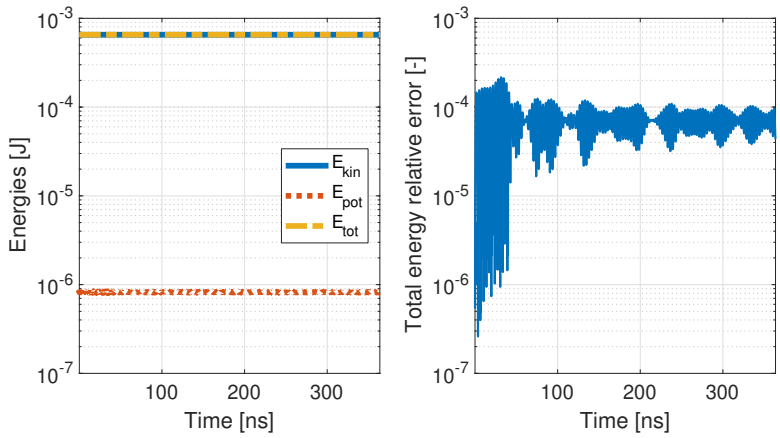

Figure 15: Left: Time evolution of the total (yellow), kinetic (blue) and potential (red) energies during the simulation. Right: Relative error on the total energy conservation during the simulation.

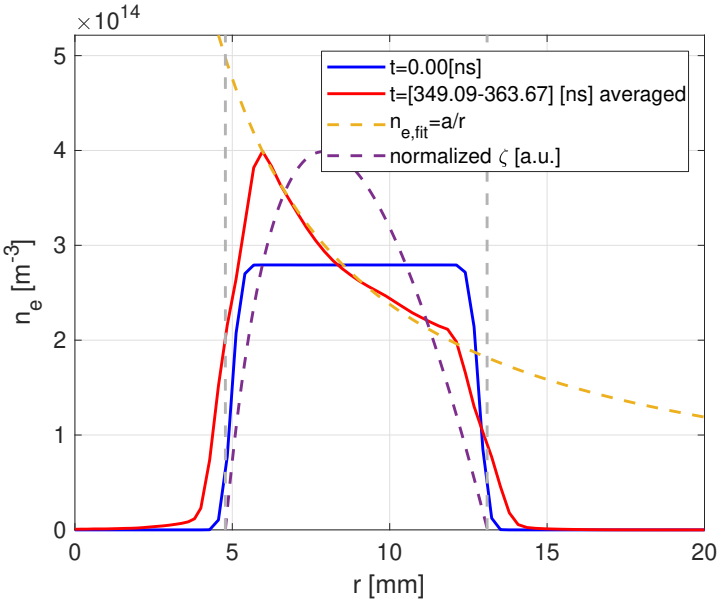

Figure 16: Initial (solid blue) and steady-state (solid red) radial density profile at $z = 0$, $1/r$ fit of the steady-state density profile (dashed yellow) and normalized ζ (dashed purple) at the same axial position. The vertical gray dashed lines highlight the radial positions for which $\zeta = 0$.

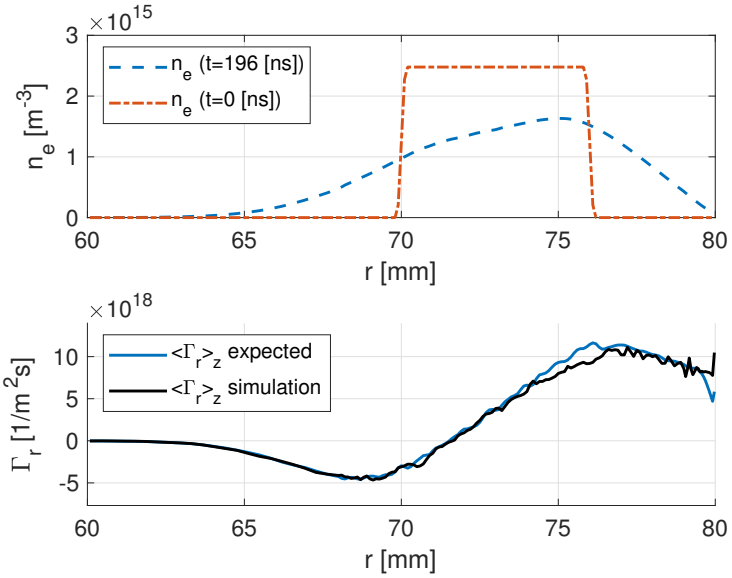

Figure 17: Top: initial and final radial electron density profile. Bottom: Expected (blue) and simulated (black) radial flux at the end of the simulation.

exchange, $<>_f$ denotes the average over the distribution function, and P is the pressure. Due to the azimuthal symmetry considered, the pressure term is identically 0 in the azimuthal direction. Neglecting all inertial terms, equation (63) in the azimuthal direction gives the radial electron flux $\Gamma_r(r) = n_e u_r$ caused by the collisional drag:

$$\Gamma_r(r) = -n_e \frac{u_\theta \nu_{\text{ela,mom}}}{\Omega_c}. \tag{64}$$

In the simulations, the system is defined with a coaxial configuration of infinite length, using periodic boundary conditions for the particles and natural boundary conditions for the fields. A bias $\Delta\phi = 300\,\text{V}$ is applied between the electrodes and the system is also subjected to a uniform external magnetic field of amplitude $B_0 = 0.28\,\text{T}$. In this configuration, a cloud of uniform density $n_e^0 = 2.5 \times 10^{15}\,\text{m}^{-3}$ and uniform velocity, as represented in Figure 17 is loaded on an annulus between $r = 70\,\text{mm}$ and $r = 76\,\text{mm}$. The simulation is set with periodic axial boundary conditions for the particles to simulate the infinite length and is run for several elastic collision characteristic times. The simulation results show indeed that the distribution function remains isotropic and that the inertial terms are small compared to the Lorentz and collisional drag terms. The radial flux is measured in the code and compared to the prediction obtained using the fluid model (64). The fluid az-

imuthal velocity is extracted from the code, and the averaged collision frequency is evaluated with the electron distribution function extracted from the simulation results. Both the expected and simulated fluxes are averaged along the axial direction to reduce the numerical noise. As shown in Figure 17, both fluxes exhibit the same behavior throughout the simulation domain with a maximum relative error of 20% at $r = 76\,\mathrm{mm}$.

4.4. Ion induced emission

The IIEE module is verified with a coaxial configuration with an inner electrode at $r_a = 1\,\mathrm{mm}$ and an outer electrode at $r_b = 10\,\mathrm{mm}$ with an axial uniform magnetic field of amplitude $B_0 = 0.21\,\mathrm{T}$. A bias $\Delta\phi = 20\,\mathrm{kV}$ is imposed between the electrodes and three clouds of 1000 Hydrogen ions each are loaded at different radial positions $r_1 = 3\,\mathrm{mm}$, $r_2 = 5\,\mathrm{mm}$ and $r_3 = 8\,\mathrm{mm}$ with zero velocity. In this configuration, the ions are accelerated radially towards the central electrode, gaining a kinetic energy $E_{kin}(r_j) \equiv E_j = e(\phi(r_j) - \phi(r_a))$ that depends on their initial radial position r_j. The effective yield obtained in the simulation is then compared to the theoretical yield $\gamma_{kin}(E_j)$ for each initial position r_j. The results of these simulations are given for three electrode materials in Table 2, and show an agreement between the theoretical yields γ_{exp} and simulation yields γ_{sim} within 2%.

Material	^{304}SS	r_1	r_2	r_3
	γ_{exp}	1.311	1.623	1.870
	γ_{sim}	1.299	1.627	1.891
	Rel. error	0.9%	0.2%	1.1%
Material	Cu	r_1	r_2	r_3
	γ_{exp}	1.237	1.522	1.746
	γ_{sim}	1.229	1.518	1.760
	Rel. error	0.6%	0.3%	0.8%
Material	Al	r_1	r_2	r_3
	γ_{exp}	0.920	1.133	1.297
	γ_{sim}	0.910	1.115	1.293
	Rel. error	1.0%	1.6%	0.3%

Table 2: Expected and simulation yield obtained for the IIEE verification case considering: stainless steel 304, copper and aluminum as electrode material, and 3 initial radial position of the ions.

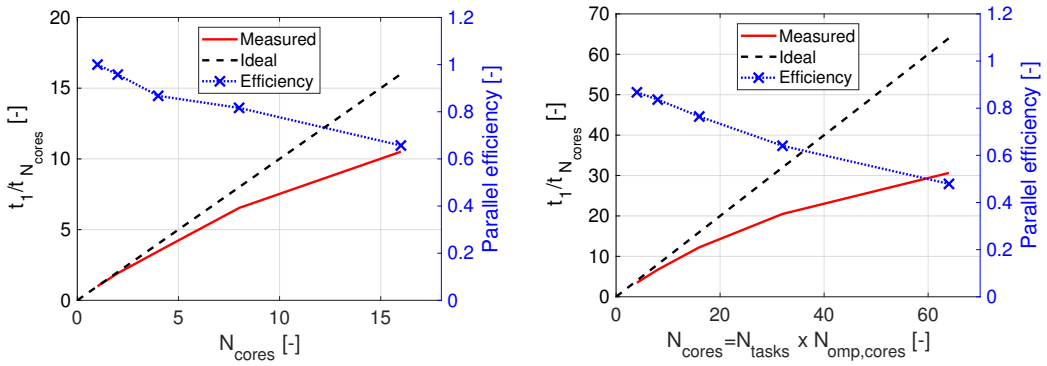

Figure 18: Strong scaling and parallel efficiency of FENNECS for a case considering 10M macro-particles in a coaxial geometry. Left: scaling using only OpenMP parallelism. Right: Scaling using MPI and OpenMP parallelism with each task set to use 4 OpenMP threads.

5. Parallel performance

The code is parallelized using a hybrid OpenMP/MPI approach. Indeed, the code is first parallelized using MPI and a non-uniform axial domain decomposition. This choice is supported by the fact that the particle distribution is usually not uniform axially, as illustrated by the test case described in Sec.4.2. Thus, to ensure a good load balancing for the computation of the particle trajectories between each task, the axial length covered by each task must be different. On each node, OpenMP parallelism is used to parallelize the computation of the particle trajectories. To avoid the use of atomic operations, the RHS of the Poisson equation is duplicated on each OpenMP thread and reduced to a single array before communications and the computation of ϕ. The OpenMP parallelism is limited by the wait times during particles creation and deletion in the particles structure (bound, boundary_loss and neutcol_step routines), which necessitate serial operations to avoid holes in the structure, and any overwrite of existing particles. These waits already take $\sim 10\%$ of the time for 16 cores, but could probably be reduced with clever parallelisation of these routines. The Poisson solver is currently implemented using the MUMPS library in serial mode, but the solver could, in principle, be adapted to leverage the OpenMP and MPI parallelism. This was motivated by the fast solution of the Poisson equation due to the problem size, even using a direct solver. Indeed in serial, the solving step takes less than 1% of runtime. However, in MPI parallelism, the RHS of the Poisson equation must be gathered on the host node and broadcasted to all the nodes at each time-steps which leads to significant portion of the code spent in waiting for the communica-

tions to be finished. As this communication time increases linearly with the number of nodes, the parallelisation scheme used in FENNECS becomes no longer sustainable above 8 nodes where communications already take 35% of the run time. There is still room for improvement of the parallelisation by leveraging the full potential of the MUMPS solver and reducing the communication time. Similarly, the OpenMP parallelism could also be improved by rearranging the order of the computations for each particle, thus reducing the need for OpenMP barriers. Given these facts, the run-time of FENNECS is acceptable for the configurations currently simulated. Nevertheless, with the future planned extension of FENNECS to 3D geometries, the added numerical and communication costs of the larger problems will surely necessitate the implementation of a parallel Poisson solver. This solver should strongly reduce the data communication and improve the overall scalability of FENNECS.

Strong scaling studies have been run on the jed cluster of EPFL [52] which is composed of nodes with 2 Intel(R) Xeon(R) Platinum 8360Y leading to 72 cores per node. The scalling studies were run using a pure OpenMP parallelism with up to 16 cores and a hybrid parallelism with up to 16 tasks and 4 threads per task as shown in Figure 18. For these studies, we considered a coaxial geometry in which 10 million particles are simulated on a (150x192) grid. The strong scaling results show a parallel efficiency above 75% for up to 4 tasks with each 4 OpenMP threads corresponding to a total of 16 CPUs. The parallel efficiency η_{par} is defined as the ratio between the time t_1 to run the task with a given number of cores, and N times t_N the time needed to run when N times more cores are used ($\eta_{\text{par}} = t_1/(Nt_N)$).

6. Summary and conclusions

The present paper describes the code FENNECS solving the Boltzmann-Poisson equations for a non-neutral plasma subject to strong electric fields generated by electrodes of complex shapes and space-charge effects, and to strong external magnetic fields. The novel numerical method used to simulate the complex electrode geometries present in gyrotron electron guns is also described. The different electron sources considered in the code are presented. More specifically, the Monte Carlo method used to simulate the electron-neutral collisions (elastic and ionization), and the emission of electrons at the electrode surfaces due to collisions of energetic ions on the electrode surfaces are also described. A set of verification cases are presented and confirm the correct implementation of the governing equations. In addition, the cases underline the efficiency of the novel

web-spline method and demonstrate its capability of reaching arbitrary precision for the electrostatic potential.

The code is parallelized using a hybrid OpenMP/MPI approach allowing to leverage the capabilities of modern computers and computational clusters. Scalability studies have shown a reasonable speed-up of the code for workstations and small cluster use. This allows the code to be used for parametric studies and to guide the design of electron guns.

FENNECS has already been used to study trapped electron clouds in existing gyrotron electron guns [9, 26] and has shown its relevance as a design tool for future gyrotron electron gun. It has also been used to support the design of the T-REX experiment [53] and to define the relevant diagnostics and geometries to study the problem of trapped electron clouds in a more controlled environment. Furthermore, the authors believe that the governing equations are sufficiently general that the code can also be applied to study arc formations, and more conventional Penning-Malmberg traps. As the code will soon be open-source, it will allow easy adaptations to study other problems of plasma physics, where electrode geometric effects are non-negligible.

Acknowledgments

The authors would like to dedicate this article to Trach-Minh Tran who greatly contributed to the development of many codes necessary for the modeling of gyrotrons, and of the initial code espic2d which served as a seed for the creation of FENNECS.

This work has been carried out within the framework of the EUROfusion Consortium, via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion) and funded by the Swiss State Secretariat for Education, Research and Innovation (SERI). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union, the European Commission, or SERI. Neither the European Union nor the European Commission nor SERI can be held responsible for them. The calculations have been performed using the facilities of the Scientific IT and Application Support Center of EPFL. This work was supported in part by the Swiss National Science Foundation under grant No. 204631.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Appendix A. FENNECS input parameters

In this appendix, we present a list of the input parameters of the code that can be defined in the input files. To define the simulation parameters for a run of FENNECS, a single text file containing Fortran namelists is given as a command line parameter to the executable. In this appendix, the current namelists and their parameters are defined and all the quantities must be written in SI units unless specified.

Appendix A.1. &BASIC

Defines the general parameters of the run, the loading of the particles, the FEM grid, the magnetic field and how the momentum equation is solved. The following parameters can be defined:

Variable (default value) Description

job_time (3600) Wall plug time in seconds allowed for the simulation to run. The simulation will be stopped safely at the end of job_time.

extra_time (60) Extra time in seconds allowed for the simulation to finish a loop and save data to disk, once job_time is finished.

nrun (1) Number of time-steps to run.

tmax (100000) Physical simulation time in seconds after which the run must be stopped.

dt (1) Time step expressed in seconds.

it0d (1) Defines the number of steps between the saving of each scalar variable not depending on r and z (E_{pot}, E_{kin} ...).

it2d (1) Defines the number of steps between each write to the hdf5 file of the 2d variables (Φ, E, moments of the distribution ...).

itparts (1) Defines the number of steps between each write to the hdf5 file of the full particles' position and velocity.

ittracer (1) Defines the number of steps between each write to the hdf5 file of the particles' position and velocity if they are defined with the tracer property.

ittext (1) Defines the number of steps between each write to the program standard output of the total potential energy, kinetic energy and error in the energy and of the population of macro-particles in each specie.

itgraph () Defines the number of time steps between the updates of the graphical interface in case FENNECS is compiled with XGRAFIX and run using **nlxg=**.TRUE..

nbcelldiag (0) Defines the number of &celldiagnostics groups in &celldiagparams.

itcelldiag (100000) Defines the number of steps between each write of the cell diagnostic data to the hdf5 file.

resfile ('results.h5') Name and path of the hdf5 file containing the simulation results.

rstfile ('restart.h5') Name and path of the hdf5 file containing the simulation and particle data at the last time-step of the simulation, to allow the restart of the run.

nlres (.FALSE.) Sets if this run is a restart. This means that during the program initialization, the particles position and velocities will be read from a "restart.h5" file, and the new simulation data will be appended to the existing **resfile**.

nlsave (.TRUE.) Defines if the simulation data at the final time-step is saved in **rstfile** as a checkpoint to allow for a restart.

newres (.FALSE.) Sets if the result file **resfile** is a new file (.TRUE.) and should be created in the case of a restart. This means that during the program initialization, the particles position and velocities will be read from a "restart.h5" file, and the new simulation data will be saved in the new **resfile**.

nlxg (.FALSE.) Sets if the graphical interface and plots should be displayed. The code should be compiled with the XGRAFIX library and run on a personal computer.

nlPhis (.TRUE.) Sets if the macro particles will interact through the self-consistent electric field. If set to .FALSE. the self-consistent electric potential and fields will be evaluated and saved in the hdf5 result, but not applied to the particles during the momentum update. However, the external electric field is always applied.

nlfreezephi (.FALSE.) If .TRUE. the electrostatic potential is first solved self-consistently at the start or restart of the simulation and not updated afterwards. If .FALSE., ϕ is updated every time-steps.

nlclassical (.FALSE.) If set to .TRUE., γ is set to 1 for every particle throughout the simulation. This is equivalent to solving the momentum equation in the classical limit. Otherwise, the relativistic momentum equations are solved.

potinn () Value of the electric potential on the surface of the inner cylinder($\Phi(r_a)$) for simple geometries. (see &geomparams)

potout () Value of the electric potential on the surface of the outer cylinder($\Phi(r_b)$) for simple geometries. (see &geomparams)

radii () Up to 11 components array containing in order for the r dimension, the lower limit of the mesh, the i^{th} limits for the subdivision of the radial grid, the upper limit of the mesh ($[r_a, r_i, r_{i+1}, ...r_b]$).

lz () Up to 11 components array containing in order for the axial dimension, the lower limit of the mesh, the i^{th} limits for the subdivision of the axial grid, the upper limit of the mesh ($[z^-, z_i, ...z^+]$).

nz () Number of intervals in z for the mesh definition. This variable is overridden if **nnz** is defined.

nnr () Up to 10 components vector containing the number of intervals in r for the n radial mesh regions ($[n_0, n_i, ..., n_{\text{end}}]$).

nnz () Up to 10 components vector containing the number of intervals in z for the n axial mesh regions ($[n_0, n_i, ..., n_{\text{end}}]$).

ngauss () 2 component array containing the number of axial and radial gauss points used for the integration in the FEM method($[ngauss_z, ngauss_r]$).

femorder () 2 component array containing the axial and radial degree of the B-splines polynomials used in the FEM method($[order_z, order_r]$).

nlppform (.TRUE.) Defines if b-splines are evaluated using the ppform or b-spline representation of the field. ppform is usually faster as less polynomial evaluations are necessary.

distribtype (1) Switch parameter defining the distribution function used to load the particles.

1. Uniform distribution in z, $1/r$ distribution in r, Gaussian distribution in each component of the velocity given the temperature **temp**.
2. Stable distribution for a magnetic mirror, as described in section **??** and in [51].
3. Same as 2. but with uniform radial density.
7. particles loaded from **partfile** input parameter

nbspecies (1) Number of simulated species and number of **partfile** species to read and load.

partfile ("") array of length 10 to set the filename of the particle input files. The files in this list from 2 to **nbspecies** are always read and loaded. **partfile**(1) is loaded only if **Distribtype**=7.

nbaddtestspecies (0) Number of added simulated species and number of **addedtestspecfile** to read and load during the restart of the simulation.

addedtestspecfile ("") array of length 10 to set the filename of the particle input files read during a restart and added to the simulation.

partperiodic (.FALSE.) If true, sets the axial particle boundary conditions to periodic, otherwise the particles are lost when they reach the axial limits of the simulation domain.

samplefactor (1) During rescale, this parameter allows multiplying the amount of macro-particles by an integer number, while keeping the same moments of the distribution function.

nplasma () Number of macro-particles simulated for the first specie if **distribtype**!=7.

npartsalloc () Size of the allocated memory for particles of specie 1 at the beginning of the simulation. More particles can be generated during the run, but only the first **npartsalloc** particles will be saved to disk. This allows better storage of the results if ionization is used in the simulation. This parameter is only used if **distribtype**!=7.

plasmadim () 4 component array containing the axial and radial limits in m, of the electron cloud used by **Distribtype**=1 at the initialization of the particles' positions ($[z_{min}, z_{max}, r_{min}, r_{max}]$).

n0 () Initial value of the density factor, in m^{-3}, used in the different **Distribtype**, which defines the weight of the macro-particles.

temp () Temperature expressed in Kelvins used for initializing the electron velocities in **Distribtype**=1.

H0 () Initial value of the hamiltonian H_0, in SI units, used in **Distribtype**=2,3 (see Section 4.2).

P0 () Initial value of the Canonical angular momentum P_0, in SI units, used in **Distribtype**=2,3 (see Section 4.2).

nblock () Number of slices in the axial direction used to approximate the electron cloud boundary for particle initialization in **Distribtype**=2,3.

weights_scale (1.0) Allows rescaling the macro-particle weight of the main specie on restart, to artificially increase or decrease the density.

B0 () Magnetic mirror scaling factor, or reference magnetic field amplitude for the time normalisation, expressed in T. This values can also be used with **bscaling** to rescale the maximum magnetic field amplitude from the one loaded from a h5 file.

Rcurv () Magnetic mirror ratio, in the case of a magnetic field defined as a magnetic mirror (see Section 4.2).

width () Magnetic mirror width in meters, in the case of a magnetic field defined as a magnetic mirror (see Section 4.2).

magnetfile ("") Name of the hdf5 file containing the description of the magnetic field.

bscaling (-1) Defines the way the magnetic field is rescaled when read from hdf5 file. (0) no rescale, (1) rescale the maximum value of the magnetic field amplitude in the h5 file (could contain points outside the simulation grid), (-1) Rescale the magnetic field amplitude after calculation of the field at the grid points.

nlmaxwellsource (.FALSE.) Sets is the ad-hoc source module is activated in this run.

Appendix A.2. Magnetic field h5 file

To input an external magnetic field a ".h5" file must be generated and loaded by setting the **magnetfile** variable in "&basic". Such h5 files can be constructed using the "savemagtoh5.m" Matlab function in the Matlab subfolder of FENNECS. Using the hdf5 nomenclature, the magnetic field data is saved in the group "/mag/" with the following structure:

r Array of dimension n_r defining the radial grid, in meters, on which the magnetic field is defined.

z Array of dimension n_z defining the axial grid, in meters, on which the magnetic field is defined.

Athet Stores the azimuthal component of the magnetic field potential vector in T m, as a 2D array of dimension (n_z,n_r).

Bz Stores the axial component of the magnetic field vector in T, as a 2D array of dimension (n_z,n_r).

Br Stores the radial component of the magnetic field vector in T, as a 2D array of dimension (n_z,n_r).

Appendix A.3. Magnetic field txt file

To input an external magnetic field a ".txt" file must be generated and loaded by setting the **magnetfile** variable and **magnetfiletype**=1 in "&magnetparams". Such text file store on each line a new coil with on the columns from left to right: the axial lower and upper limits of the coil (in m); the radial lower and upper limits (in m); the number of turns; the number of axial and radial subdivisions for the computation of the field using elliptic integrals; the current in amperes. The total magnetic field and the magnetic field potential are then computed on each grid points using elliptic integrals [54].

Appendix A.4. &magnetparams

This input parameter allows the definition of the magnetic field using three different methods. Some input parameters are the same as the one used in the "&basic" namelist and will overwrite these values.

magnetfiletype (0) type of magnet definition. If set to 0 and **magnetfile** is empty, then the default mirror magnet definition is used. If it is set to 0 and **magnetfile** is not empty, it will assume a hdf5 input file. If this variable is set to one, the code assumes a text input file describing the geometry and currents of each coils forming the magnet. **bscaling** to rescale the maximum magnetic field amplitude from the one loaded from a h5 file.

magnetfile ("") Name of the hdf5 file containing the description of the magnetic field, or of the text file containing the geometry and currents of the azimuthally symmetric magnet.

Appendix A.5. &partsload

Defined in specific particles loading files, this namelist allows more flexibility than the legacy loader. It is also combined with either a list of individual macro particles to load or a list of slices and their respective number of macro-particles to load non-trivial distributions. Each specie should have its own particle file list, leading to a separate storage structure in the code.

Variable (default value) Description

partformat ('slices') Type of particle file.

'slices' the cloud is defined spacially as a list of slices with, on each line, the left axial limit, the lower and upper radial limit and the number of macro-particles to load in this slice. The right limit is given on the next line, and the last line comports only the axial limit.

'parts' the cloud is defined by individual particles with in order: the radial, azimuthal and axial positions, then the radial, azimuthal and axial velocities in SI units.

nblock () number of slices in the 'slices' description or number of particles in the 'parts' description.

mass (m_e) the mass of a physical particle in kg.

charge (q_e) the charge of a physical particle in C.

weight (1) the number of physical particles that one macro-particle represents.

npartsalloc () size of the initial particles array to prepare for an increase of particle numbers. In the diagnostics, only the first **npartsalloc** particles will be saved in the hdf5 file. However, more particles can be created or present in the simulations. The additional particles are simply not saved.

radialtype (2) type of radial distribution used when creating particles:

1. $1/r$ distribution in r.
2. uniform distribution in r.
3. $1/r^2$ distribution in r.
4. Gaussian distribution in r centred at $0.5(rlimits(1)+rlimits(2))$ and with $\sigma = 0.1(rlimits(2)-rlimits(1))$.

velocitytype (1) type of velocity distribution to use when creating particles

1. Maxwellian velocity of mean $< v >= 0\,\mathrm{m\,s^{-1}}$ and temperature defined in **temperature**.
2. Davidson stable distribution defined with **H0** and **P0** (see section 4.2).
3. Flat top velocity distribution with mean $< v >=$ meanv and span $\max(v) =$ meanv + spanv, $\min(v) =$ meanv − spanv

temperature (10000) Temperature in Kelvin used for the Maxwellian velocity distribution function.

H0 ($3.2 \cdot 10^{-14}$) Energy in joules of the particles for Davidson distribution function.

P0 ($8.66 \cdot 10^{-25}$) Canonical angular momentum in $\mathrm{kg\,m^2\,s^{-1}}$ of the particles for Davidson distribution function.

is_test (.FALSE.) Defines if the specie is a test specie for which all the particles properties should be saved every **ittest** time steps.

is_field (.TRUE.) If .TRUE. the specie is used to calculate the RHS of Poisson's equation, otherwise the particles see the electric field, but do not participate in its resolution.

calc_moments (.FALSE.) Determines if the moments of the distribution function of this specie needs to be calculated on the grid and saved every **it2d** time steps.

meanv (0,0,0) mean velocity, expressed in $\mathrm{m\,s^{-1}}$, in the radial, azimuthal and axial directions for the flat-top velocity loading.

spanv (0,0,0) span of the flat-top velocity, expressed in $\mathrm{m\,s^{-1}}$, in the radial, azimuthal and axial directions for the flat-top velocity loading. Velocities will be contained between meanv-spanv and meanv+spanv.

iiee_id (-1) When positive, defines that this specie contains ions that undergo ion induced electron emission and sets the index of the specie id, in the Fortran array **partslist**, of the electronic specie where to add emitted electrons.

neuttype_id (1) Defines the type of ion simulated to determine the correct yield coefficients used to calculate the number of emitted electrons per collision with an electrode. The implemented values are:

1. H_2 gas.

material_id (1) Defines the type of material for the electrodes during the ion induced electron emission events. The implemented values are:

1. Stainless steel 304.
2. Copper.
3. Aluminium.

zero_vel (.TRUE.) If .TRUE. the electrons emitted during IIEE events are generated at the electrode surfaces with $\vec{v} = 0$, otherwise their velocity is initialized normal to the electrode surface, and with a non-uniform distribution of kinetic energy described in section 3.9.

Appendix A.6. &celldiagparams

This defines the behaviour of the cell diagnostic that stores the particles' velocity and position for all particles in a given cell, and at every **itcelldiag** steps. The arguments are a list of size **nbcelldiag** that allows you to define several of these diagnostics.

Variable (default value) Description

specieid () List of species with index corresponding to the indices in the Fortran array **partslist**.

rindex () radial index of the cell to consider.

zindex () axial index of the cell to consider.

Appendix A.7. &geomparams

Namelist used to define the geometry using weighted-extended-b-splines and allowing the definition of boundary conditions on curved surfaces. Currently, Dirichlet boundary conditions are used on the metallic parts and Neumann are used otherwise. The centre cylinder will be set at a potential **Potinn** and the external cylinder and ellipse are set at a potential **Potout** of the &Basic namelist.

Variable (default value) Description

walltype (0) Type of configuration to consider. A list of predefined weights are given here, but others could be defined. For detailed examples, see the Fortran source files "geometry_mod.f90" and "weighttypes_mod.f90".

-*. If the **walltype** number is negative, a verification case with an analytical source-term is imposed, and the geometry is defined by the absolute value of **walltype**. With this, the correctness of the Poisson solution can be tested in many geometries. The manufactured electric potential solution has the form: $sin(\pi(z-z_c)/L_z) * sin(\pi(r-r_c)/L_r)+2$ as in section 4.1.

0. Coaxial configuration of constant radius with central cylinder and external cylinder.
1. Center cylinder of infinite length and external ellipse.
2. Center cylinder and combination of external cylinder with a metallic ellipse added. This configuration is used in Chapter **??**.
3. Two facing ellipses with extended cylinders where the centre ellipse radii are defined with $r_{a,inner} = z_r + r_b - r_a$, $r_{b,inner} = z_r + r_b - r_a$ and the ellipse centres are the same between inner and outer ellipse. The inner ellipse is an enlarged version of the outer ellipse.
4. Two facing ellipses with extended cylinders with identical major and minor radii.
5. Central cylinder and tilted upper cylinder, combined with tilted ellipse and left and right flat section. Natural boundary conditions are imposed at the left and right boundaries.
6. Same as previous but with a metallic wall at ground on the lower axial limit of the simulation grid.
7. Same as previous but with a metallic wall at ground on the upper axial limit of the simulation grid.
8. Same geometry as previous but only the right wall has a set potential, the rest of the electrodes are at ground.

9. Geometry read from a b-spline description of the boundaries and defined by the namelist &spldomain and a hdf5 input file.
10. Coaxial configuration closed on both ends. The applied bias is between the end electrodes and the cylindrical electrodes. This approximates an ion pump configuration.
11. Concentric ellipses as used in section 4.1.
12. Central electrode with elliptic cut biased at **Potinn**, left disc and outer cylinder are set at a potential **Potout**.

nlweb (.TRUE.) Toggle if weighted-extended-b-splines (.TRUE.) or simple weighted-b-splines (.FALSE.) must be used. There is better numerical precision and stability if set to true. Most cases will crash with mumps using **nlweb=**.FALSE..

testkr (1) For testing purposes (negative **walltype**), this defines the radial wavelength $L_r = (r_{max} - r_{min})/testkr$ of the imposed source term.

testkz (1) For testing purposes (negative **walltype**), this defines the axial wavelength $L_z = (z_{max} - z_{min})/testkz$ of the imposed source term.

z_0 () axial centre of the ellipse.

r_0 () radial centre of the ellipse.

z_r () axial radius of the ellipse.

r_r () radial radius of the ellipse.

r_a () radius of the central metallic cylinder.

r_b () radius of the external metallic cylinder.

z_a () axial position of a left metallic wall.

z_b () axial position of a right metallic wall.

Interior (-1) Defines if the inside or the outside of the ellipses are considered in the geometry.

above1 (1) Defines if the vacuum region is outside (1) or inside (-1) the cylinder of radius r_a.

above2 (-1) Defines if the vacuum region is outside (1) or inside (-1) the cylinder of radius r_b.

alpha () angle of the tilted wall and tilted ellipse w.r.t. the z axis.

r_bLeft () radial limit of the left wall for **walltype** 5 to 8.

r_bRight () radial limit of the right wall for **walltype** 5 to 8.

Appendix A.8. &spldomain

Defines the behaviour of the splinebound module which allows setting boundaries using a b-spline curve representation of the metallic, vacuum and insulating surfaces.

Variable (default value) Description

dist_extent () Set the distance in meters over which the geometric weight goes from 0 to 1 away from the boundary (see section 3.6).

h5fname () name of the h5 file containing the boundaries descriptions.

Dvals () array storing the fixed potential in V for each of the metallic boundaries defined in h5fname.

nelexact (.FALSE.) if .TRUE. calculates the geometric weight using the exact blended distance function all the time. If .FALSE., the weight is precomputed at each grid cell and used to generate an interpolant of the weight with bivariate b-splines of degree 3. The weights are then calculated by interpolation. This can add small errors for very coarse grids, but greatly increase the execution speed of the code.

Appendix A.9. Geometry h5 file

To input a geometry using a b-spline curve a ".h5" file must be generated and loaded by setting the **h5fname** variable in "&spldomain". Such hdf5 files can be constructed using the "savegeomtoh5.m" Matlab function in the Matlab subfolder of FENNECS. Using the hdf5 nomenclature, the geometry data is saved in the group "/geometry_spl/" with the following structure:

nbsplines Defines the number of spline curves stored in this file

***splineid** group named by the identifier of the spline structure written in the form "%2.2i" which give "01" for the first spline, "02" for the second ... In each of the spline curve groups, the following parameters must be defined:

order order of the spline curve.

dim dimension of the spline curve.

name name of the spline curve for easier debugging and post-processing.

type integer defining the type of boundary condition to apply on the curve surface: 0 Dirichlet constant on the full surface, 2 natural boundary condition.

periodic if set to 1 the curve is closed and periodic, and if set to 0 the curve is opened.

pos (dim,n)-array of control points of the spline curve. The first dimension defines the dimension of the spline object and the second dimension is of size n, the number of control points.

Appendix A.10. &maxwellsourceparams

This section defines the behaviour of a volumetric source creating particles uniformly in the axial direction, according to a specified distribution in the radial direction and according to a Maxwellian distribution in velocity.

Variable (default value) Description

frequency () Number of macro-particles created per second of simulated time.

temperature () temperature in Kelvins used in the Maxwellian distribution function.

rlimits () 2 element array storing the radial extent of the source.

zlimits () 2 element array storing the axial extent of the source.

time_start (-1) time in seconds of the simulation time at which the source must be turned on. -1 means start from the beginning of the simulation

time_end (-1) time in seconds of the simulation time at which the source must be turned off. -1 means, never turn off the source.

radialtype (2) type of radial distribution to use when creating particles. The implemented values are:

1. $1/r$ distribution in r.
2. uniform distribution in r.
3. $1/r^2$ distribution in r.
4. Gaussian distribution in r centred at $0.5(rlimits(1)+rlimits(2))$ and with $\sigma = 0.1(rlimits(2)-rlimits(1))$.

Appendix A.11. &neutcolparams

This defines the behaviour of the elastic and ionisation collisions between electrons and the residual neutral gas particles.

Variable (default value) Description

nlcol (.FALSE.) defines if the collisions are active or not. If both ela_cross_sec_file and io_cross_sec_file are empty, the collisions are deactivated.

neutdens ($2.4 \cdot 10^{16} m^{-3}$) density of the RNG expressed in m^{-3}.

Eion (21.56 eV) first ionization energy, expressed in eV, of the neutral considered.

scatter_fac (24.2 eV) tabulated scatter factor, expressed in eV, used to compute the fraction of energy between scattered and created electrons in an ionization event. To set this parameter, please refer to [30].

io_cross_sec_file () name of the file containing the table of cross-sections as a function of energy in eV for the ionisation. In this file the comments are indicated with "!" and the table is just a two column list with the energy in eV and the cross-section in m^2. Example files are stored in the "wk" folder of the repository, but can also be downloaded from the LXCat database [28].

ela_cross_sec_file () name of the file containing the table of cross-sections as a function of energy in eV for the elastic collisions. In this file the comments are indicated with "!" and the table is just a two column list with the energy in eV and the cross-section in m^2. Example files are stored in the "wk" folder of the repository, but can also be downloaded from the LXCat database [28].

species (1,-1) The first number sets the colliding specie ID in the **partslist** Fortran array. In the case of an ionisation, the second number defines in which specie the ions should be added with 0 velocity. If the second number is lower than 1, no ions are created.

isotropic (.FALSE.) Defines if the scattering of the velocity vector during collisions is isotropic or anisotropic according to the differential cross-section described in [29]. The default is anisotropic.

itcol (1) allows running the collision routine every **itcol** time steps of the particle pushing. If the pressure is too low, this parameter can reduce unnecessary calculations.

Appendix A.12. &psupplyparams

Sets the parameters of the non-ideal power supply described in section 3.7. The input parameters are:

Variable (default value) Description

active (.FALSE.) Defines if this module is active.

expneutdens () neutral density, in m^{-3}, measured in the experiment we want to simulate. This permits correct timescale separation and rescaling because we accelerate the ionisation time-scales (see sections 3.7 and 2.4).

PsResistor () Internal resistance of the power supply in Ohms.

geomcapacitor () Total capacitance of the geometry and connecting cables in Farrads.

targetbias () Set bias v_s in V requested on the power supply.

nbhdt () Number of Boris algorithm time steps between each half time-steps of the Runge-Kutta algorithm used to compute the time-evolution of the bias at the surface of the electrodes.

bdpos (0) Array of integers indicating for each boundary the direction of the current for the collected species. The boundaries with **bdpos**(i)=-1 are set to $V_i = -V_s$ and all the other are set to ground.

References

[1] M. Vogel, Particle Confinement in Penning Traps, Vol. 100 of Springer Series on Atomic, Optical, and Plasma Physics, Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-76264-7.
URL http://link.springer.com/10.1007/978-3-319-76264-7

[2] R. C. Davidson, Physics of Nonneutral Plasmas, CO-PUBLISHED WITH WORLD SCIENTIFIC PUBLISHING CO, 2001. doi:10.1142/p251.
URL http://www.worldscientific.com/worldscibooks/10.1142/p251

[3] K. R. Chu, The electron cyclotron maser, Rev. Mod. Phys. 76 (2) (2004) 489–540, publisher: American Physical Society. doi:10.1103/RevModPhys.76.489.
URL https://link.aps.org/doi/10.1103/RevModPhys.76.489

[4] T. Mohamed, A. Mohri, Y. Yamazaki, Comparison of non-neutral electron plasma confinement in harmonic and rectangular potentials in a very dense regime, Physics of Plasmas 20 (1) (2013) 012502, publisher: American Institute of Physics. doi:10.1063/1.4773900.
URL https://aip.scitation.org/doi/10.1063/1.4773900

[5] W. A. Bertsche, E. Butler, M. Charlton, N. Madsen, Physics with antihydrogen, Journal of Physics B: Atomic, Molecular and Optical Physics 48 (23) (2015) 232001, publisher: IOP Publishing. doi:10.1088/0953-4075/48/23/232001.
URL https://dx.doi.org/10.1088/0953-4075/48/23/232001

[6] P. Pérez, D. Banerjee, F. Biraben, D. Brook-Roberge, M. Charlton, P. Cladé, P. Comini, P. Crivelli, O. Dalkarov, P. Debu, A. Douillet, G. Dufour, P. Dupré, S. Eriksson, P. Froelich, P. Grandemange, S. Guellati, R. Guérout, J. M. Heinrich, P.-A. Hervieux, L. Hilico, A. Husson, P. Indelicato, S. Jonsell, J.-P. Karr, K. Khabarova, N. Kolachevsky, N. Kuroda, A. Lambrecht, A. M. M. Leite, L. Liszkay, D. Lunney, N. Madsen, G. Manfredi, B. Mansoulié, Y. Matsuda, A. Mohri, T. Mortensen, Y. Nagashima, V. Nesvizhevsky, F. Nez, C. Regenfus, J.-M. Rey, J.-M. Reymond, S. Reynaud, A. Rubbia, Y. Sacquin, F. Schmidt-Kaler, N. Sillitoe, M. Staszczak, C. I. Szabo-Foster, H. Torii, B. Vallage, M. Valdes, D. P. Van der Werf, A. Voronin, J. Walz, S. Wolf, S. Wronka, Y. Yamazaki, The GBAR antimatter gravity experiment, Hyperfine Interactions 233 (1) (2015) 21–27. doi:10.1007/s10751-015-1154-8.
URL https://doi.org/10.1007/s10751-015-1154-8

[7] M. R. Stoneking, T. S. Pedersen, P. Helander, H. Chen, U. Hergenhahn, E. V. Stenson, G. Fiksel, J. v. d. Linden, H. Saitoh, C. M. Surko,

J. R. Danielson, C. Hugenschmidt, J. Horn-Stanja, A. Mishchenko, D. Kennedy, A. Deller, A. Card, S. Nißl, M. Singer, S. König, L. Willingale, J. Peebles, M. R. Edwards, K. Chin, A new frontier in laboratory physics: magnetized electron–positron plasmas, Journal of Plasma Physics 86 (6) (2020) 155860601, publisher: Cambridge University Press. doi:10.1017/S0022377820001385.
URL https://www.cambridge.org/core/journals/journal-of-plasma-physics/article/new-frontier-in-laboratory-physics-magnetized-electronpositron-plasmas/F86C1C197E256296EE9C86045C692FD7

[8] I. G. Pagonakis, J. Hogge, T. Goodman, S. Alberti, B. Piosczyk, S. Illy, T. Rzesnicki, S. Kern, C. Lievin, Gun design criteria for the refurbishment of the first prototype of the EU 170GHz/2MW/CW coaxial cavity gyrotron for ITER, in: 2009 34th International Conference on Infrared, Millimeter, and Terahertz Waves, 2009, pp. 1–2. doi:10.1109/ICIMW.2009.5324625.

[9] G. Le Bars, J.-P. Hogge, J. Loizu, S. Alberti, F. Romano, A. Cerfon, Self-consistent formation and steady-state characterization of trapped high-energy electron clouds in the presence of a neutral gas background, Physics of Plasmas 29 (8) (2022) 082105, publisher: American Institute of Physics. doi:10.1063/5.0098567.
URL https://aip.scitation.org/doi/full/10.1063/5.0098567

[10] I. G. Pagonakis, B. Piosczyk, J. Zhang, S. Illy, T. Rzesnicki, J.-P. Hogge, K. Avramidis, G. Gantenbein, M. Thumm, J. Jelonnek, Electron trapping mechanisms in magnetron injection guns, Physics of Plasmas 23 (2) (2016) 023105. doi:10.1063/1.4941705.
URL https://aip.scitation.org/doi/abs/10.1063/1.4941705

[11] LSP Suite.
URL https://www.northropgrumman.com/space/pic-code-software/lsp-suite

[12] A. J. Woods, L. D. Ludeking, MAGIC electromagnetic FDTD-PIC code dense plasma model comparison with Lsp, in: 2009 IEEE International Vacuum Electronics Conference, 2009, pp. 165–166. doi:10.1109/IVELEC.2009.5193488.

[13] S. Mattei, K. Nishida, M. Onai, J. Lettry, M. Q. Tran, A. Hatayama, A fully-implicit Particle-In-Cell Monte Carlo Collision code for the simulation of inductively coupled plasmas, Journal of Computational Physics 350 (2017) 891–906. doi:10.1016/j.jcp.2017.09.015.
URL https://www.sciencedirect.com/science/article/pii/S0021999117306745

[14] B. Herrmannsfeldt, ELECTRON TRAJECTORY PROGRAM.

[15] S. Illy, J. Zhang, J. Jelonnek, Gyrotron electron gun and collector simulation with the ESRAY beam optics code, in: 2015 IEEE International Vacuum Electronics Conference (IVEC), 2015, pp. 1–2. doi:10.1109/IVEC.2015.7223779.

[16] T. Tran, D. Whaley, S. Merazzi, R. Gruber, Daphne, a 2d axisymmetric electron gun simulation code, Tech. rep., lRP–494/94 INIS Reference Number: 25050357 (1994).

[17] I. G. Pagonakis, J. L. Vomvoridis, The self-consistent 3D trajectory electrostatic code ARIADNE for gyrotron beam tunnel simulation, in: Infrared and Millimeter Waves, Conference Digest of the 2004 Joint 29th International Conference on 2004 and 12th International Conference on Terahertz Electronics, 2004., 2004, pp. 657–658. doi:10.1109/ICIMW.2004.1422262.

[18] D. P. Grote, A. Friedman, J. Vay, I. Haber, The WARP Code: Modeling High Intensity Ion Beams, AIP Conference Proceedings 749 (1) (2005) 55–58, publisher: American Institute of Physics. doi:10.1063/1.1893366.
URL https://aip.scitation.org/doi/abs/10.1063/1.1893366

[19] J.-L. Vay, D. Grote, R. Cohen, A. Friedman, Novel methods in the Particle-In-Cell accelerator Code-Framework Warp, Computational Science & Discovery 5 (2012) 014019. doi:10.1088/1749-4699/5/1/014019.

[20] A. Friedman, R. H. Cohen, D. P. Grote, S. M. Lund, W. M. Sharp, J.-L. Vay, I. Haber, R. A. Kishek, Computational Methods in the Warp Code Framework for Kinetic Simulations of Particle Beams and Plasmas, IEEE Transactions on Plasma Science 42 (5) (2014) 1321–1334, conference Name: IEEE Transactions on Plasma Science. doi:10.1109/TPS.2014.2308546.

[21] K. Höllig, C. Apprich, A. Streit, Introduction to the Web-method and its applications, Advances in Computational Mathematics 23 (1) (2005) 215–237. doi:10.1007/s10444-004-1811-y.
URL https://doi.org/10.1007/s10444-004-1811-y

[22] G. Apaydin, N. Ari, Use of WEB-splines of arbitrary domain for waveguides, in: 2008 12th International Conference on Mathematical Methods in Electromagnetic Theory, 2008, pp. 385–388, iSSN: 2161-1750. doi:10.1109/MMET.2008.4581003.

[23] V. V. K. S. Kumar, B. V. R. Kumar, P. C. Das, Weighted extended B-spline method for the approximation of the stationary Stokes problem, Journal of Computational and Applied Mathematics 186 (2) (2006) 335–348. doi:10.1016/j.cam.2005.02.008.
URL https://www.sciencedirect.com/science/article/pii/S0377042705000981

[24] K. Höllig, U. Reif, J. Wipper, Weighted Extended B-Spline Approximation of Dirichlet Problems, SIAM Journal on Numerical Analysis 39 (2) (2001) 442–462, publisher: Society for Industrial and Applied Mathematics. doi:10.1137/S0036142900373208.
URL https://epubs.siam.org/doi/abs/10.1137/S0036142900373208

[25] K. Höllig, Finite Element Methods with B-Splines, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, 2003. doi:10.1137/1.9780898717532.
URL https://epubs.siam.org/doi/book/10.1137/1.9780898717532

[26] G. Le Bars, J. Loizu, J.-P. Hogge, S. Alberti, F. Romano, J. Genoud, I. G. Pagonakis, First self-consistent simulations of trapped electron clouds in a gyrotron gun and comparison with experiments, Physics of Plasmas 30 (3) (2023) 030702, _eprint: https://pubs.aip.org/aip/pop/article-pdf/doi/10.1063/5.0136340/16793537/030702_1_online.pdf. doi:10.1063/5.0136340.
URL https://doi.org/10.1063/5.0136340

[27] A. Anders, Cathodic Arcs: From Fractal Spots to Energetic Condensation,

Springer Science & Business Media, 2009, google-Books-ID: rwIUhsbB-HQYC.

[28] Biagi-v8.9 database, private communication, www.lxcat.net, retrieved on June 1 (Jun. 2021).

[29] A. Okhrimovskyy, A. Bogaerts, R. Gijbels, Electron anisotropic scattering in gases: A formula for Monte Carlo simulations, Physical Review E 65 (3) (2002) 037402, publisher: American Physical Society. doi:10.1103/PhysRevE.65.037402.
URL https://link.aps.org/doi/10.1103/PhysRevE.65.037402

[30] C. B. Opal, W. K. Peterson, E. C. Beaty, Measurements of Secondary-Electron Spectra Produced by Electron Impact Ionization of a Number of Simple Gases, The Journal of Chemical Physics 55 (8) (1971) 4100–4106, publisher: American Institute of Physics. doi:10.1063/1.1676707.
URL https://aip.scitation.org/doi/10.1063/1.1676707

[31] T. Holstein, Energy Distribution of Electrons in High Frequency Gas Discharges, Physical Review 70 (5-6) (1946) 367–384, publisher: American Physical Society. doi:10.1103/PhysRev.70.367.
URL https://link.aps.org/doi/10.1103/PhysRev.70.367

[32] S. Yoshida, A. V. Phelps, L. C. Pitchford, Effect of electrons produced by ionization on calculated electron-energy distributions, Physical Review A 27 (6) (1983) 2858–2867, publisher: American Physical Society. doi:10.1103/PhysRevA.27.2858.
URL https://link.aps.org/doi/10.1103/PhysRevA.27.2858

[33] J. P. Boris, others, Relativistic plasma simulation-optimization of a hybrid code, in: Proc. Fourth Conf. Num. Sim. Plasmas, 1970, pp. 3–67.

[34] Ahlberg, J. H., Nilson, Edwin N., Walsh, J. L., The Theory of Splines and Their Applications, Academic Press New York, 1967.
URL https://www.elsevier.com/books/the-theory-of-splines-and-their-applications/ahlberg/978-1-4831-9792-0

[35] K. Höllig, J. Hörner, Approximation and Modeling with B-Splines, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2013. doi:10.1137/1.9781611972955.

URL https://epubs.siam.org/doi/book/10.1137/1.9781611972955

[36] E. T. Y. Lee, Marsden's identity, Computer Aided Geometric Design 13 (4) (1996) 287–305. doi:10.1016/0167-8396(95)00027-5.
URL https://www.sciencedirect.com/science/article/pii/0167839695000275

[37] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, International Journal for Numerical Methods in Engineering 46 (1) (1999) 131–150, publisher: Wiley. doi:10.1002/(SICI)1097-0207(19990910)46:1<131::AID-NME726>3.0.CO;2-J.
URL https://hal.science/hal-01004829

[38] T. Strouboulis, I. Babuška, K. Copps, The design and analysis of the Generalized Finite Element Method, Computer Methods in Applied Mechanics and Engineering 181 (1-3) (2000) 43–69. doi:10.1016/S0045-7825(99)00072-9.
URL https://linkinghub.elsevier.com/retrieve/pii/S0045782599000729

[39] I. Babuška, U. Banerjee, Stable Generalized Finite Element Method (SGFEM), Computer Methods in Applied Mechanics and Engineering 201-204 (2012) 91–111. doi:10.1016/j.cma.2011.09.012.
URL https://linkinghub.elsevier.com/retrieve/pii/S0045782511003082

[40] V. L. Rvachev, Method of R-functions in boundary-value problems, Soviet Applied Mechanics 11 (4) (1975) 345–354. doi:10.1007/BF00882900.
URL https://doi.org/10.1007/BF00882900

[41] V. L. Rvachev, T. I. Sheiko, V. Shapiro, I. Tsukanov, Transfinite interpolation over implicitly defined sets, Computer Aided Geometric Design 18 (3) (2001) 195–220. doi:10.1016/S0167-8396(01)00015-2.
URL http://www.sciencedirect.com/science/article/pii/S0167839601000152

[42] C. Birdsall, Particle-in-cell charged-particle simulations, plus Monte Carlo collisions with neutral atoms, PIC-MCC, IEEE Transactions on Plasma Science 19 (2) (1991) 65–85, number: 2 Conference Name: IEEE Transactions on Plasma Science. doi:10.1109/27.106800.

[43] S. F. Biagi, Fortran program, magboltz v8.9, lxcat.net (Jun. 2021).

[44] J. F. J. Janssen, L. C. Pitchford, G. J. M. Hagelaar, J. v. Dijk, Evaluation of angular scattering models for electron-neutral collisions in Monte Carlo simulations, Plasma Sources Science and Technology 25 (5) (2016) 055026, publisher: IOP Publishing. doi:10.1088/0963-0252/25/5/055026.
URL https://doi.org/10.1088/0963-0252/25/5/055026

[45] L. M. Kishinevsky, Estimation of electron potential emission yield dependence on metal and ion parameters, Radiation Effects 19 (1) (1973) 23–27. doi:10.1080/00337577308232211.
URL http://www.tandfonline.com/doi/abs/10.1080/00337577308232211

[46] D. Hasselkamp, H. Rothard, K.-O. Groeneveld, J. Kemmler, P. Varga, H. Winter (Eds.), Particle Induced Electron Emission II, Vol. 123 of Springer Tracts in Modern Physics, Springer, Berlin, Heidelberg, 1992. doi:10.1007/BFb0038297.
URL http://link.springer.com/10.1007/BFb0038297

[47] J. Schou, Transport theory for kinetic emission of secondary electrons from solids, Physical Review B 22 (5) (1980) 2141–2174, publisher: American Physical Society. doi:10.1103/PhysRevB.22.2141.
URL https://link.aps.org/doi/10.1103/PhysRevB.22.2141

[48] J. F. Janni, Energy loss, range, path length, time-of-flight, straggling, multiple scattering, and nuclear interaction probability: In Two Parts. Part 1. For 63 Compounds Part 2. For Elements 1 ≤ Z ≤ 92, Atomic Data and Nuclear Data Tables 27 (4) (1982) 341–529. doi:10.1016/0092-640X(82)90005-5.
URL https://www.sciencedirect.com/science/article/pii/0092640X82900055

[49] J. F. Janni, Energy loss, range, path length, time-of-flight, straggling, multiple scattering, and nuclear interaction probability: In two parts. Part 1. For 63 compounds Part 2. For elements 1 ≤ Z ≤ 92, Atomic Data and Nuclear Data Tables 27 (2) (1982) 147–339. doi:10.1016/0092-640X(82)90004-3.
URL https://www.sciencedirect.com/science/article/pii/0092640X82900043

[50] D. Hasselkamp, S. Hippler, A. Scharmann, Ion-induced secondary electron spectra from clean metal surfaces, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 18 (1-6) (1987) 561–565. doi:10.1016/S0168-583X(86)80088-X.
URL https://linkinghub.elsevier.com/retrieve/pii/S0168583X8680088X

[51] R. C. Davidson, Equilibrium and stability of mirror-confined nonneutral E layers, Physics of Fluids 16 (12) (1973) 2199. doi:10.1063/1.1694287.
URL https://aip.scitation.org/doi/10.1063/1.1694287

[52] Jed cluster.
URL https://www.epfl.ch/research/facilities/scitas/jed

[53] F. Romano, S. Alberti, J.-P. Hogge, J. Genoud, G. Le Bars, J. Loizu, The TRapped Electrons eXperiment (T-REX), in: 2022 13th International Workshop on Non-Neutral Plasmas, Milano, To be published.
URL https://drive.google.com/file/d/1v77t-8oKMTDmqLGU6fKkSpaZL_cYSYcs/view

[54] W. R. Smythe, Smythe - Static and Dynamic Electricity, McGraw-Hill Book Company, 1950.