

Understanding of the physics and numerical simulation of helicon double layer thruster

Final Report

Authors: D.Pavarin¹, M.Manente¹, J.Carlsson¹, S.Suman¹, D.Melazzi¹,
Cristina Bramanti²

Affiliation: ¹University of Padova - CISAS, ²Advanced Concepts Team,
European Space Agency.

Date: July 2009

Contacts:

Daniele Pavarin, University of Padova - CISAS

Tel: +39 0498276854

Fax: +39 0498276788

e-mail: daniele.pavarin@unipd.it

Leopold Summerer, Advanced Concepts Team

Tel: +31-71-565 6227

Fax: +31-71-565 8018

e-mail: act@esa.int



Available on the ACT website
<http://www.esa.int/act>

Ariadna ID: 05/3201CCN

Study Type: Standard

Contract Number: 19694/05/NL/CB (CCN)

Understanding of the physics and numerical
simulation of helicon double layer thruster concept
ESA ARIADNA ITT AO4919 05/3201CCN

D.Pavarin M.Manente J.Carlsson S.Suman D.Melazzi

Contents

1	Coupled Codes	4
2	Codes Driver	6
3	Wavecode	7
3.1	Wavecode Theoretical Model	8
3.1.1	Governing Equations	8
3.1.2	Equations for Waves in the Helicon Experiment	9
3.1.3	Hypothesis	9
3.1.4	Constitutive Relations	11
3.1.5	Normalization	12
3.1.6	Boundary Conditions	12
3.1.7	Representation of Waves	13
3.2	Wavecode Numerical Model	13
3.2.1	Solution Scheme	14
3.2.2	Numerical Scheme	14
3.3	Wavecode Staggered Yee Mesh	17
3.3.1	3D Cartesian Coordinate System	17
3.3.2	Transformation to 3D Cylindrical Coordinate System	18
3.3.3	Transformation to 1D Cylindrical Coordinate System	19
3.3.4	Spatial Discretization for 1D Wave Code	19
3.4	Wavecode Perfectly Matched Layer	20
3.4.1	Complex Coordinate Stretching	21
3.4.2	Wavecode Implementation	22
3.5	Wavecode Upgrade	25
4	Wavecode Validation & Verify	26
4.0.1	Results	27
5	Particle Code	33
5.1	Startup section	34
5.2	Physical to Computational ratio	35
5.3	Interface with WaveCode	36
5.3.1	Current density weighting	37
5.4	Load particles distributions	40

5.4.1	Positions	43
5.4.2	Velocities	44
5.5	Parameters definition	47
5.6	Time loops	47
5.6.1	Timestep setup phase	48
5.6.2	Radial self-consistent electrostatic field solver	48
5.6.3	Fields interpolation	51
5.6.4	Integration of the equations of motion	53
5.6.5	Boundaries check and plasma source	54
5.7	Current density FFT	56
5.7.1	FFT Normalization	56
5.8	Phase space saving	57
5.9	Plasma power absorption	57
6	Particle Code Validation & Verify	58
6.1	Single Particle Validations	58
6.1.1	Cyclotron motion	58
6.1.2	Constant axial electric field	59
6.2	Initial velocity distribution	61
6.3	Current density validation	65
7	Coupled Codes Results	69
7.1	Results	69
7.2	Results with self-consistent electrostatic field and collisions	73
7.3	Procedures for Antenna Design	76
7.3.1	$m = 0$ case	77
7.3.2	$m = +1$ case	77

1 Coupled Codes

Helicon plasma sources sustain steady-state plasma production through propagation and absorption of helicon waves, which are launched applying an axial magnetic field and coupling a radio-frequency (RF) antenna to the plasma confined inside a cylindrical device. Throughout the last years, helicon sources have been recognized to be much more effective rather than capacitive and inductive sources in generating plasmas and so this kind of sources have been recently considered for space electric propulsion devices. However, deep comprehension of coupling mechanism has not been reached yet. Moreover, in order to design an efficient plasma source for space applications a code with low computational cost is also necessary. A set of software tools for designing the antenna of a helicon plasma thruster has been developed. The main software requirements are to:

- Simulate the propagation, absorption and mode conversion of all relevant waves in the plasma
- Be computationally efficient to allow parameter scans in reasonable time and with reasonable computational resources

The software tools will be used to systematically compare helicon waves launched with different axial and azimuthal mode numbers to find the one that gives optimal power absorption. With the optimal mode numbers identified, an antenna can be designed that excites the desired mode. New codes have been developed to simulate wave propagation, absorption and mode conversion: a Wavecode and a Particle Code.

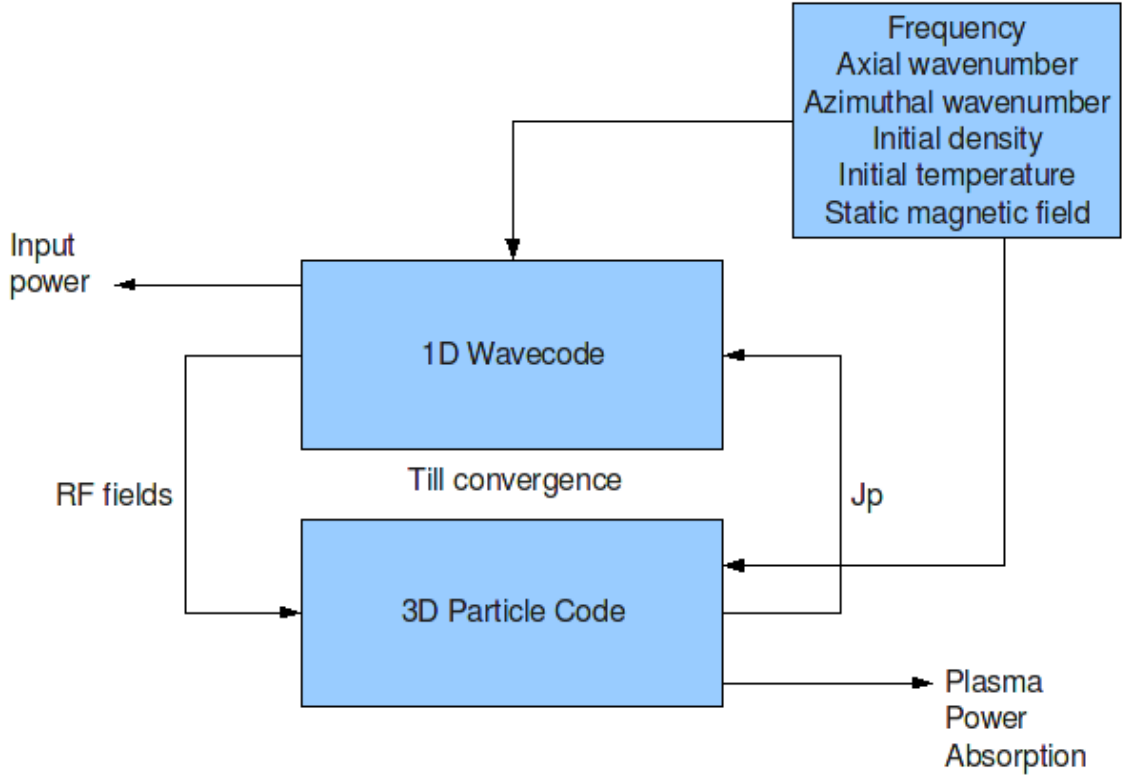


Figure 1: Codes concept

The Wavecode solves Ampere’s circuital Law and Faraday’s Law of induction in 1D (radial only), in order to obtain the values of the Electric field and Magnetic field acting on plasma particles. On the source term of Ampere’s law by the way, the current density J should take into account both the antenna current density and the plasma current density. One can split J into J_a (antenna current) and J_p (plasma current). Then one could model J_p by extending the dielectric tensor ϵ . For our code setup we want to directly calculate J_p with a 3D Particle Code instead of trying to make the dielectric tensor model it. This will make it much easier to fully take into account kinetic effects. The disadvantage is that we have to iterate between the two codes, as can be seen in Fig. 1. Our J_p will always be calculated with the old \mathbf{E} and \mathbf{B} , so we have to iterate until convergence for the absorbed power.

2 Codes Driver

The Codes Driver is responsible of controlling the Wavecode and the Particle Code during coupled simulations, running fully automated test matrices. It's written using the Python programming language, that allows easy and powerful interface with the operating system managing the execution of two independent codes. Test matrix parameters can be defined in the lists at the beginning of the driver. The driver, for each test matrix case, writes correct input files for the Wavecode and the Particle Code, and then loops between the two codes for the simulation. At the end of each test case, all the data relevant for the simulation is stored, that is the iterations history of Wavecode outputs, the history of Particle Code outputs, and the input files of the two codes for that test case.

3 Wavecode

Plasma waves play an important role in plasma technology and space plasma physics. In particular, they have long been under study for the heating of fusion plasma, and recently, helicon wave driven plasma sources have become very important contenders as sources for space-electric-propulsion devices.

A 1D plasma wave code, written in ANSI C and named Wavecode, specifically designed for the treatment of radiofrequency wave excitation and propagation in cylindrical plasma, is developed. The code will solve Maxwell equations, on steady state condition, on the radial direction only, assuming spatial periodicity through modes.

The code will address two main issues: the former is to analyze the wave propagation within the plasma source, finding extensive application in the study of helicon wave driven plasma sources and in a deep comprehension of power coupling mechanism, while the latter is to work as a computationally efficient design tool so as to find the optimal antenna configuration.

Therefore Wavecode will be used to systematically compare helicon waves launched with different axial and azimuthal mode numbers to find the one that gives optimal power absorption, and which input power to the antenna is needed to achieve a specified specific impulse and thrust. Once that the optimal mode numbers will be identified, an antenna, which will excite the desired mode, will be designed.

In order to achieve the results explained above, Wavecode will work in a computational loop with a 3-D Particle Code.

For the geometry and the input RF power levels considered in the helicon experiment, relatively weak fields are expected so as to make nonlinearities and kinetic effects negligible; hence, the antenna spectrum can be described as a linear superimposition of individual modes and all waves of interest have the same frequency as the driving antenna current. Therefore, the code will model each EM field component as harmonic function with respect to time and space, taking into account only the radial dependence; the radial derivatives will be approximated by second-order finite differences scheme in a staggered mesh of Yee type, while the azimuthal and axial derivatives will be modelled by wave numbers.

The output will provide time-harmonic electromagnetic fields generated by the antenna, both in vacuum and plasma conditions; vacuum solutions will be calculated imposing PML boundary conditions, while plasma ones will be obtained calculating self-consistently the plasma current driven by the waves. This will be done by a

very general interface, where the plasma current is either modeled by a dielectric tensor or treated as externally given, as for the antenna current.

The discretized differential equations become a linear system with complex data type and a square, sparse matrix. This kind of problem will be solved by the implementation into the code of the MULTifrontal Massively Parallel Solver (MUMPS), which solves linear systems with a direct method in a computationally efficient and numerically accurate way.

To make an independent benchmarking of the wave code, a detailed quantitative comparison has been done; therefore, wave fields calculated by the wave code have been compared against analytical solutions and numerical results.

3.1 Wavecode Theoretical Model

3.1.1 Governing Equations

Maxwell's equations describe all classical electromagnetic phenomena:

$$\left\{ \begin{array}{l} \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \\ \nabla \cdot \mathbf{D} = \rho \\ \nabla \cdot \mathbf{B} = 0 \end{array} \right. \quad (1)$$

The displacement current term $\frac{\partial \mathbf{D}}{\partial t}$ in Ampere's law is essential in predicting the existence of propagating electromagnetic waves; moreover, its role is fundamental to guarantee the charge conservation.

Eqs. (1) are in SI units. The quantities \mathbf{E} and \mathbf{H} are the electric and magnetic *field intensities* and are measured in units of $[V/m]$ and $[A/m]$, respectively. The quantities \mathbf{D} and \mathbf{B} are the electric and magnetic *flux densities* and are in units of $[C/m^2]$ and $[Wb/m^2]$, or $[T]$. \mathbf{B} is also called the *magnetic induction*.

The quantities ρ and \mathbf{J} are the *volume charge density* and *electric current density* (charge flux) of any external charges. They are measured in units of $[C/m^3]$ and $[A/m^2]$.

The charge and current densities ρ , \mathbf{J} may be thought of as the *sources* of the electromagnetic fields. For wave propagation problems, these densities are localized in space; the generated electric and magnetic fields are radiated away from these sources and can propagate to large distances to the receiving antennas.

Away from the sources, in source-free regions of space, Maxwell's equations take the simpler form:

$$\begin{cases} \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} \\ \nabla \cdot \mathbf{D} = 0 \\ \nabla \cdot \mathbf{B} = 0 \end{cases} \quad (2)$$

3.1.2 Equations for Waves in the Helicon Experiment

Since we will design and write a wave code that simulates the wave physics in the helicon experiment, the governing equations are Faraday's law,

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \quad (3)$$

and Ampere's law,

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J}, \quad (4)$$

both expressed in a cylindrical coordinate system (r, θ, z) .

Moreover, the current density \mathbf{J} can be splitted into two components: the antenna current density \mathbf{J}_a and the plasma current density \mathbf{J}_p . This approach anyway is convenient only for cold plasma, because it means that the plasma current would be modelled by the dielectric tensor ϵ ; actually this current component will be calculated by the 3-D Particle Code, which will use the electromagnetic fields, obtained by the Wavecode, so as to move the charged particles inside the plasma source. In this way all kinetic effects (finite gyro radius, non-Maxwellian particle distributions, etc...) can be easily taken into account.

These equations, when combined with the Lorentz force, provide a complete description of the classical dynamics of interacting charged particles and electromagnetic fields.

So we need to translate \mathbf{H} and \mathbf{D} into \mathbf{B} and \mathbf{E} , respectively, and we have to use the *constitutive relations*, which summarize, for macroscopic media, the dynamical response of the aggregate atoms.

3.1.3 Hypothesis

In order to avoid making the wave code unnecessarily complicated, we did some hypothesis:

- Cylindrical coordinate system (r, θ, z) .
- Monodimensional model.
- Weak \mathbf{E} and \mathbf{B} fields.

Let us analyze the first hypothesis. The choice of this kind of coordinate system is due to the geometry configuration of the helicon experiment; indeed, the injected neutral gas is contained in a Pyrex glass tube.

Moreover, as far as a plasma thruster for space propulsion is concerned, a cylindrical geometry is suitable to satisfy the requirements and drivers for a space propulsion system design.

Regarding the second hypothesis, it is worth noticing that the code, developed in this project, is a new *wave code*, which simulates the wave physics inside a plasma source and the power coupling between the antenna and the plasma source itself; indeed, we just want it to be as much simple as it could be, so as to be sure that it simulates well the physics phenomena involved in the helicon experiment. Moreover we are interested in what happens on a circular section of the plasma source, particularly along the radial direction. Therefore a monodimensional model is considered, where only the radial dependence is fully taken into account.

Finally, let us consider the last hypothesis. When we talk about weak \mathbf{E} and \mathbf{B} fields we mean that the wave-particle interaction only cause a perturbation of the particle orbits. Starting from this assumptions, follows that:

1. simplified constitutive relations can be considered , providing a good approximation for most relevant materials involved in the helicon experiment;
2. simplified non-zero elements of the tensors used to model the plasma medium inside the computational domain;
3. we can assume that all waves of interest, involved in the helicon experiment, have the same frequency as the driving antenna current, so that:
 - $\frac{\partial}{\partial t} = -i\omega$;
 - we can rule out the possibility of nonlinear effects such as mode-mode coupling;
4. for the azimuthal derivatives we will thus replace: $\frac{\partial}{\partial \theta} = im$, where m stands for the azimuthal wave number;

5. while for the axial derivatives: $\frac{\partial}{\partial z} = ik_z$, where k_z stands for the axial wave number.

3.1.4 Constitutive Relations

The electric and magnetic flux densities \mathbf{D} , \mathbf{B} are related to the field intensities \mathbf{E} , \mathbf{H} via the so-called *constitutive relations*, whose precise form depends on the material in which the fields exist. Anyway, this kind of relations allow us not only to translate \mathbf{H} and \mathbf{D} into \mathbf{B} and \mathbf{E} , respectively, but also to close the electromagnetic propagation problem. In vacuum, they take their simplest form:

$$\begin{aligned}\mathbf{D} &= \varepsilon_0 \mathbf{E} \\ \mathbf{B} &= \mu_0 \mathbf{H}\end{aligned}\tag{5}$$

where ε_0 , μ_0 are the *permittivity* and *permeability* of vacuum. More generally, constitutive relations may be inhomogeneous, anisotropic, nonlinear, frequency dependent (dispersive), or all of the above. Therefore, in general, $\boldsymbol{\varepsilon}$ and $\boldsymbol{\mu}$ are both rank-three tensors but, to avoid making the wave code unnecessarily complicated and considering the hypothesis made, we would like to simplify them.

For the gasses relevant for the helicon experiment, the dielectric coefficient differs from the vacuum value by less than 0.1%. For Pyrex glass, a scalar dielectric coefficient is still a very good approximation, but its value is much larger than for vacuum, approximately $5.6\varepsilon_0$.

There are dielectric materials (e.g. boron nitride) that are anisotropic, and a scalar dielectric coefficient does not suffice. For our experiment, they are not relevant; however, there are still three reasons to use tensors.

The first one is that diagonal tensors are needed to implement a Perfectly Matched Layer (PML), which is the best way to prevent the wave from bouncing back from the artificial domain boundary and numerically pollute the solution.

The second one is that a cold plasma can be modelled as a dielectric tensor. The most important reason anyway is that it will be useful to validate our algorithm against such a simplified model.

So the permittivity and permeability tensors, $\boldsymbol{\varepsilon}$ and $\boldsymbol{\mu}$ respectively, are defined by:

$$\boldsymbol{\varepsilon} = \varepsilon_0 \begin{bmatrix} \varepsilon_r & 0 & 0 \\ 0 & \varepsilon_\theta & 0 \\ 0 & 0 & \varepsilon_z \end{bmatrix}, \quad \boldsymbol{\mu} = \mu_0 \begin{bmatrix} \mu_r & 0 & 0 \\ 0 & \mu_\theta & 0 \\ 0 & 0 & \mu_z \end{bmatrix}\tag{6}$$

where all the diagonal elements of the two matrix in Eq. (6) are in the cylindrical coordinate system and are function of the radius r .

3.1.5 Normalization

Starting from the governing Eq. (4) and using the constitutive relation with the tensors $\boldsymbol{\epsilon}$ and $\boldsymbol{\mu}$ defined by Eq. (6), we can translate \mathbf{H} and \mathbf{D} so as to write Eq. (4) as function of \mathbf{E} and \mathbf{B} .

Now we have the six equations to solve for the six components $(E_r, E_\theta, E_z, B_r, B_\theta, B_z)$. Anyway we will not solve the governing equations in this form. Once we will store the unknowns in a 1D array, we will get a system of linear equations of the form $\mathbf{Ax} = \mathbf{b}$, where \mathbf{A} will be a square sparse matrix, the *coefficient matrix*. Moreover the non-zero elements of this matrix will be the coefficient of the six field components we will solve for, in Eqs. (3) and (4).

Therefore, to make the non zero-elements of the coefficient matrix all of roughly the same order of magnitude, but also to give all the unknowns the same units $[V/m]$, we will normalize the magnetic field by multiplying by the speed of light.

We have now six normalized equations, where the new six field components, we want to solve for, are $(E_r, E_\theta, E_z, cB_r, cB_\theta, cB_z)$.

3.1.6 Boundary Conditions

The governing equations in addition to the constitutive relations form a system of two partial differential equations of the first order, which describes the wave phenomena we are interested in, strongly coupled in the unknowns, which are the two vectors \mathbf{E} and \mathbf{B} .

In order to solve these equations numerically we need to impose boundary conditions at the bounds of our computational domain, respectively $r = 0$ and $r = L$.

Let us consider the outside boundary conditions, for $r = L$. Since the first version code will work both with conducting wall boundary and PML, so as to damp the wave oscillation outside the computational domain, we want a conducting boundary condition.

The chosen conducting BC requires:

$$E_\theta = E_z = 0, \tag{7}$$

and through Eq. (3), follows that:

$$B_r = 0. \tag{8}$$

Let us now derive the inner boundary conditions, for $r = 0$. Considering that the first version of the wave code is limited to waves with $m \neq 0$ and that inside the cylindrical device we have plasma, permitting charges to move, we look for the real, physical solutions, which must be finite on axis. With these assumptions, considering the discretized equations of our theoretical model, we take the limit of the governing equations as r approaches zero, so that we get $E_z(0) = 0$; we get $B_r = 0$ and $E_\theta = 0$, respectively. In conclusion, on the axis, we get the same relations obtained for the outer BCs.

3.1.7 Representation of Waves

Considering the hypothesis regarding the physical model, we can define the general wave quantity \mathbf{W} used in the wave code, as follows:

$$\mathbf{W} = \tilde{\mathbf{W}}_c(r) e^{i(m\theta + k_z z - \omega t)} \quad , \quad (9)$$

where $\tilde{\mathbf{W}}_c$ is a complex quantity containing phase information, m is the azimuthal wave number, k_z is the axial or parallel wave number and, finally, ω is the angular frequency of the wave.

3.2 Wavecode Numerical Model

In the wave code, which we have developed and which will be briefly explained in its main functions, we have transformed Maxwell's equations into a new set of differential equations in the frequency domain and then transformed these last differential equations into a set of linear algebraic equations using finite-difference method. This procedure is basically the *finite-difference frequency-domain* (FDFD) method, which usually gives a numerical solution for problems in electromagnetism and is the simplest numerical method to implement so as to solve Maxwell's equations; it is excellent, indeed, for field visualization and modelling devices with complex geometry or structures of finite size even if it is not well-suited to computing things that involve evolution of the fields.

Moreover, since it is a frequency method, it is able to resolve sharp resonances and obtain solutions at a single frequency in a more accurate and numerically stable way than time-domain methods; its accuracy and efficiency can be improved by using non-uniform and unstructured grids.

Wavecode is a code, written in ANSI C, born to be a portable program, interfacing with other plasma codes, and because we want it to be as fast as possible and to handle the memory resources in the most efficient way. These two last aspects are needed since the *Wavecode* will be interfaced with a PIC (Particle-in-Cell) code, which will be time consuming and computationally onerous, and so avoiding to take into account other interpreted programming languages.

3.2.1 Solution Scheme

Once we get the normalized equations describing the helicon experiment, we need to solve only the spatial discretization. We will use finite differences to approximate radial derivatives; in particular, to achieve second order spatial discretization, we will use a staggered mesh of Yee type. Anyway, since we will solve numerically our set of partial differential equations by a radial discretization, we will impose the boundary conditions on the boundary grid points.

Once that the numerical values, derived from the input parameters of the helicon experiment, are introduced, the system of linear equations is thus set up and ready to be solved. Anyway it is worth noticing that \mathbf{A} is a square sparse complex matrix of order n , where n depends on the resolution we want to achieve in our computational domain.

We will make use of **MUMPS** (**M**U**M**U**M** **P** **S**) package to handle this kind of linear system.

3.2.2 Numerical Scheme

We need now to translate the solution scheme into a numerical one. The wave code is made of different blocks and each one does a different job, as highlighted in Fig. 2.

The blocks can be summarized as follows:

1. INPUT;
2. \mathbf{A} , \mathbf{b} INITIALIZATION;
3. \mathbf{A} , \mathbf{b} UPGRADE;
4. SOLVE;
5. INTERPOLATION;

6. OUTPUT;

The first block sets up the geometrical informations regarding the particular device used for the helicon experiment, the antenna informations and the plasma physical properties.

The block called "**A** , **b** INITIALIZATION" initializes the structure of the coefficient matrix **A**. Since we will use the MUMPS package to solve the linear system, we will have to implement the matrix with a structure which could be recognized by MUMPS library. The "**A** , **b** UPGRADE" block is concerned with the coefficient matrix and the column vector inputting and updating.

In "SOLVE" the system $\mathbf{Ax} = \mathbf{b}$ is solved and the MUMPS package does it in three main steps: analysis, factorization and solution. During the first step MUMPS performs an ordering, preserving sparsity of the matrix, and carries out symbolic factorization; during the next step the numerical factorization is carried out, while, in the last one, the unknowns vector \mathbf{x} is obtained.

Because of the staggered Yee mesh and because of the particular boundary conditions implemented in the wave code we will have the **E** and **B** solution, as function of the radius of the grid point in which they are calculated, throughout the whole computational domain only for a few components and so there will be some electromagnetic components with no numerical values at the boundary grid point. The interpolation, so as to have all the solution components in every grid point, is the aim of the "INTERPOLATION" block.

The last block, called "OUTPUT", attends to write the solution numerical values into the output files; the structure of the output has been decided so that they could be easily plotted and read by the *3-D Particle Code*; the *particle code*, indeed, needs the numerical values of the frequency ω and each components of the **E** and **B** fields, obtained in the first loop using a trial value for the current J , so as to stimate the new value of the current J , which will be used by the wave code in the next loop.

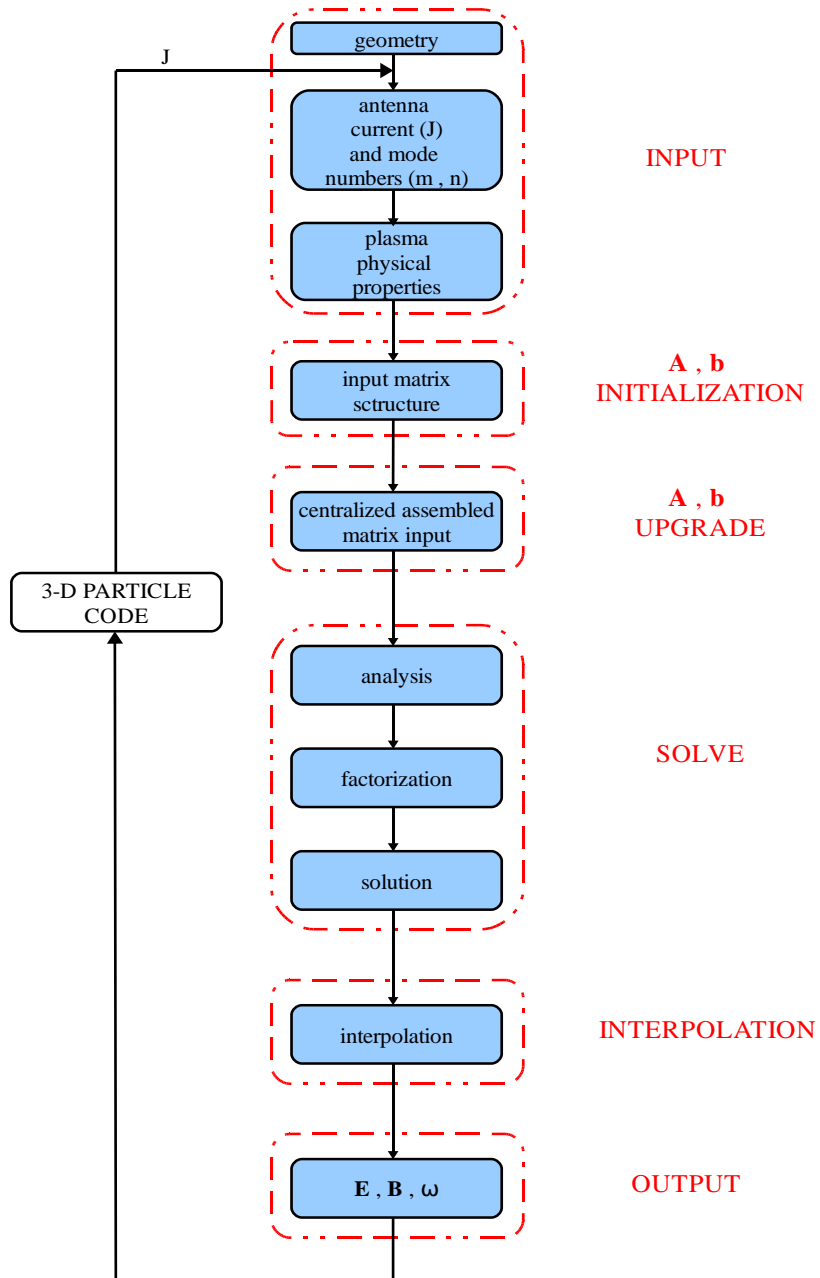


Figure 2: Numerical scheme for the simulation of the helicon experiment.

3.3 Wavecode Staggered Yee Mesh

Since our physical model is in the frequency domain, we need to implement a FDFD method and, first of all, its spatial discretization based upon the Yee lattice. Let us now consider a staggered Yee mesh.

3.3.1 3D Cartesian Coordinate System

Consider the system of partial differential equations made of Eqs. (3) and (4), defined in a cartesian coordinate system (x, y, z) . Assume that each quantity, involved in these equations, is a function of the space coordinate (x, y, z) and traslate this system into the following system of scalar equations, in the frequency domain:

$$\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} = i\omega B_x \quad (10)$$

$$\frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x} = i\omega B_y \quad (11)$$

$$\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} = i\omega B_z \quad (12)$$

$$\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} = -i\omega D_x + J_x \quad (13)$$

$$\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} = -i\omega D_y + J_y \quad (14)$$

$$\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} = -i\omega D_z + J_z \quad (15)$$

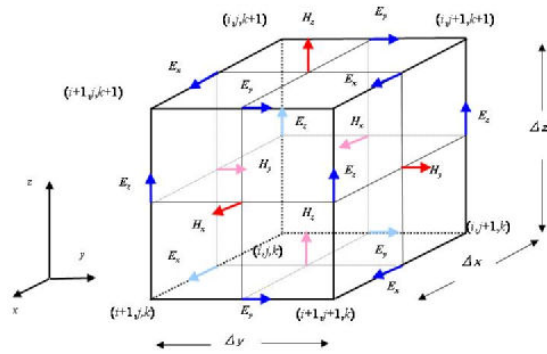


Figure 3: Illustration of a standard cartesian Yee cell about which electric and magnetic field vector components are distributed.

As one can see from Fig. 3, which represents an illustration of a standard cartesian Yee cell about which \mathbf{E} and \mathbf{H} field vector components are distributed, we denote a grid point of the space as:

$$(i, j, k) = (i\Delta x, j\Delta y, k\Delta z),$$

and for any function of space and in the frequency domain, we put:

$$F(i\Delta x, j\Delta y, k\Delta z) = F(i, j, k).$$

A set of finite difference equations for Eqs. (10) - (15) that will be suitable for perfectly conducting boundary conditions is, for Eq. (10):

$$\begin{aligned} & \frac{E_z(i, j+1, k+\frac{1}{2}) - E_z(i, j, k+\frac{1}{2})}{\Delta y} + \\ & - \frac{E_z(i, j+\frac{1}{2}, k+1) - E_y(i, j+\frac{1}{2}, k)}{\Delta z} = i\omega B_x\left(i, j+\frac{1}{2}, k+\frac{1}{2}\right). \end{aligned} \quad (16)$$

The finite difference equations corresponding to Eqs. (11) and (12) can be similarly constructed. For Eq. (13) we get:

$$\begin{aligned} & \frac{H_z(i+\frac{1}{2}, j+\frac{1}{2}, k) - H_z(i+\frac{1}{2}, j-\frac{1}{2}, k)}{\Delta y} - \\ & \frac{H_y(i+\frac{1}{2}, j, k+\frac{1}{2}) - H_y(i+\frac{1}{2}, j, k-\frac{1}{2})}{\Delta z} = -i\omega D_x\left(i+\frac{1}{2}, j, k\right) + \\ & + J_x\left(i+\frac{1}{2}, j, k\right), \end{aligned} \quad (17)$$

and, as previously, the equations corresponding to Eqs. (14) and (15) can be similarly constructed.

3.3.2 Transformation to 3D Cylindrical Coordinate System

Consider now the system of scalar equations, Eqs. (10) - (15), and translate them into the cylindrical coordinate system (r, θ, z) . Utilizing the formalism explained above, we denote a grid point of the space as:

$$(i, j, k) = (i\Delta r, j\Delta\theta, k\Delta z),$$

and for any function of space:

$$F(i\Delta r, j\Delta\theta, k\Delta z) = F(i, j, k).$$

As above, the finite difference equations corresponding to the new coordinate system can be similarly constructed.

3.3.3 Transformation to 1D Cylindrical Coordinate System

Starting from the discretized equations just derived, through a limit operation we are able to figure out the equations to be implemented into the code. Thereby, taking first the limit as θ approaches zero and then the limit as z approaches zero, we get the discretized governing equations in the monodimensional domain. Moreover the tridimensional cartesian Yee cell, illustrated in Fig. 3, becomes a monodimensional Yee line, like in Fig. 4, where one can see that it is a staggered grid since the field vector components are staggered.

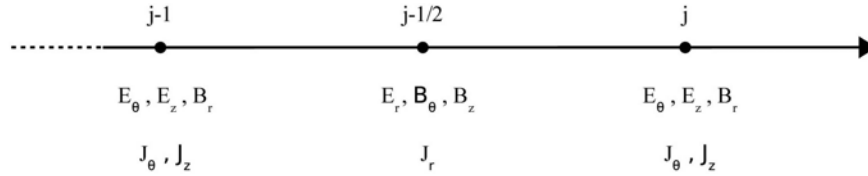


Figure 4: 1D staggered Yee mesh.

It is worth noticing that the space grid must be such that over one increment the electromagnetic field does not change significantly; this means that, to have meaningful results, the linear dimension of the grid must be only a fraction of the wavelength. Moreover, since the wave code is in the frequency domain, we do not have rigorous relations between spatial and temporal discretization to fulfill in order that the numerical stability of the code is satisfied.

3.3.4 Spatial Discretization for 1D Wave Code

Considering the assumptions made in Sec. 3.1.3 and the considerations made so far, we get the six equations for each grid point j , where we use finite differences to approximate the radial derivatives so as to achieve second order spatial discretization error:

$$(r\epsilon_r E_r)_{j-1/2} - \frac{k_z}{k_0} \left(r \frac{cB_\theta}{\mu_\theta} \right)_{j-1/2} + \frac{m}{k_0} \left(\frac{cB_z}{\mu_z} \right)_{j-1/2} = -i \frac{(rJ_r)_{j-1/2}}{\epsilon_0 \omega} \quad (18)$$

$$i \frac{r_{j-1/2}}{k_0 \Delta_r} (E_z)_{j-1} - \frac{k_0}{k_z} (rE_r)_{j-1/2} + (rcB_\theta)_{j-1/2} + -i \frac{r_{j-1/2}}{k_0 \Delta_r} (E_z)_j = 0 \quad (19)$$

$$-i\frac{1}{k_0\Delta_r}(rE_\theta)_{j-1} + \frac{m}{k_0}(E_r)_{j-1/2} + (rcB_z)_{j-1/2} + i\frac{1}{k_0\Delta_r}(rE_\theta)_j = 0 \quad (20)$$

$$(rcB_r)_j + \frac{k_z}{k_0}(rE_\theta)_j - \frac{m}{k_0}(E_z)_j = 0 \quad (21)$$

$$\begin{aligned} -i\frac{r_j}{k_0\Delta_r}\left(\frac{cB_z}{\mu_z}\right)_{j-1/2} + \frac{k_z}{k_0}\left(r\frac{cB_r}{\mu_r}\right)_j + (r\varepsilon_\theta E_\theta)_j + \\ + i\frac{r_j}{k_0\Delta_r}\left(\frac{cB_z}{\mu_z}\right)_{j+1/2} = i\frac{(rJ_\theta)_j}{\varepsilon_0\omega} \end{aligned} \quad (22)$$

$$\begin{aligned} i\frac{1}{k_0\Delta_r}\left(r\frac{cB_\theta}{\mu_\theta}\right)_{j-1/2} - \frac{m}{k_0}\left(\frac{cB_r}{\mu_r}\right)_j + (r\varepsilon_z E_z)_j + \\ - i\frac{1}{k_0\Delta_r}\left(r\frac{cB_\theta}{\mu_\theta}\right)_{j+1/2} = i\frac{(rJ_z)_j}{\varepsilon_0\omega} \end{aligned} \quad (23)$$

where $(E_r)_{j-1/2}$, $(cB_\theta)_{j-1/2}$, $(cB_z)_{j-1/2}$, $(cB_r)_j$, $(E_\theta)_j$ and $(E_z)_j$ are the six unknowns per radial grid point j , which we get after discretization; we will store these unknowns in a 1D array, packed in the order as above to get a coefficient matrix \mathbf{A} and a column vector \mathbf{b} .

3.4 Wavecode Perfectly Matched Layer

Partial difference equation based methods, which solves a PDE numerically by discretization, have had difficulties dealing with open region problems. In these methods, the computational domain has to be truncated without introducing significant artifacts into the computation.

Therefore, wave equations require something that behaves like an *absorbing boundary*, which will somehow absorb waves striking it.

In 1994, instead of finding an absorbing boundary condition, Berengere found an absorbing boundary layer, which is a layer of artificial absorbing material that is placed adjacent to the edges of the grid, completely independent of the boundary condition. When a wave enters the absorbing layer, it is attenuated by the absorption and decays exponentially; even if it reflects off the boundary, the returning wave after one round trip through the absorbing layer is exponentially tiny.

Anyway there could be a reflection at the transition zone from non-absorbing to absorbing material, but the absorbing medium could be constructed so that waves do not reflect at the interface, and so the name of *perfectly* matched layer.

This formulation can be derived in a general way, through a *complex coordinate stretching*.

3.4.1 Complex Coordinate Stretching

Considering the general formulation of a wave equation in infinite space:

$$\frac{\partial \mathbf{w}}{\partial t} = \widehat{D} \mathbf{w}$$

where $w(\mathbf{x}, t)$ is the wave function and \widehat{D} is an anti-Hermitian operator, we want to truncate space outside the region of interest in such a way as to absorb radiating waves. If we assume that the space far from the region of interest is x -invariant, linear and time invariant, we get this superposition of radiating solutions, which take the form of a superposition of planewaves:

$$\mathbf{w}(\mathbf{x}, t) = \sum_{\mathbf{k}, \omega} \mathbf{W}_{\mathbf{k}, \omega} e^{i(\mathbf{k} \cdot \mathbf{x} - \omega t)}$$

for some constant amplitudes $\mathbf{W}_{\mathbf{k}, \omega}$, where ω is the angular frequency and \mathbf{k} is the propagation constant. Assuming that $k > 0$ we will truncate the problem in the $+x$ direction. We will get this in three steps, which are:

1. In infinite space, analytically continue the solutions and equations to a complex x contour, which changes oscillating waves into exponentially decaying ones outside the region of interest without reflections.
2. In infinite space, perform a coordinate transformation to express the complex x as a function of a real coordinate. Therefore we have *real coordinates* and *complex materials*.
3. Truncate the domain of this new real coordinate inside the complex material region, in which the solution is decaying and boundary conditions won't matter.

Analytic Continuation Mapping the x coordinate to the complex coordinate \tilde{x} , we can analytically continue the wave solution, which satisfies the same differential wave equation; so, instead of evaluating x along a real axis, we have added a growing imaginary part that transforms the oscillating solution into a decaying one as the imaginary part $Im(x)$ gradually increases:

$$e^{ikx} \longrightarrow e^{ik(Re(x)+iIm(x))}.$$

This is a *reflectionless absorbing material* since it acts like an absorbing material only where $Im(x) \neq 0$ while it does not modify the solution where $Im(x) = 0$, which

means that it generates no reflections at the interface between the physical domain of interest and the absorbing material; it is *perfectly matched*. This means that a wave propagating in the region of interest does not produce any reflection when it meets the interface with the absorbing layer.

Coordinate Transformation Since it is inconvenient to solve differential equation along contours in the complex plane, we need another coordinate transformation back to real x . Expressing the complex coordinate as $\tilde{x}(x) = x + if(x)$, where $f(x)$ is some function indicating how we have deformed our contour along the imaginary axis, and denoting $\frac{df}{dx} = \frac{\sigma_x(x)}{\omega}$, the transformation of our original equation can be summed up by:

$$\frac{\partial}{\partial x} \longrightarrow \frac{1}{1 + i\frac{\sigma_x(x)}{\omega}} \frac{\partial}{\partial x} \quad (24)$$

$$x \longrightarrow x + \frac{i}{\omega} \int^x \sigma(x') dx' \quad (25)$$

where σ is often called the *PML absorption coefficient*, and it is pointed as the greek letter *sigma* because it plays a role of a *conductivity*.

The PML regions are located where $\sigma_x(x) > 0$, and inside this regions our oscillating solutions turn into exponentially decaying ones. In the regions where $\sigma_x(x) = 0$ our wave equation is unchanged and so also our solution. Moreover it is worth noticing that $\frac{df}{dx} = \frac{\sigma_x(x)}{\omega}$ and not just $\frac{df}{dx} = \sigma_x(x)$, so that

$$e^{ikx} e^{-\frac{k}{\omega} \int^x \sigma_x(x') dx'} , \quad (26)$$

and so that the attenuation rate in the PML is independent of angular frequency ω and therefore all wavelengths decay at the same rate.

Truncating the computational domain Now we can truncate the computational region regardless the boundary conditions, because only the tiny exponential tails encounter this conditions and could reflect on them, and even they attenuate on the way back to the region of interest, the effect on the solutions in the region of interest will be exponentially small.

3.4.2 Wavecode Implementation

Regarding the code developed, the analytically continuation of the governing equations is obtained through the complex coordinate stretching for a cylindrical geometry, as explained above, but, since in the wavecode only the radial dependence

is taken into account, the radial variable r is transformed into the complex plane, through the following relations:

$$\tilde{r} \longrightarrow r + \frac{i}{\omega} \int_r \sigma(\xi) d\xi \quad (27)$$

$$\frac{\partial}{\partial \tilde{r}} \longrightarrow \frac{1}{1 + i \frac{\sigma(r)}{\omega}} \frac{\partial}{\partial r} \quad (28)$$

where \tilde{r} is the radius coordinate in the complex plane, r' is a dummy variable and $\sigma(r)$ is a user-defined function which accounts of how the contour has been deformed, the attenuation rate of the PML and finally the reflectionless of the PML itself.

The wavecode is also in the frequency domain and so we do not have to turn the equations back in the time domain and so avoiding to implement the $\frac{1}{\omega}$ dependence, used to make frequency-independent the attenuation rate, directly in the time domain via Fourier anti-transform and auxiliary differential equation approach.

The six equations for a generic grid point j become:

$$(\tilde{r}\varepsilon E_r)_{j-1/2} - \frac{k_z}{k_0} \left(\tilde{r} \frac{cB_\theta}{\mu_\theta} \right)_{j-1/2} + \frac{m}{k_0} \left(\frac{cB_z}{\mu_z} \right)_{j-1/2} = -i \frac{(\tilde{r}J_r)_{j-1/2}}{\varepsilon_0\omega} \quad (29)$$

$$\begin{aligned} i \frac{\tilde{r}_{j-1/2}}{k_0 \Delta r} (E_z)_{j-1} - \left(1 + i \frac{\sigma(r)}{\omega} \right) \frac{k_z}{k_0} (\tilde{r}E_r)_{j-1/2} + \\ + \left(1 + i \frac{\sigma(r)}{\omega} \right) (\tilde{r}cB_\theta)_{j-1/2} - i \frac{\tilde{r}_{j-1/2}}{k_0 \Delta r} (E_z)_j = 0 \end{aligned} \quad (30)$$

$$\begin{aligned} -i \frac{1}{k_0 \Delta r} (\tilde{r}E_\theta)_{j-1} + \left(1 + i \frac{\sigma(r)}{\omega} \right) \frac{m}{k_0} (E_r)_{j-1/2} + \\ + \left(1 + i \frac{\sigma(r)}{\omega} \right) (\tilde{r}cB_z)_{j-1/2} + i \frac{1}{k_0 \Delta r} (\tilde{r}E_\theta)_j = 0 \end{aligned} \quad (31)$$

$$(\tilde{r}cB_r)_j + \frac{k_z}{k_0} (\tilde{r}E_\theta)_j - \frac{m}{k_0} (E_z)_j = 0 \quad (32)$$

$$\begin{aligned} -i \frac{\tilde{r}_j}{k_0 \Delta r} \left(\frac{cB_z}{\mu_z} \right)_{j-1/2} + \left(1 + i \frac{\sigma(r)}{\omega} \right) \frac{k_z}{k_0} \left(\tilde{r} \frac{cB_r}{\mu_r} \right)_j + \left(1 + i \frac{\sigma(r)}{\omega} \right) \\ (\tilde{r}\varepsilon_\theta E_\theta)_j + i \frac{\tilde{r}_j}{k_0 \Delta r} \left(\frac{cB_z}{\mu_z} \right)_{j+1/2} = -i \left(1 + i \frac{\sigma(r)}{\omega} \right) \frac{(\tilde{r}J_\theta)_j}{\varepsilon_0\omega} \end{aligned} \quad (33)$$

$$\begin{aligned}
& i \frac{1}{k_0 \Delta r} \left(\tilde{r} \frac{cB_\theta}{\mu_\theta} \right)_{j-1/2} - \left(1 + i \frac{\sigma(r)}{\omega} \right) \frac{m}{k_0} \left(\frac{cB_r}{\mu_r} \right)_j + \left(1 + i \frac{\sigma(r)}{\omega} \right) (\tilde{r} \varepsilon_z E_z)_j \\
& - i \frac{1}{k_0 \Delta r} \left(\tilde{r} \frac{cB_\theta}{\mu_\theta} \right)_{j+1/2} = -i \left(1 + i \frac{\sigma(r)}{\omega} \right) \frac{(\tilde{r} J_z)_j}{\varepsilon_0 \omega}
\end{aligned} \tag{34}$$

where $\tilde{r} = r + \frac{i}{\omega} \int_{R_i}^r \sigma(\xi) d\xi$ and $\sigma(r)$ is defined as:

$$\sigma(r) = \begin{cases} 0 & \text{if } 0 \leq r < R_i, \\ \frac{\sigma_{max}}{l_{PML}^p} (r - R_i)^p & \text{if } r \geq R_i. \end{cases} \tag{35}$$

where r is the radius of the grid point considered, R_i is the radius at which there is the interface between the region of interest and the PML absorbing layer, p is the order of the polynomial variation, σ_{max} is an absorption coefficient which influences the attenuation rate of the PML and the slope of the deformed contour in the complex plane and, finally, l_{PML} is the thickness of the PML cylindrical layer. Berenger found that if $\sigma(r)$ is constant throughout the PML media significant reflections are encountered at the interface of the PML. This is due to the discrete approximation of the fields and the material parameters of the PML layer at the interface between the two media, which results in a spurious impedance loading just at the interface. This problem in the discrete space can be tempered by using a gradually spatially varying $\sigma(r)$ function. In the wavecode we turn on $\sigma(r)$ quadratically ($p = 2$) from zero, over a region of length a half wavelength.

There are three fundamental parameters to be set to decide the PML behaviour:

1. the order of the polynomial variation (p);
2. the absorption coefficient (σ_{max});
3. the thickness of the PML cylindrical layer (l_{PML}).

The first one takes into account the slope of the deformed contour and so also the variation of the PML medium; a simple quadratic turn-on of the PML has been chosen so as to have a variation as smooth as possible.

The second one is related not only with the variation of the PML medium but mainly with the attenuation rate; in fact, it is closely related to the reflection error. One then expects that by increasing this parameter, the reflection error will continuously reduce; however, due to discretization error and numerical rounding error, there are a lower and upper bound, inside of which an optimal value for σ_{max} can be found, so as to minimize the reflection error.

Finally, the thickness of the PML medium layer is necessary to obtain a finite computational domain, even if it means introducing some errors. Its value is closely related to the wavelength inside the region of interest and so also with the frequency of our wave problem. It is strictly coupled with the σ_{max} parameter because one can make the PML layer as thin as one want, just by making σ_{max} very large, but, as said above, using a very large σ_{max} can cause numerical reflections into the spatial discretized problem.

The implementation of the PML formulas does not change the mathematical form of the equations for each grid point, even if it deeply changes the coefficients of discretized Maxwell's equations and so the coefficient matrix.

3.5 Wavecode Upgrade

Starting from the previous version of Wavecode, a few features of the code have been improved so far. First of all, the electric permittivity tensor ϵ is no more a diagonal tensor, but now it includes also the non-diagonal terms and so it becomes:

$$\epsilon = \epsilon_0 \begin{bmatrix} K_1 & K_3 & 0 \\ -K_3 & K_1 & 0 \\ 0 & 0 & K_2 \end{bmatrix}, \quad (36)$$

while the magnetic permeability tensor μ remains the same. Initially, K_1 , K_2 and K_3 are functions of the antenna frequency ω , the plasma frequency ω_p and the electron cyclotron frequency ω_c ; then, since it is a trivial generalization, collisions have been added through the collision frequency parameter ν .

Then the governing equations have been changed so as to improve the ability of the code to simulate plasma waves with $m = 0$ and to avoid boundary conditions on the axis; this means a slight change in the coefficient matrix. Moreover, since these new governing equations don't use any on-axis fields, we no longer pretend to know what the on-axis fields are (through inner BCs) and we won't solve for them either. We will use linear extrapolation to compute them after we have solved for the fields (with second-order error) on the interior grid points.

This new settings allow us to simulate not only classical helicon waves in a better way, but also Trivelpiece-Gould waves. Last but not least, we are now able to evaluate numerically the generalized dispersion relationship for annularly bounded helicon plasma and compare the numerical results with the available analytical ones. Actually, new test cases have been investigated and the *validation & verify* of this new version of the code is currently under development.

4 Wavecode Validation & Verify

The aim of this kind of test is to reproduce a purely right propagating wave with zero reflections at the boundary of the computational domain; a particular physical case to simulate is not necessary, but just only a simple numerical test so as to be sure that the PML is working properly. The following assumptions are made:

1. vacuum medium;
2. azimuthal current impulse;
3. small antenna approximation;
4. finite boundary conditions.

These assumptions have some important consequences which strongly simplify Maxwell's equations; primarily, the first assumption simplify the permittivity and permeability tensors, while the second one involves an azimuthal current density impulse (modelled through the Dirac impulse) which excites the propagating wave and simplify the azimuthal and axial mode numbers, which become $m = k_z = 0$. Moreover the small antenna approximation means that:

$$\lambda \gg L. \tag{37}$$

Anyway the last aspect, highlight by Eq. (37), has particular effects in the implementation in the Wavecode, because λ on the one hand is strictly linked with the angular frequency ω , while, on the other hand, it is strictly related to the PML layer; therefore, since the absorbing PML layer impose $l_{PML} \ll L$ and $l_{PML} = \frac{\lambda}{2}$, while the small antenna approximation impose Eq. (37), using the relation $\lambda f = c$ for waves of frequency f propagating in vacuum, we get:

$$r_a \ll \lambda \ll 2L, \tag{38}$$

where r_a is the antenna radius from the azimuthal current density impulse propagates. Finally, in order to solve this reduced problem, boundary conditions at the bounds of the computational domain, respectively $r = 0$ and $r = L$, are to be imposed.

Consider the outside boundary conditions, for $r = L$. Since a PML will damp the wave oscillation outside the computational domain, a conducting boundary condition

outside of it is necessary so that the wave gets reflected on the conducting boundary and passes back through the PML. Otherwise, if an open (radiation) boundary condition had been used with the PML, the PML layer should have been twice as thick to get the same damping as with the conducting boundary condition.

The chosen conducting BC requires:

$$E_\theta = E_z = 0, \quad (39)$$

and the same BCs will be implemented on the axis of the cylindrical device, so as to solve the simplest numerical problem. The previous assumptions are now translated into numerical conditions and constraints which can be directly implemented into the Wavecode so as to get the expected numerical results.

In order to solve the same reduced problem in a different way, the same assumptions have been used to simplify Maxwell's equations; this new set of equations has been solved partially in an analytical way, through Bessel's functions, and partially in a numerical way.

Since the last approach deals with the solution of a boundary-value problem, which cannot be solved by Runge-Kutta or other such methods, that has been dealt with the *shooting method*.

4.0.1 Results

The campaigns of numerical simulations with the Wavecode produced these results, in which are presented both the real and imaginary part of each electromagnetic field, and obtained with $N = 10^5$ grid points. The same field components are obtained with the numerical approach implemented via the shooting method. At the end, the comparison between the results obtained by the Wavecode and by the shooting method is presented so as to show not only the accuracy of the Wavecode as a numerical code to simulate helicon waves in a plasma source, but also that the PML works properly, avoiding waves from bouncing back from the artificial domain boundary and numerically pollute the solution.

Comparison

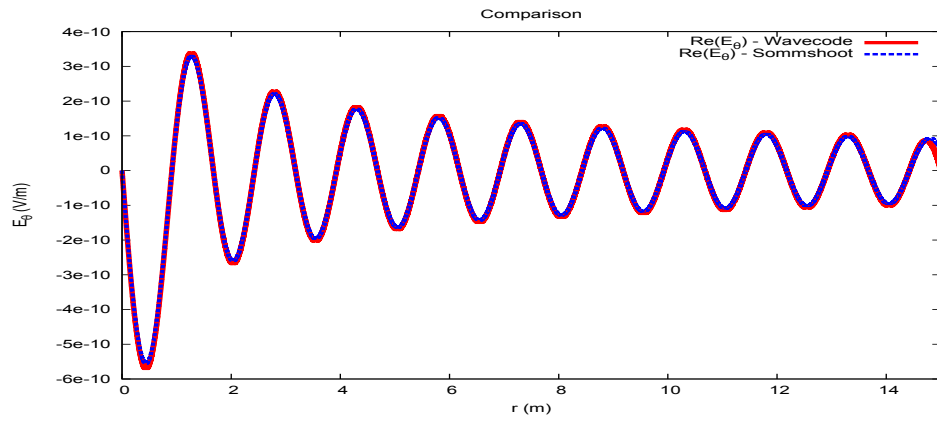


Figure 5: Comparison of the real part of the azimuthal component of the electric field E_θ [V/m], between Wavecode and the shooting method.

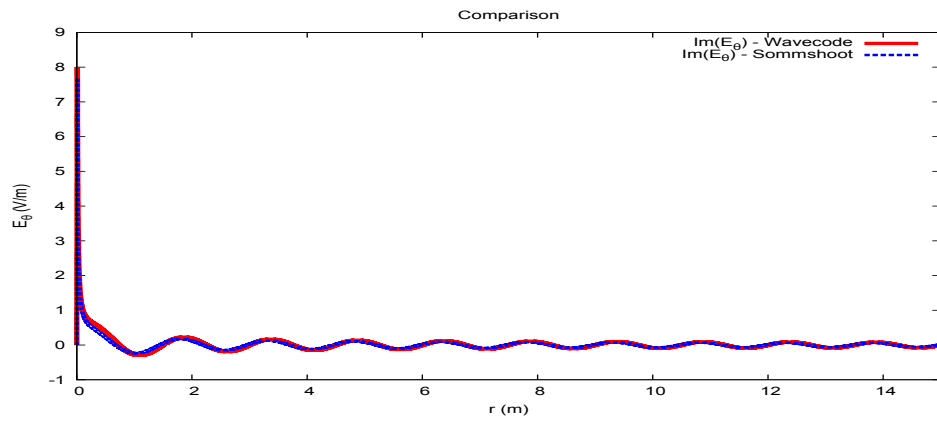


Figure 6: Comparison of the imaginary part of the azimuthal component of the electric field E_θ [V/m], between Wavecode and the shooting method.

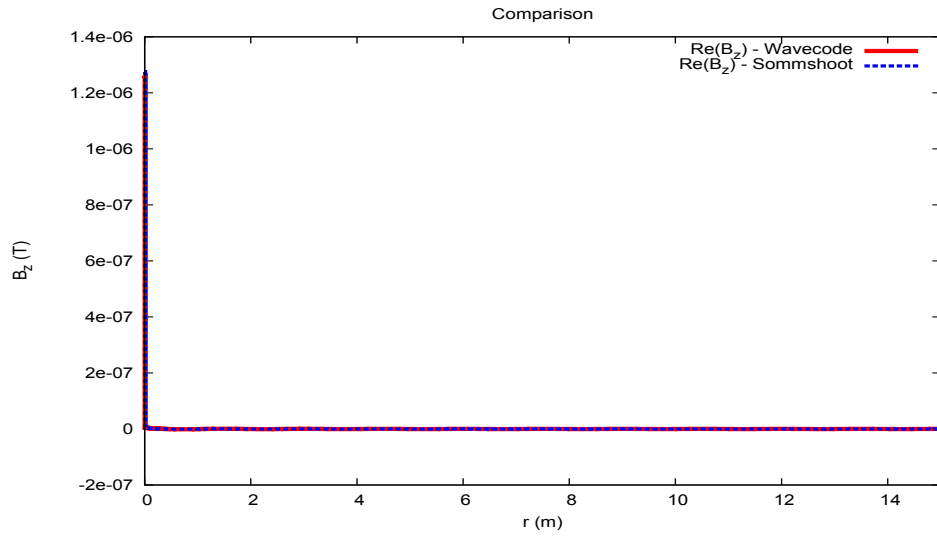


Figure 7: Comparison of the real part of the axial component of the magnetic field B_z [T], between Wavecode and the shooting method.

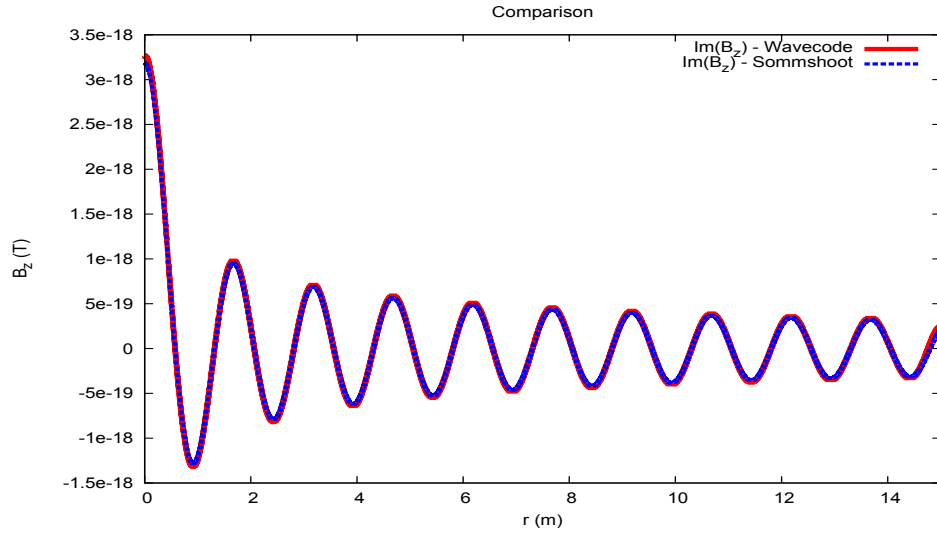


Figure 8: Comparison of the imaginary part of the axial component of the magnetic field B_z [T], between Wavecode and the shooting method.

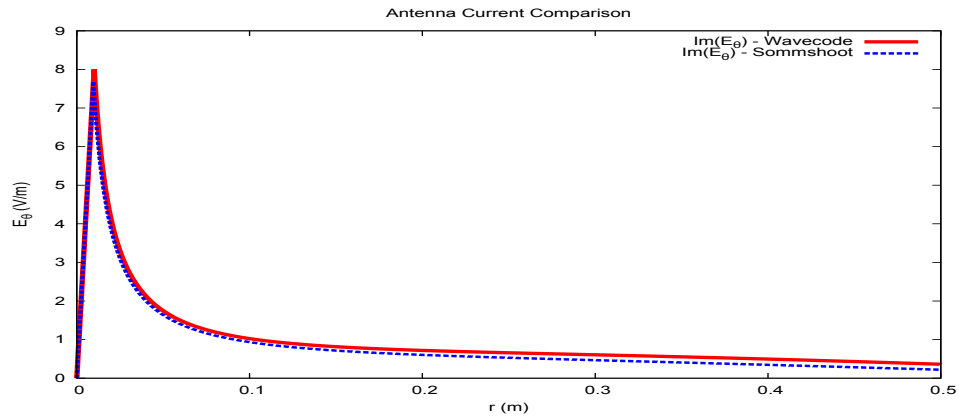


Figure 9: Comparison of the current impulse, as shown by the imaginary part of the azimuthal component of the electric field E_θ [V/m], between Wavecode and the shooting method.

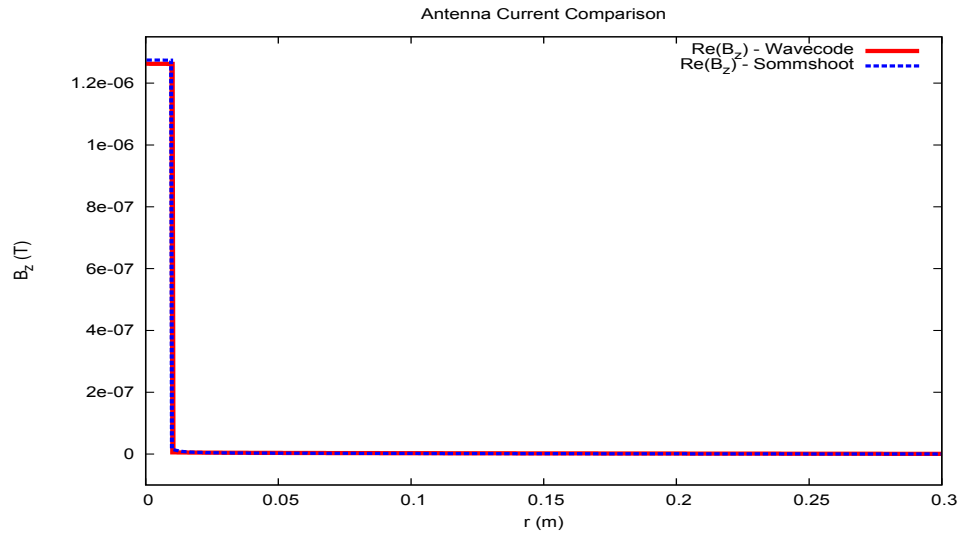


Figure 10: Comparison of the current impulse, shown by the real part of the axial component of the magnetic field B_z [T], between Wavecode and the shooting method.

For the last Figs. 9 and 10, one can see how the current impulse, used to excite a right propagating wave, is split into an imaginary electrostatic wave and a real magnetic wave, respectively.

Otherwise, considering Figs. 5, 6, 7 and 8, it is worth noticing that the match, between the Wavecode wave solution and the shooting method one, suggest that the Perfectly Matched Layer is working properly; the evidence is that we use two completely independent ways to calculate the solution for a simple test case with an open outer boundary and get the same answer. This means that the wave code with PML is thereby verified.

5 Particle Code

The Particle Code is an ANSI C code whose purpose is to follow charged particles orbits subjected to RF electromagnetic fields in a cylindrical geometry. The code is completely 3D, and features a radial finite volumes solver for the self-consistent electrostatic field using a Particle in Cell approach. The code computes the resulting plasma current density, partially Fourier transformed along the time, azimuthal, and axial directions so the 1D Wavecode can use it as a source term in Ampere's circuital Law. ANSI C has been chosen for its execution speed and for the availability of portable open source libraries that implement advanced numerical methods. The code workflow is shown in Fig. 11

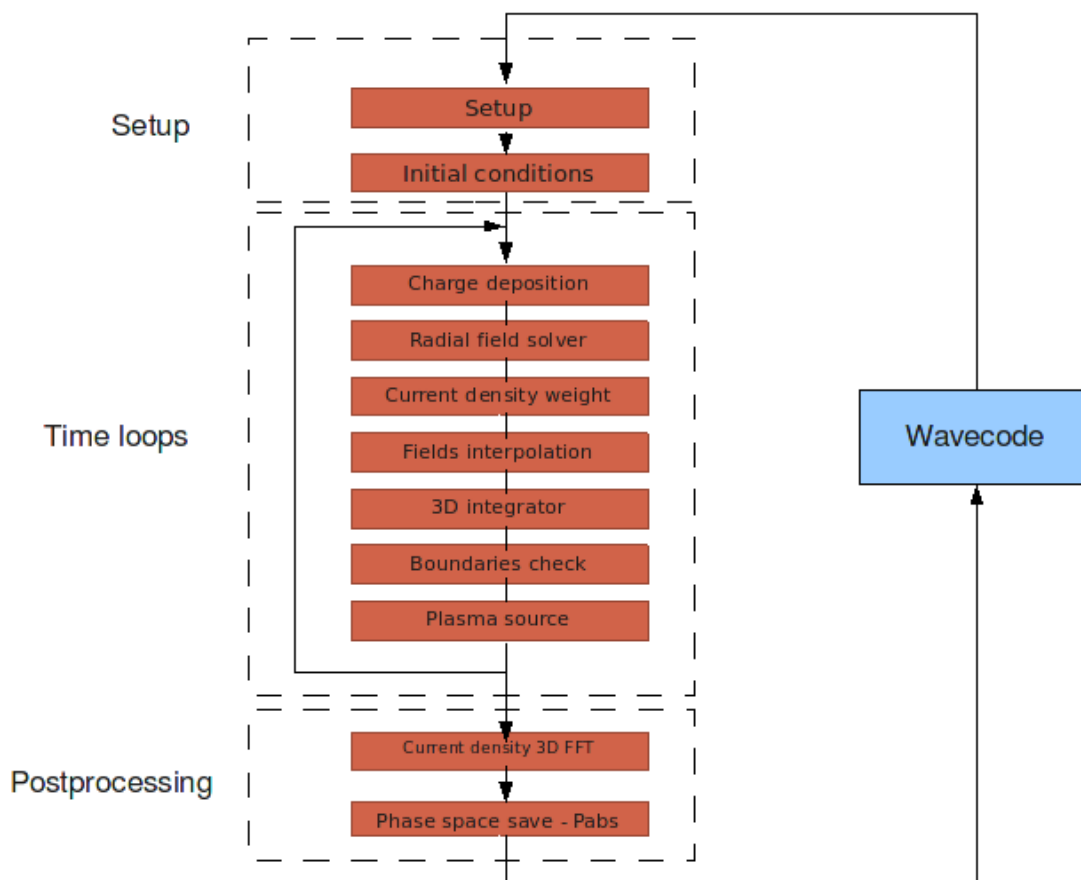


Figure 11: Particle code flow

The code is composed of the following sections:

- a SETUP section, necessary to read the RF fields from the Wavecode and an input file containing inputs necessary only for the Particle Code. In this section memory is dynamically allocated for grids, particles and species. Several simulation parameters are defined, such as the timestep definition and the ratio between physical to computational particles (to improve execution speed and simulate the high densities in plasmas, every particle in the simulation is supposed to be a collection of physical particles close enough that they are subjected to the same electromagnetic force). The particles state vectors (positions and velocities) are loaded in order to have an uniform initial density distribution and a Maxwell-Boltzmann speed distribution.
- an iterative TIME LOOP section: in every time loop the time is supposed to be constant and is the product of the current time loop iteration and a timestep, which can be an input or can be automatically computed based on the main plasma characteristic frequencies. In each time loop current densities are computed on a 3D cylindrical grid and logged for further postprocessing, the self-consistent electrostatic field is computed, fields from the wavecode and from the coils confinement axial magnetic field are interpolated to particles positions, particles (subjected to the Lorentz force) positions and velocities are advanced in time and are checked for exiting the domain boundaries. Finally, new particles are reinjected simulating a constant plasma source rate.
- a POSTPROCESSING section, where the current density is processed by an FFT code so that it can be used by the 1D Wavecode, the particles phase space is saved so the simulation can be restarted from the same point with different RF fields, and power absorbed by the plasma is computed.

5.1 Startup section

At the beginning of the simulation, the code looks through the input text file in order to search for the number of species to be included in the simulation. The Particle Code main input file is an ASCII file read by a custom parser to initialize the physical parameters of the simulation. The use of an input file is needed foreseeing the Codes Driver scanning all combinations of physical parameters that are supposed to influence the power absorption. The code makes extensive use of dynamic memory allocation, allowing flexibility in changing physical parameters without the need to recompile the code. The inputs for the Particle Code (in SI units) are:

- Number of species
- Charge, mass, density, temperature for each species
- Plasma source radius
- Number of the wave periods to observe
- Time needed to reach steady state
- Total number of particles to use in the simulation
- Wave angular frequency, axial angular wavenumber and azimuthal wavenumber
- Axial confinement magnetic field
- Radial, azimuthal and axial grid points for the current density computation

The code is axially periodic, so the device length L is computed in a way that contains an integer number of waves in axial direction.

5.2 Physical to Computational ratio

Every charged particle in the Particle Code is thought as a collection of real particles, whose positions are close enough to feel the same external force. This is necessary because it is otherwise impossible for computers to simulate all the charged particles in a plasma in a reasonable time. The number of physical particles contained in each computational particle is called *particle weight*, named p_w from here. The mass of a computer particle is the product of the real particle one and p_w , and so is the charge of the computer particle. This approach does not affect the motion of computational particles, that can be thought as the motion of a group consisting of p_w particles close enough to each other so they are subjected to the same electromagnetic force. But, it affects current density and charge density because every computational particle carries more charge than a physical particle. It is important to keep p_w as low as one can, because higher values could mean more noise in the results. This can be lowered increasing the number of particles in the simulation. It is necessary to assign the correct p_w computing the total volume of the cylindrical device:

$$V = \pi a^2 l$$

Where a is the plasma source radius, and l is the device length. Then, the particle density of each species is multiplied by the device volume and then summed, to obtain the total number of elementary particles that theoretically are contained in the volume:

$$n_{part} = \sum_{s=1}^{n_s} n_i V$$

Where n_s is the number of species, n_i the density for that species and V the device volume. The initial total number of particles in the simulation is given in the input file and is fixed. The number of computational particles for each species is then:

$$n_{part,si} = \frac{n_i V n_{tot}}{n_{part}}$$

Finally, p_w is computed as:

$$p_w = \frac{n_{part}}{n_{tot}}$$

5.3 Interface with WaveCode

The Wavecode output files (one for each component of the RF fields) are read and memory is allocated for data contained in it. For every radial position, these files contain the real part of the external fields and the imaginary part. The wavecode files have the layout in the following table:

radius	Re (field)	Im (field)
--------	------------	------------

Table 1: Wavecode output format

Files are read to count the number of radial nodes the wavecode used to compute the RF fields inside the device and then to allocate memory for vectors containing:

- radial values for i wavecode radial grid points: E_θ, E_z, B_r
- radial values for $i + \frac{1}{2}$ wavecode radial grid points: E_r, B_θ, B_z

- magnitude and phase for each of the six fields

Magnitude and phase are then computed, being the fields from the wavecode rotating vectors as in Fig. 12 in the complex plane with angular frequency $-\omega$, azimuthal mode number m and axial mode number k :

$$\begin{aligned} \text{magnitude} &= \sqrt{\text{Re}^2 + \text{Im}^2} \\ \text{phase} &= \text{atan2}\left(\frac{\text{Im}}{\text{Re}}\right) \end{aligned}$$

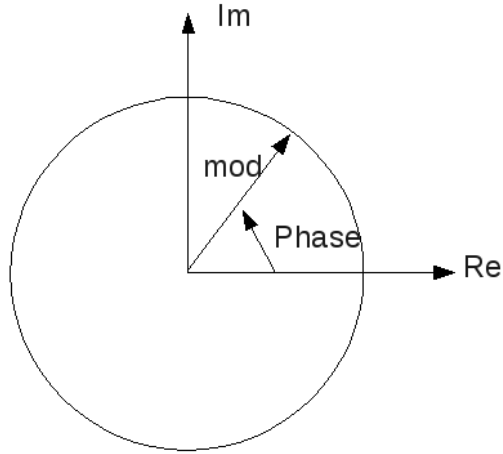


Figure 12: Wavecode fields rotating vectors in the complex plane

5.3.1 Current density weighting

The Wavecode needs the plasma current density as an input to solve Ampere's circuital law:

$$\nabla \times \vec{B} = \mu \vec{J} \quad (40)$$

In which \vec{J} is the sum between the Antenna Current \vec{J}_a from the Helicon antenna, and the Plasma current \vec{J}_p :

$$\vec{J} = \vec{J}_a + \vec{J}_p$$

The Wavecode, however, is a 1-D (radial) code and relies on *modes* along the azimuthal and axial direction. Thus the Particle Code has to compute the current

density on a grid in (r, θ, z, t) and then partially fourier transform the results along 2 spatial dimension θ, z and time t , to give in output:

$$\vec{J}(r, \theta, z, t) \rightarrow \vec{J}(r, m, k, \omega) \quad (41)$$

where m is the azimuthal mode number, k is the axial mode number and ω is the angular frequency. Current density is defined as:

$$\vec{J} = nq\vec{v} \quad (42)$$

Where n is the density, q is the electric charge and \vec{v} is the velocity.

It is necessary to obtain the current density on equally spaced discrete grid points on a 3D spatial cylindrical grid, to allow for the execution of Fast Fourier Transforms. So, at the end of each time loop the 3D spatial grid will contain J_r, J_θ, J_z . J_r is computed on a subgrid that is staggered radially. The result is then known on radial grid points compatible with the Wavemode Yee grid, which is radially staggered and assumes that J_θ, J_z are known at grid point i , while J_r is known at grid points $i + \frac{1}{2}$. In general, particles are not located exactly on grid points so it's necessary to use a weighting factor for the current density from particles positions to grid points. The cylindrical domain has been divided in:

- n_r grid points along the radial direction
- n_θ grid points along the azimuthal direction
- n_z grid points along the axial direction

This creates a spatial grid on which current densities will be weighted on each timestep fro each time loop. The weighting used is named *volume weighting*. Every particle carries a $p_w q \vec{v}$ quantity, that can be written along r, θ, z to obtain:

$$Q_{v,r} = p_w q v_r$$

$$Q_{v,\theta} = p_w q v_\theta$$

$$Q_{v,z} = p_w q v_z$$

Let's call $Q_{v,f} = p_w q v_f$ the quantity along the f direction Each particle is located inside one cell of the 3D grid. The code looks in which cell the particle is, and assigns the three quantities above to the 3D spatial grid. Each of these eight quantities is weighted to eight nodes of the corresponding 3D spatial grid, using volume weighting.

We call i, j, k the cell indexes (that is, the highest integer cell index lower than the particle position) in which the particle is located along the three directions.

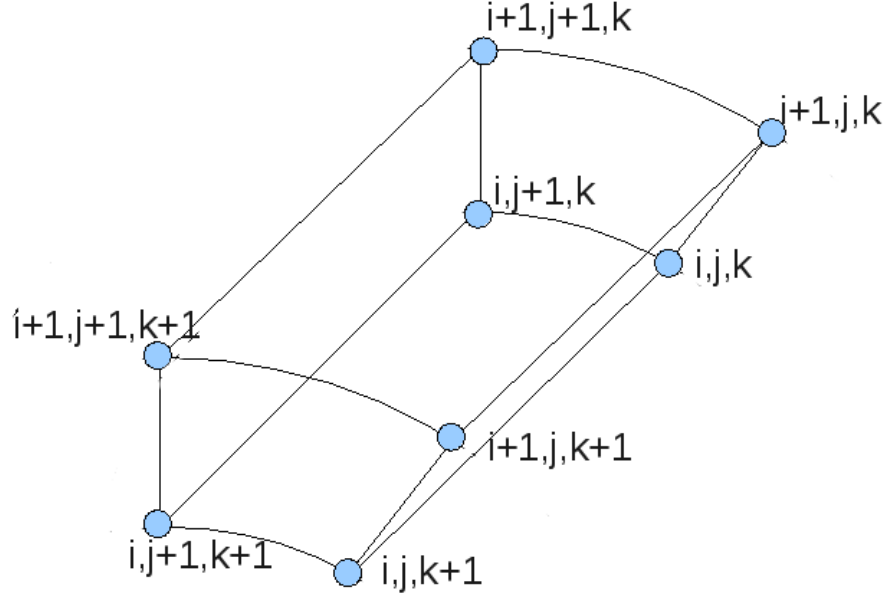


Figure 13: Nodes surrounding the particle

The volume of each cell is computed as:

$$V_{cell} = \frac{1}{n_\theta} \pi (r_{i+1}^2 - r_i^2) \Delta_z \quad (43)$$

where Δ_z is the device axial length divided by $n_z - 1$. Then, quantities are weighted to the grid points using the following functions:

$$Q_{v,f,i,j,k} = Q_{v,f} \frac{\pi (r_{i+1}^2 - r^2) \frac{(\theta_{j+1} - \theta)}{2\pi} (z_{k+1} - z)}{V_{cell}} \quad (44)$$

$$Q_{v,f,i,j,k+1} = Q_{v,f} \frac{\pi (r_{i+1}^2 - r^2) \frac{(\theta_{j+1} - \theta)}{2\pi} (z - z_k)}{V_{cell}} \quad (45)$$

$$Q_{v,f,i,j+1,k} = Q_{v,f} \frac{\pi (r_{i+1}^2 - r^2) \frac{(\theta - \theta_j)}{2\pi} (z_{k+1} - z)}{V_{cell}} \quad (46)$$

$$Q_{v,f,i,j+1,k+1} = Q_{v,f} \frac{\pi (r_{i+1}^2 - r^2) \frac{(\theta - \theta_j)}{2\pi} (z - z_k)}{V_{cell}} \quad (47)$$

$$Q_{v,f,i+1,j,k} = Q_{v,f} \frac{\pi (r_{i+1}^2 - r^2) \frac{(\theta_{j+1}-\theta)}{2\pi} (z_{k+1} - z)}{V_{cell}} \quad (48)$$

$$Q_{v,f,i+1,j,k+1} = Q_{v,f} \frac{\pi (r_{i+1}^2 - r^2) \frac{(\theta_{j+1}-\theta)}{2\pi} (z - z_k)}{V_{cell}} \quad (49)$$

$$Q_{v,f,i+1,j+1,k} = Q_{v,f} \frac{\pi (r_{i+1}^2 - r^2) \frac{(\theta-\theta_j)}{2\pi} (z_{k+1} - z)}{V_{cell}} \quad (50)$$

$$Q_{v,f,i+1,j+1,k+1} = Q_{v,f} \frac{\pi (r_{i+1}^2 - r^2) \frac{(\theta-\theta_j)}{2\pi} (z - z_k)}{V_{cell}} \quad (51)$$

It should be noted that the sum of all the weighting functions is equal to one, and more of $Q_{v,f}$ is weighted to the nearest grid points. This approach equals the assumption of particles being rigid particles clouds that partially overlap the surrounding eight grid points. After this is done for all particles, we have the total $Q_{v,f}$ for each grid point. To obtain the current density, we divide these variables by the cell volumes around each grid point, whose are not the same as the cell volume V_{cell} defined for the weighting, but are staggered by $\frac{\Delta_r}{2}, \frac{\Delta_\theta}{2}, \frac{\Delta_z}{2}$: this way the volumes are centered on the grid points. For instance the current density at r, i, j, k is:

$$J_{r,i,j,k} = \frac{Q_{v,r,i,j,k}}{\pi \left(r_{i+\frac{1}{2}}^2 - r_{i-\frac{1}{2}}^2 \right) \frac{(\theta_{j+\frac{1}{2}} - \theta_{j-\frac{1}{2}})}{2\pi} (z_{k+\frac{1}{2}} - z_{k-\frac{1}{2}})} \quad (52)$$

After this is done, we have the current density components on a 3D spatial grid in each of the three directions (radial, azimuthal, axial) in a timestep. For every timestep and every discretized radius, a temporary file is saved, containing the current density values on a 2D spatial grid θ, z (named shell) at each discretized radial value. The code, after the time loops and using the FFTW library, reads the data of all the timesteps from the 2D spatial grid, thus building a 3D grid t, θ, z for each radius, and performs a 3D FFT on this grid, obtaining the desired $\vec{J}(r, m, k, \omega)$ where m is the azimuthal mode number and k is the axial mode number. These results are then written in an ASCII file so the Wavecode can read them.

5.4 Load particles distributions

The loading strategy differ depending on the iteration number of the coupled codes:

- If the Particle Code is launched for the first time in the current test case, the phase space is loaded using a custom function assuming uniform density distribution and Maxwellian speed distribution
- If at least one iteration of the two codes has been performed for the current test case, the loaded phase space for particles will be the phase space at the end of the previous codes iteration, reading it from a file.

This strategy allows to increase convergence rate of the coupled codes and to decrease numerical instabilities. The custom function used in the first loading is called "Quiet Loader", because it loads a phase space consisting of a quiet Maxwellian distribution with uniform density distribution. Its purpose is to fill particles state vectors, given the initial density and temperature. The state vector is a six-dimensional vector that stores particles positions and velocities:

$$(v_r, v_\theta, v_z, r, \theta, z) \quad (53)$$

This could be implemented using a random number generator. In computer implementations by the way, random generators are not so random: they produce, in the case of particle density, zones with higher or lower density, with particles grouped in narrow regions of the control volume. This problem is partially solved by using *Quasi-Random* numbers generators. These are deterministic sequences that generate numbers which maintain low the discrepancy. Discrepancy measures how uniformly distributed the point set is. A lower discrepancy means that the points distribution approaches an uniform distribution in a better way. Given that the state vector is a six-dimensional vector, we need to generate uncorrelated points with an uniform distribution in a six-dimensional space. For the six-dimensional case, the Halton sequence generates points in a 6D hypercube having unitary edge length, so in the interval $[0, 1)^6$. The Halton sequence is built according to a deterministic method that uses a prime number as its base. Each non negative integer k (which can be thought as a particle identifier) can be expanded using a prime base p :

$$k = a_0 + a_1p + a_2p^2 + \dots + a_r p^r \quad (54)$$

where each a_i is an integer in $[0, p - 1]$. Now define a function Φ_p of k by:

$$\Phi_p(k) = \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \dots + \frac{a_r}{p^{r+1}} \quad (55)$$

If $p = 2$, the sequence $\Phi_2(k)$ is called the *Van der Corput sequence*. Let 6 be the dimension of the space to be sampled. Any sequence $p_1, p_2, p_3, p_4, p_5, p_6$ of prime numbers defines a sequence

$$\Phi_{p_1}(k), \Phi_{p_2}(k), \Phi_{p_3}(k), \Phi_{p_4}(k), \Phi_{p_5}(k), \Phi_{p_6}(k)$$

of functions, and then the 6-dimensional Halton point is:

$$(\Phi_{p_1}(k), \Phi_{p_2}(k), \Phi_{p_3}(k), \Phi_{p_4}(k), \Phi_{p_5}(k), \Phi_{p_6}(k)) \quad (56)$$

$$\text{for } k = 0, 1, 2, \dots, n - 1$$

Where $p_1 < p_2 < p_3 < p_4 < p_5 < p_6$ are prime numbers and n is the total number of Halton points (coinciding with the number of initial particles) to be generated. To evaluate the function $\Phi_p(k)$ one can use this algorithm:

```

p' = p, k' = k, Phi = 0
while k' > 0 do
  a = k' mod p
  Phi = Phi + a / p'
  k' = int(k' / p)
  p' = p'p

```

Where $\text{int}(x)$ is the integer part of x , and p is the prime base. Even though standard Halton sequences perform very well in low dimensions, correlation problems have been noted between sequences generated from higher primes (Vandewoestyne Cools, 2006). For example if we start a 2D sequence with the primes 11 and 13, the first 11 pairs of points would have perfect linear correlation. Halton sequences are indeed known to have more correlation between points generated from higher primes, so for our 6-dimensional case, we should choose the lowest prime numbers sequence possible, that is: $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11, p_6 = 13$. However, there is still correlation between 2-d projections from the higher dimensions (e.g. from dimensions 5 and 6 and thus between initial azimuthal and axial position, using prime numbers 11 and 13)

A recent algorithm has been implemented in this work to reduce the *L2-discrepancy*, which is the L2 distance in a six-dimensional space between an uniform distribution and the effective generated points distribution. The correlation between points of the Halton sequence can be broken by scrambling the digits of the sequence in a

way that preserves the low-discrepancy properties, defining the scrambled radical inverse function $S_b(k)$ in analogy with Eq. (55) as:

$$S_b(k) = \frac{\pi_b(d_0)}{p} + \frac{\pi_b(d_1)}{p^2} + \dots + \frac{\pi_b(d_r)}{p^{r+1}} \quad (57)$$

Consider the permutations $\pi_b = (0, p-1, p-2, \dots, 1)$ for the prime bases p of the Halton sequence, where each digit a_i is replaced by $p - a_i$, except when it is zero. This is called *Reverse Permutations*. A reverse Halton point is a standard Halton point shifted by an amount that depends on the digits of k . The algorithm is really similar to the Halton one, changing only one line between the two, and is:

```

p' = p, k' = k, Phi = 0
while k' > 0 do
  if (k' mod p) = 0
    a = 0
  else
    a = p - (k' mod p)
  a = k' mod p
  Phi = Phi + a / p'
  k' = int(k' / p)
  p' = p' p

```

5.4.1 Positions

After generating uniform 6D points using the scrambled Halton sequence, we have to uniformly fill the cylindrical space inside the device. The last three numbers in the 6D vector $[0, 1)^6$ are mapped using this transform:

- $r = \text{square root of the 4th dimensional } [0, 1)^6 \text{ number} * device_{radius}$
- $\theta = \text{5th dimensional } [0, 1)^6 \text{ number} * 2\pi$
- $z = \text{6th dimensional } [0, 1)^6 \text{ number} * device_{length}$

This way, the positions fills uniformly the space inside the device, as can be seen in the Fig. 14 which is a particle positions 2D plot after Scrambled Halton generation and the remapping on cylindrical coordinates:

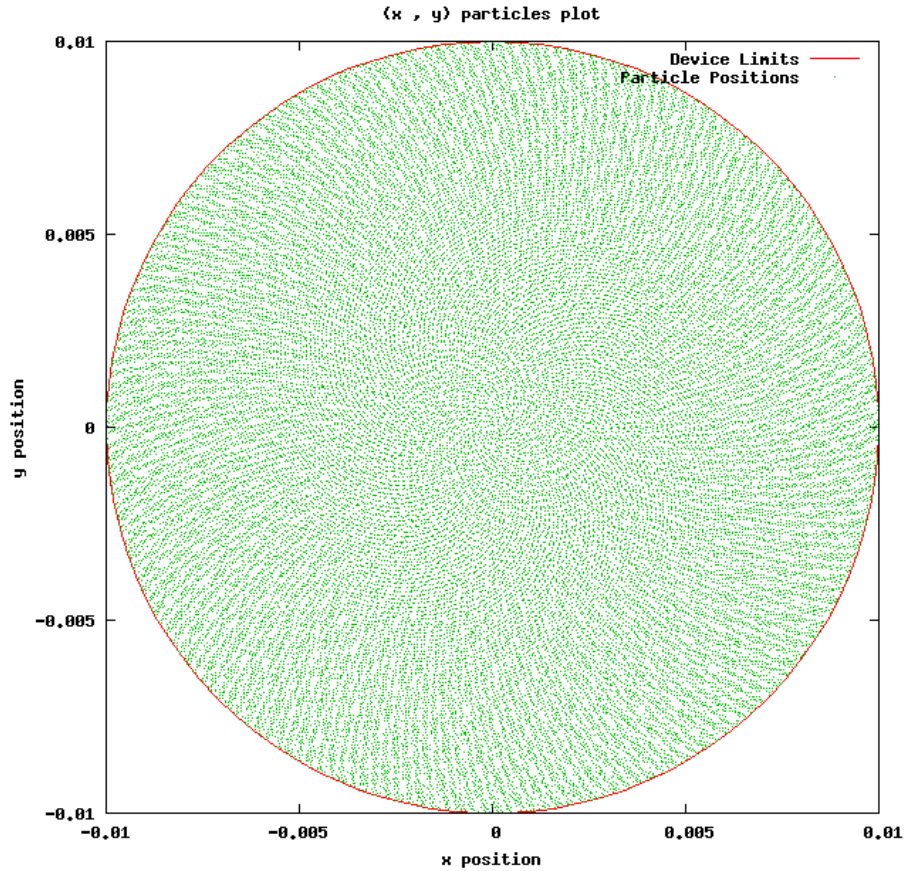


Figure 14: 2D plot of initial particle positions inside the plasma source

5.4.2 Velocities

The first three of the six dimensions generated by the Halton sequence are used to obtain a Maxwellian particle speed distribution. The uniform numbers from the first three dimensions of the scrambled Halton points are processed by a Gaussian number generator. The first three dimensions are chosen because they exhibit less correlation between 2D projections compared to the higher three dimensions which use higher prime numbers. The Gaussian number generator accepts in input a number from an uniform distribution in the interval $[0, 1)$ and returns a number with a Gaussian distribution, with mean 0 and standard deviation 1. So, applying the Gaussian number generator to each of the three dimensions returns points with Gaussian distribution along each of the three dimensions, or a Maxwell-Boltzmann speed distribution in 3D. The output of the Gaussian number generator is in the

interval $(0, \infty)$ with standard deviation 1, so it must be multiplied by the thermal velocity value expressed in [m/s] in order to obtain a non-normalized velocity for the particle that can be actually be used by the integrator. The thermal velocity used is:

$$v_{th} = \sqrt{\frac{k_B T}{m}} \quad (58)$$

Where k_B is the Boltzmann constant, T is the temperature measured in K and m is the mass of the species. The Particle Code implements the Newton method in order to solve the nonlinear equation which returns the numbers with Gaussian distribution. The expression for the Cumulative Distribution Function (CDF) for a variable having Normal distribution is:

$$y = \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) \right] \quad (59)$$

Where erf is the Error Function, defined as:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (60)$$

So the nonlinear equation to solve for x is:

$$\frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) \right] - y = 0 \quad (61)$$

Where y is the input number with uniform distribution, and x is the output with Gaussian distribution. The Newton algorithm is:

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)} \quad (62)$$

$f(x)$ is the equation we are searching for solutions:

$$f(x) = \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) \right] - y \quad (63)$$

In other terms, this equation represents a translation along the vertical axis of the CDF. We need to find the zero of this function in order to obtain a point with a Gaussian distribution starting from a point with an Uniform distribution.

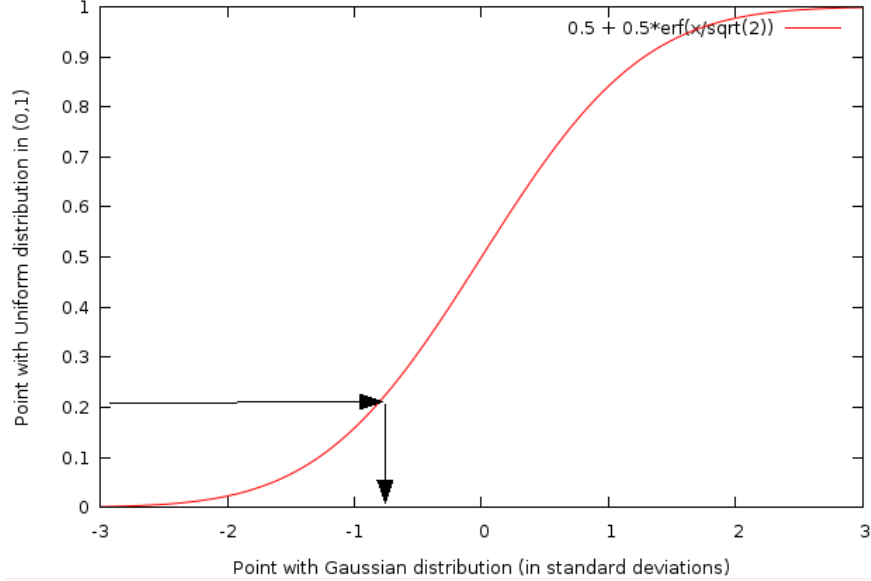


Figure 15: Change from Uniformly distributed points to Gaussian distributed ones

The derivative of the CDF is the Normal Probability Density Function (PDF), and it can be seen from these equations:

$$\begin{aligned} \frac{d}{dx} \left[\frac{1}{2} + \frac{1}{2} \operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) - y \right] &= \frac{d}{dx} \left[\frac{1}{2} \operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) \right] = \\ \frac{1}{2} \frac{d}{dx} \left[\operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) \right] &= \frac{1}{2} \frac{2}{\sqrt{\pi}} e^{-\left(\frac{x}{\sqrt{2}}\right)^2} \frac{1}{\sqrt{2}} = \\ &= \frac{1}{\sqrt{2\pi}} e^{-\left(\frac{x}{\sqrt{2}}\right)^2} \end{aligned}$$

So, the Newton scheme for the solution of Eq. (61) is:

$$x_{n+1} = x_n - \frac{\frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) \right] - y}{\frac{1}{\sqrt{2\pi}} e^{-\left(\frac{x}{\sqrt{2}}\right)^2}} \quad (64)$$

Or, in terms of the deviation:

$$s_n = x_{n+1} - x_n = - \frac{\frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) \right] - y}{\frac{1}{\sqrt{2\pi}} e^{-\left(\frac{x}{\sqrt{2}}\right)^2}} \quad (65)$$

$$x_{n+1} = x_n + s_n \quad (66)$$

Starting from an initial guess of 1.0, and an y value coming from the first three dimensions of the Halton sequence generator, Eq. (65) is computed and the deviation from each iteration is added to the previous iteration x value following Eq. (66), until convergence is reached. The stop test is done on the deviation, and iterations are stopped when:

$$s_n \leq \sqrt{\epsilon} \quad (67)$$

Where ϵ is the desired accuracy. Finally, to obtain the velocity to be inserted in the particle state vector, the resulting number is multiplied by the thermal velocity in $\left[\frac{m}{s}\right]$.

5.5 Parameters definition

In this code section several parameters needed to run the simulation are defined:

- The simulation timestep, as $\frac{1}{10}$ the electron cyclotron period:

$$timestep = \frac{1}{10} \frac{m_e 2\pi}{-q_e B_{z,static}}$$

If the electron plasma frequency is higher, it will be used in place of the electron cyclotron frequency in the timestep definition using the following equation:

$$timestep = \frac{1}{10} 2\pi \sqrt{\frac{m_e \epsilon_0}{n_e q_e^2}}$$

- The number of timesteps
- The number of timesteps needed to reach steady state

5.6 Time loops

A *for* loop is used to execute the time loops simulation. The number of loops equals the total number of timesteps, that is the number needed to reach steady state and the number of timesteps during which the system runs at steady state. The integration time is a multiple of the RF wave period, so the time-to-frequency part of the current density 3D FFT in the postprocessing section will have an harmonic exactly at the Wavecode frequency. Each time loop is composed of several sections here explained.

5.6.1 Timestep setup phase

At the beginning of each timestep, current density for all grid points of the current density grid is set to zero, and so are the species vectors containing the charge of that species on grid points of the Gauss' Law grid for the computation of the radial self-consistent electric field. The current time, given the `tscount` variable the number of the time loop being executed, is computed by:

$$t = tscount \cdot timestep \quad (68)$$

Current density is now weighted to the 3D grid and stored in a temporary file for this timestep. It will be read later by the code section that will Fourier transform it.

5.6.2 Radial self-consistent electrostatic field solver

The code can solve the self-consistent electrostatic field in each timestep. It includes custom functions for charge deposition on a spatial grid and field solve and must be executed in each time loop. To consider the electrostatic field arising from nonuniform charge densities, it is necessary that the Particle Code solves one of the Maxwell equations: the *Gauss' Law*. We can solve it in *Electrostatic Approximation*: we suppose that internal magnetic fields do not change in time, or if they do, they change very slowly with time so $\frac{dB}{dt} \approx 0$ and the self-consistent electrostatic field can be computed only using Gauss' Law, considering that in electrostatic approximation Faraday's law says that the self-consistent electrostatic field is irrotational. The approach used to solve for Gauss' Law is Particle in Cell (*PIC*). Particles are like charged rigid clouds that can overlap with a shape factor that determines how charge is deposited to the radial grid. The Gauss' Law equation relates charge or charge density to the self-consistent electrostatic field. The equation, in S.I. units is:

$$\Phi_{E,S} = \frac{Q}{\epsilon_0} \quad (69)$$

Where Q is the total charge contained in the control volume, and $\Phi_{E,S}$ is called the *electric flux* through the surface S . The electric flux can be written as:

$$\Phi_{E,S} = \oint_S \vec{E} \cdot \vec{n} dA \quad (70)$$

Where \vec{E} is the electric field at the surface boundaries, and \vec{n} is the normal vector at the surface. Applying the divergence theorem, one can write Eq. (69) in differential form:

$$\nabla \times \vec{E} = \frac{\rho}{\epsilon_0} \quad (71)$$

However, for our cylindrical coordinates 1d model (radius only) we can exploit system symmetry and use Eq. (69). In 1d, radius only, the grid quantities are spaced and indexed as shown in Fig. 16



Figure 16: Grid definition for Radial Field solver

In a radial model the particles are like cylindrical shells uniform along θ and z . Let the charges q_i of the particles be assigned to the grid points j using a weighting which guarantees charge conservation, that is:

$$\sum_{i=0}^N q_i = q_{total} = \sum_{j=0}^{N_r-1} Q_j \quad (72)$$

Applied at the cylindrical surface $j = \frac{1}{2}$, the integral form of Gauss' law produces the first radial electric field from charges as:

$$\frac{Q_0}{\epsilon_0} = 2\pi r_{\frac{1}{2}} E_{\frac{1}{2}} \quad (73)$$

where $r_{\frac{1}{2}}$ is a radius in the interval, like $\frac{1}{2}(r_{j+1} - r_j)$, and between the $j = \frac{1}{2}, \frac{3}{2}$ surfaces:

$$\frac{Q_1}{\epsilon_0} = 2\pi r_{\frac{3}{2}} E_{\frac{3}{2}} - 2\pi r_{\frac{1}{2}} E_{\frac{1}{2}} \quad (74)$$

and so on for the other charge grid points. As in all purely radial models, E_r at a certain radius a depends only on the net charge enclosed in $r \leq a$ and is unaffected by the charges at $r > a$. In the absence of a line charge at the origin (along the axis), $E_0 = 0$. So, from Eq. (73) we can obtain the first value of the radial self-consistent electric field:

$$E_{\frac{1}{2}} = \frac{Q_0}{2\pi r_{\frac{1}{2}} \epsilon_0} \quad (75)$$

And, from Eq. (74) we can obtain all the other radial electric fields:

$$E_{\frac{3}{2}} = \frac{\frac{Q_1}{\epsilon_0} + 2\pi r_{\frac{1}{2}} E_{\frac{1}{2}}}{2\pi r_{\frac{3}{2}}} \quad (76)$$

and so on for the other radial electric fields until we find the last value, right before the plasma source radius. In all these equations, the charges Q_j are actually *charge over length*, where the length is the device dimension along z .

Particle weights The method we choose to transfer particle charges from the continuous particle positions to the discrete grid points for the computation of the radial electric field, is *Area Weighting*, or linear weighting in r^2 . This keeps in account that we are working in a cylindrical geometry.

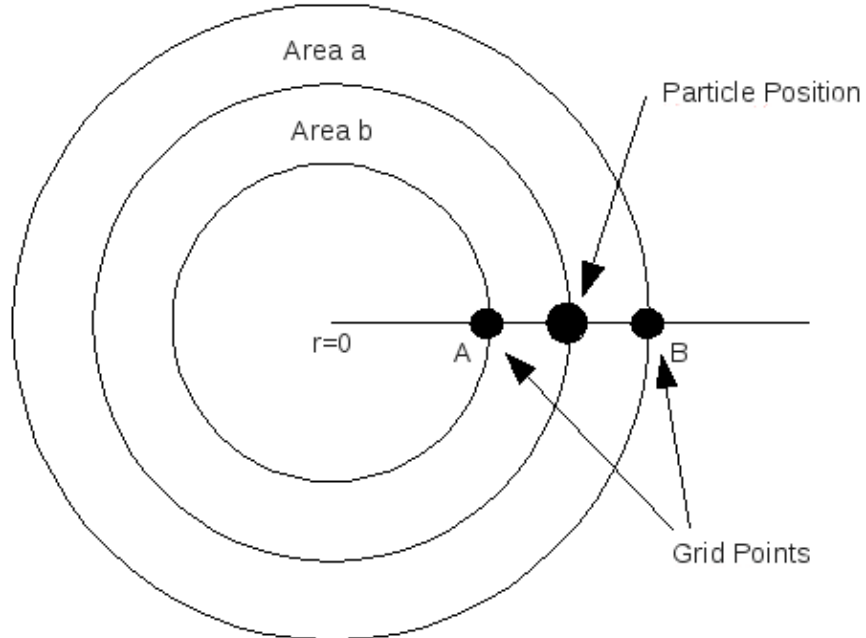


Figure 17: Area Weighting charge deposition

The fraction of the charge q_j assigned to point A (at grid point radius r_j) is $\frac{area_a}{area_a + area_b}$, leading to:

$$Q_A = p_w q_i \frac{r_{j+1}^2 - r_i^2}{r_{j+1}^2 - r_j^2} \quad (77)$$

$$Q_B = p_w q_i \frac{r_i^2 - r_j^2}{r_{j+1}^2 - r_j^2} \quad (78)$$

One should note that the charge of the particle is multiplied by its weight p_w in the charge deposition stage, to account for the computer particle to be a collection of p_w real particles that have the same position.

5.6.3 Fields interpolation

A custom function receives the radius r_i of the particle being processed, and then returns the interpolated electromagnetic fields from discrete grid points to the current particle radial position. Two interpolations are performed:

- Self-consistent electric field interpolation.

The weighting of the E_r field to the particles is done using area weighting. However, the fields we have obtained are not at the grid points, but halfway between them. Two methods were implemented to obtain the fields at the grid points:

- Unweighted average

$$E_{r,j} = \frac{E_{r,j-\frac{1}{2}} + E_{r,j+\frac{1}{2}}}{2} \quad (79)$$

- Flux weighted average

$$E_{r,j} = \frac{r_{j-\frac{1}{2}} E_{r,j-\frac{1}{2}} + r_{j+\frac{1}{2}} E_{r,j+\frac{1}{2}}}{2r_j} \quad (80)$$

Then, area weighting is used again to weight these E_r to the particles, to obtain the correct component $E_{r,sc}$ in the Lorentz force, for the particle having the continuous position r_i :

$$E_{r,sc}(r_i) = E_{r,j} \frac{r_{j+1}^2 - r_i^2}{r_{j+1}^2 - r_j^2} + E_{r,j+1} \frac{r_i^2 - r_j^2}{r_{j+1}^2 - r_j^2} \quad (81)$$

This is the value of the self-consistent electric field interpolated from the discrete grid points j and $j + 1$ to the continuous particle position r_i .

- RF fields interpolation.

The external fields from the Wavecode are complex numbers varying according to this equation:

$$E_r = |E_r| e^{j(-\omega t + m\theta + kz + \Phi)} \quad (82)$$

So, the particle position in the radial Wavecode grid and the fields at the current timestep are computed for the two radial grid nodes surrounding the particle radial position, according to the following equations:

$$E_r = |E_r| \cos(\Phi_{E_r} - \omega t + m\theta_p + kz_p)$$

$$E_\theta = |E_\theta| \cos(\Phi_{E_\theta} - \omega t + m\theta_p + kz_p)$$

$$E_z = |E_z| \cos(\Phi_{E_z} - \omega t + m\theta_p + kz_p)$$

$$B_r = |B_r| \cos(\Phi_{B_r} - \omega t + m\theta_p + kz_p)$$

$$B_\theta = |B_\theta| \cos(\Phi_{B_\theta} - \omega t + m\theta_p + kz_p)$$

$$B_z = |B_z| \cos(\Phi_{B_z} - \omega t + m\theta_p + kz_p)$$

Where the phase angles Φ are computed from the arctangent of imaginary and real part of the Wavecode output fields. z_p, θ_p, t are respectively the current particle axial position, azimuthal position and the current time. The fields from the two radial grid points surrounding the particle radial position are linearly interpolated to obtain the external fields used in the Lorentz force for the integration of equations of motion.

After the two interpolations, the static coils magnetic field $B_{z,static}$ is added to B_z . This gives the six external fields interpolated to the particle radial position, accounting for the coils axial magnetic field. The interpolated radial electric field from Gauss' Law solution is also added to the interpolated RF radial electric field to obtain the total radial electric field acting at the current particle position.

5.6.4 Integration of the equations of motion

The integrator is the Particle Code function that advance in time charged particles positions and velocities subjected to Electromagnetic forces. The time integration is performed on all active particles in the simulation, to advance in time their state vectors when they are subjected to electromagnetic forces. The trajectory of a particle is described by:

$$\frac{d\mathbf{v}}{dt} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \quad (83)$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \quad (84)$$

Eq. (83) is integrated following Buneman-Boris leap-frog formulation indicated in Fig. 18

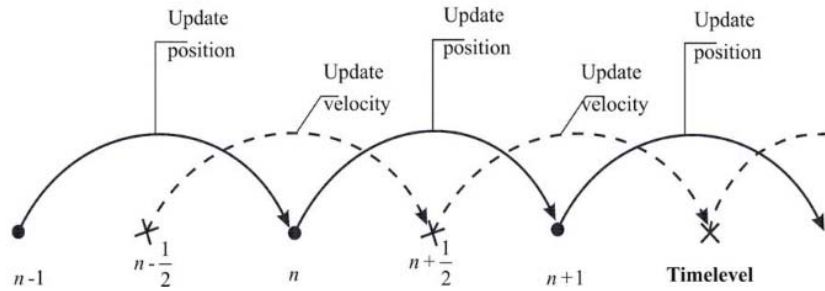


Figure 18: Leap-Frog scheme

The integrator works in Cartesian coordinate system instead of cylindrical, because of equations singularities otherwise arising near the axis of the plasma source. The leap-frog scheme is time-centered and involves:

- an acceleration due to half the electrostatic force

$$\mathbf{v}^- = \mathbf{v}(t - \frac{\Delta t}{2}) + \frac{q\mathbf{E}(t)\Delta t}{2m} \quad (85)$$

- a rotation of the velocity vector due to the interaction between the velocity vector and the magnetic field. We define a vector \mathbf{t} parallel to the magnetic field vector \mathbf{B} :

$$\mathbf{t} = \frac{q\mathbf{B} \Delta t}{m} \frac{\Delta t}{2} \quad (86)$$

We then find \mathbf{v}' as:

$$\mathbf{v}' = \mathbf{v}^- + \mathbf{v}^- \times \mathbf{t} \quad (87)$$

The kinetic energy after the rotation is unchanged if:

$$\mathbf{s} = \frac{2\mathbf{t}}{1 + t^2} \quad (88)$$

Then, the velocity after the rotation is:

$$\mathbf{v}^+ = \mathbf{v}^- + \mathbf{v}' \times \mathbf{s} \quad (89)$$

- an acceleration due to the remaining half of the electrostatic force.

$$\mathbf{v}(t + \frac{\Delta t}{2}) = \mathbf{v}^+ + \frac{q\mathbf{E}(t)\Delta t}{2m} \quad (90)$$

- the particle position is then updated using the new value of the velocity vector.

$$\mathbf{x}(t + \Delta t) = \mathbf{v}(t + \frac{\Delta t}{2})\Delta t + \mathbf{x}(t) \quad (91)$$

5.6.5 Boundaries check and plasma source

In this section, the code checks if, after the integration of the equations of motion, the resulting particle position is outside of the simulation domain.



Figure 19: Device limits (simulation domain)

If it is the case, two things can happen if the particle exited the domain radially:

- If the particle is an electron, the memory location containing its state vector will be flagged as not used. In the following timesteps, if that memory location is not used, it will not be subjected to time integration, to charge weighting and to current density weighting. The electron is "lost".
- If the exiting particle is an ion, the code uses the custom Quiet Loader function to load a new state vector for that ion, which has the uniform density distribution and Maxwellian speed distribution. It then searches for electron species memory locations flagged as "not used" and then loads there a new state vector for a new electron flagging it as "used". If all electrons memory locations are already used, it makes room for other memory locations available for storing the new electron state vector. This will then load an ion-electron couple.

The physical consequence of this is that we assume *plasma source rate to be the same as ions loss rate*, so a neutral atom is dissociated to an ion and an electron only when an ion exits the system. This results in a constant source rate, as the ion loss rate is constant at steady state. The code allows implementation of arbitrary plasma source rates and dynamic memory allocation allows variation of the number of particles during the simulation; currently a standalone code named Quicksource is in final development phase and will give the real plasma source rate as an input for the Particle Code.

If the particle presents an axial position lower than zero or greater than the device length, it means it exited the computational domain axially. It will then be reinjected from the opposite side with the same velocity. The two possible cases are:

- Particle exited from $z = 0$ boundary:

$$z_{p,new} = z_p - L$$

- Particle exited from $z = L$ boundary:

$$z_{p,new} = z_p + L$$

Where $z_{p,new}$ is the new particle position after reinjection. Thus, the code is axially periodic.

After this code section, the successive time loops are performed until the end of simulation.

5.7 Current density FFT

After the time loops, to obtain the Current Density as a function of radius, frequency and axial and azimuthal modes, a Fourier Transform is performed on the current density grid nodes. The code implements a 1D time-to-frequency FFT followed by a 2D partial spatial FFT (azimuthal and axial coordinates to azimuthal and axial modes). The 1D transform is a backward FFT (that is, an FFT in which the sign of the exponential is positive) and the 2D transform is a forward FFT, whose input is the result of the 1D transform. The backward FFT is needed because of the definition of wave quantities in the Wavecode. We use the open-source library FFTW, because of the following reasons:

- It's generally the fastest FFT implementation available
- It's portable, so it can work with every operating system
- It's based on plans that choose the best FFT algorithm for the given problem, depending on the transform type and size
- It's not limited to power-of-two transform sizes

5.7.1 FFT Normalization

The library used for Fast Fourier Transforms, FFTW, computes an unnormalized transform: computing a forward followed by a backward transform (or vice versa) will result in the original data multiplied by the size of the transform (the product of the dimensions). The multi-dimensional transforms of FFTW compute simply the separable product of the given 1d transform along each dimension of the array. Since each of these transforms is unnormalized, computing the forward followed by the backward/inverse multi-dimensional transform will result in the original array scaled by the product of the normalization factors for each dimension (e.g. the product of the dimension sizes, for a multi-dimensional DFT). For our transform strategy we had to find the normalization factor that has to be multiplied by the result of the 3D FFT. To do this, a sample program has been developed. This uses the same function wrote for the particle code, and transforms a sinusoidal current density wave, featuring a single frequency, dc axial wavenumber and dc azimuthal wavenumber. The result of the FFT should be a single harmonic having an amplitude equal to the input amplitude of the current density. For this to be true, we found that the normalization constant is:

$$n_c = \frac{2}{n_t n_z n_k} \quad (92)$$

Where n_t , n_z , n_k are respectively the time, axial and azimuthal grid sizes.

5.8 Phase space saving

At the end of the code, a file is saved containing the phase space of all particles, to be used in the next coupled codes iteration. This allows a better convergence rate of the two codes. The data saved to the temporary file, which will be used to read initial conditions for the next Particle Code run, is:

- The number of particles at the end of the simulation for every species
- The physical to computational ratio for every particle
- Position and velocity vectors of every particle in the simulation

This file is also useful for diagnostics at the end of the simulation, being a snapshot of the phase space for all the particles.

5.9 Plasma power absorption

The power absorbed by the plasma is computed by the Particle Code at the end of the run with the volume integral:

$$\int_V \mathbf{E} \cdot \mathbf{J} dV \quad (93)$$

Where \mathbf{E} and \mathbf{J} are complex numbers, and V is the volume of the Particle Code domain. \mathbf{E} is the RF electric field, while \mathbf{J} is the current density computed by the Particle Code. The Particle Code implements this integral computing cylindrical shells volumes centered on the Wavecode grid points, where both the electric field and the current density are known, and multiplies that volume by the scalar product between the complex-valued vectors \mathbf{E} and \mathbf{J} . After this is done for all the Wavecode grid points, the contributions to absorbed power computed at each radius are added to obtain the total power.

6 Particle Code Validation & Verify

6.1 Single Particle Validations

The integration scheme adopted is a second-order, time-reversible Buneman-Boris Leapfrog. This showed to be the best option in this code, being a tradeoff between accuracy and execution speed. Its time-reversibility property is desirable because it allows to maintain cyclotron orbits even after long integration times. In order to verify the integrator, several test cases have been executed.

6.1.1 Cyclotron motion

The first analysis involves the motion of an electron particle in the perpendicular plane, subjected to an uniform, static axial magnetic field (typically generated by the confinement coils). The motion should be a pure cyclotron motion: the electron follows a circular orbit, with a radius depending on the velocity of the electron in the perpendicular plane and the magnetic field magnitude, and the frequency depending only on the magnitude of the axial magnetic field. Two parameters were used in the validation:

- Larmor radius: it is the radius of the circular motion, theoretically given by Eq. (94)

$$r_g = \frac{mv_{\perp}}{|q|B} \quad (94)$$

- Cyclotron frequency: it is the frequency of the circular motion, theoretically given by Eq. (95)

$$\nu = \frac{|q|B}{2\pi m} \quad (95)$$

The particle has an initial state vector generated by the Particle Code functions using the built-in Quasi-Random number generator, and its state vector has been logged to a file for the analysis.

The parameters used are:

$$q = -1.60217646 * 10^{-19} C$$

$$m = 9.10938188 * 10^{-31} kg$$

$$B = 0.01 T$$

The test particle generated speed in the perpendicular plane is:

$$v_{\perp} = 312816.3m/s$$

From these parameters, one could obtain the theoretical values of the analysis parameters from Eq. (94) and Eq. (95):

$$r_g = 0.00017786m$$

$$\nu = 2.79924924 * 10^8 Hz$$

In order to compute the Larmor radius, the time evolution of the radial position of the particle has been analyzed, showing a minimum of:

$$r_{min} = 6.619867 * 10^{-3}m$$

at time

$$t = 1.393231 * 10^{-9}s$$

and a maximum of:

$$r_{max} = 6.975727 * 10^{-3}m$$

at time

$$t = 3.179424 * 10^{-9}s$$

So, the Larmor radius is:

$$r_g = \frac{6.975727 * 10^{-3} - 6.619867 * 10^{-3}}{2} = 0.00017793m$$

and the frequency is:

$$\nu = \frac{1}{2(3.179424 * 10^{-9} - 1.393231 * 10^{-9})} = 279924957 Hz$$

This result has been obtained with a timestep equal to 1/100 the theoretical cyclotron period.

6.1.2 Constant axial electric field

An electron has been placed in the domain, having zero initial axial position and zero initial velocity. An axial electric field has been activated, with a value of $-100 \frac{V}{m}$. The electron is subjected to an uniform acceleration along the z axis. The acceleration is given by:

$$a = \frac{qE_z}{m} = 1.7588 * 10^{13} \frac{m}{s^2}$$

The integration time in the Particle Code has been set as:

$$t = 3.096068 * 10^{-8} s$$

The theoretical velocity and position at the end of the integration time are given by:

$$z = \frac{1}{2}at^2 = 0.008429609m$$

$$v_z = at = 5.4454 * 10^5 \frac{m}{s}$$

The Particle Code values after time integration are:

$$z = 0.008429707m$$

$$v_z = 5.4244 * 10^5 \frac{m}{s}$$

This result has been obtained with a timestep equal to 1/10 the theoretical cyclotron period.

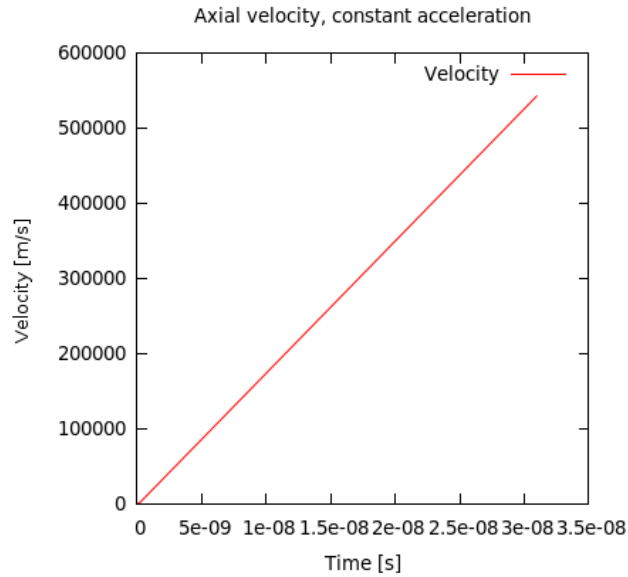


Figure 20: Particle Code output: velocity of an electron subjected to constant acceleration

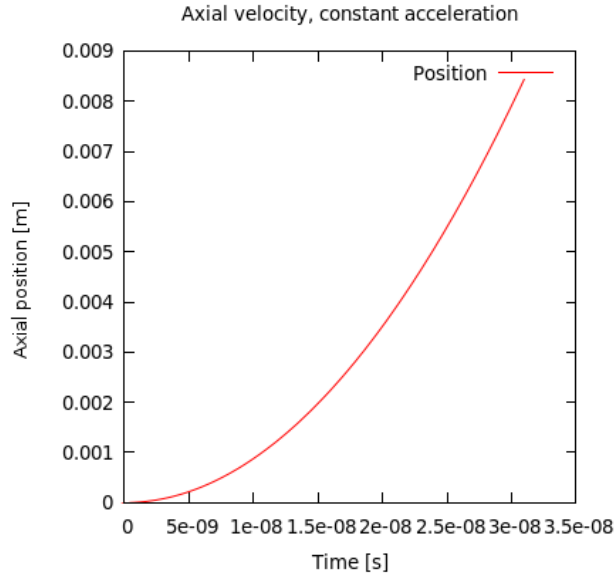


Figure 21: Particle Code output: position of an electron subjected to constant acceleration

6.2 Initial velocity distribution

The initial velocity distribution should be a quiet Maxwellian. To generate it, the code generates Quasi-Random numbers in $[0,1)$ along the three dimensions, and a Newton method to solve for the shifted Gaussian CDF, in order to obtain a Gaussian distribution in each direction, with zero mean and standard deviation equal to the 1D thermal velocity. To check the distribution, the first statistical moments (that is, the mean value) of the actual particles velocity distribution is computed by the code. This value should be as close as zero as possible. The following table lists the computed first statistical moments after the loading of two species represented by 500000 particles each, the Electrons having a temperature of 34800K and the Argon+ ions having a temperature of 300K (cold plasma)

Type	Dimension	Standard dev. [m/s]	Computed mean [m/s]
Electron	1	7.262515e+05	2.748970e+00
Electron	2	7.262515e+05	3.190278e+00
Electron	3	7.262515e+05	-9.794813e-01
Ion	1	2.498794e+02	1.097670e-03
Ion	2	2.498794e+02	-3.370075e-04
Ion	3	2.498794e+02	1.008112e-04

Table 2: Velocity distribution

A function interfacing with Gnuplot has been written, to visually check the velocity distribution in the plane perpendicular to the static uniform axial magnetic field. The velocity distribution plot is in Fig. 22

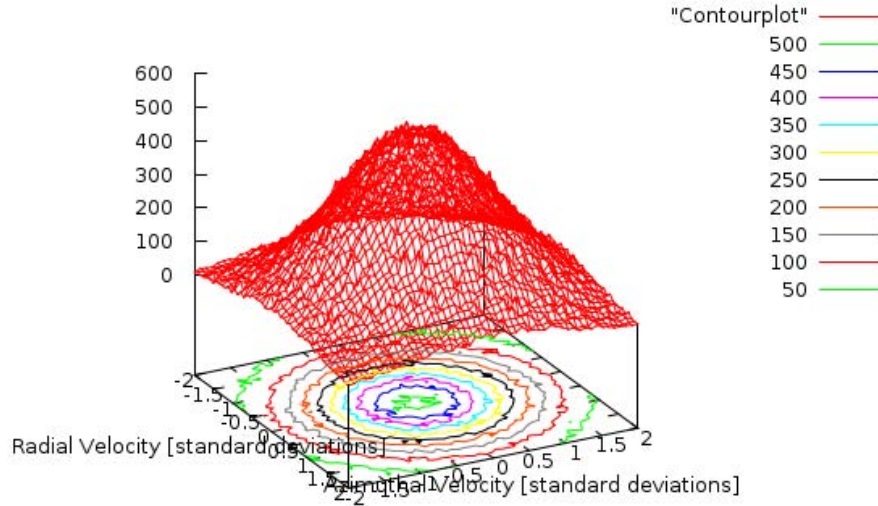


Figure 22: 2D velocity distribution

The phase spaces in Fig. 23, Fig. 24, Fig. 25 show a gaussian with zero mean for the velocities, uniformly distributed in space.

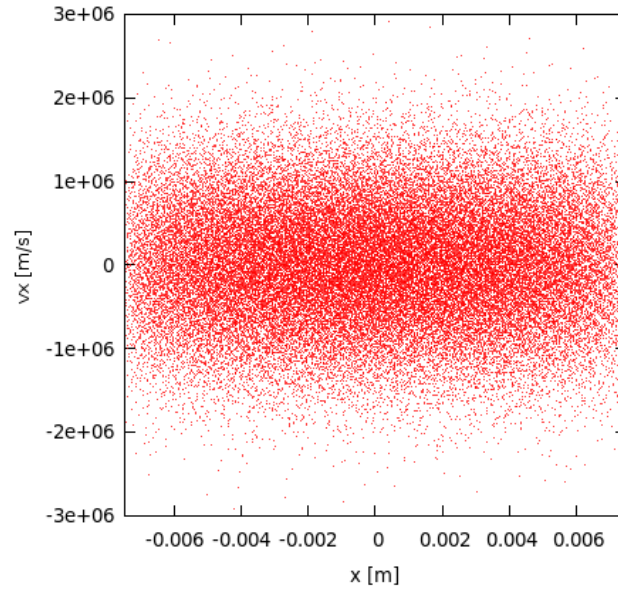


Figure 23: x-vx initial phase space

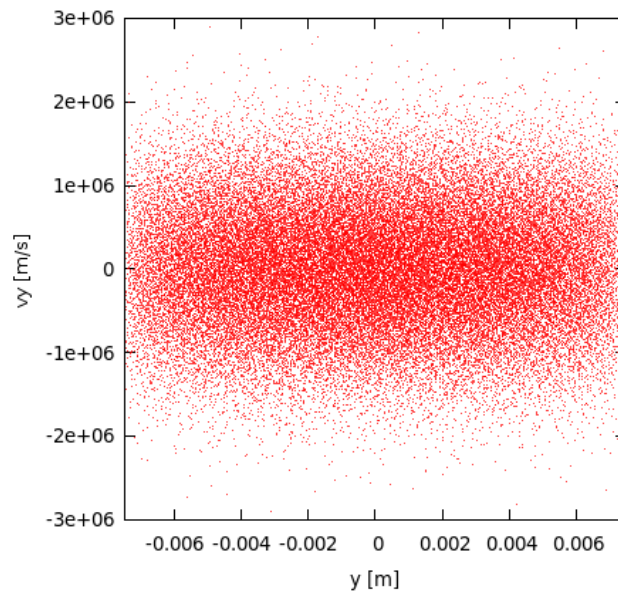


Figure 24: y-vy initial phase space

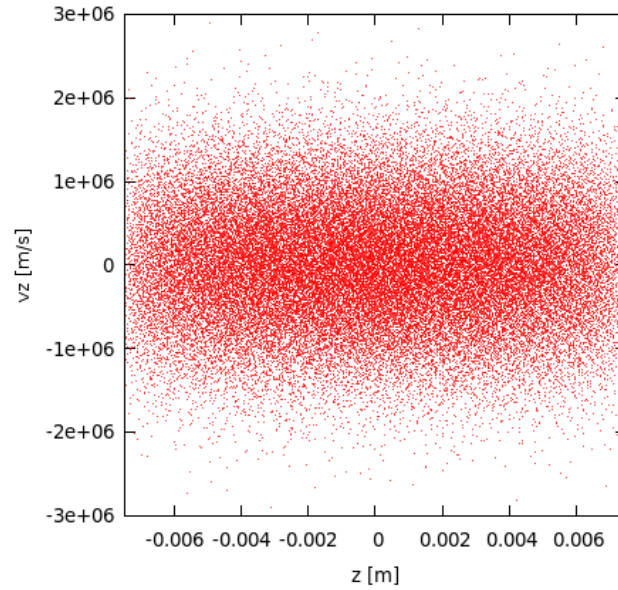


Figure 25: z-vz initial phase space

In Fig. 26 is shown the electrons speed distribution, which matches theoretical results, the most probable speed being $\sqrt{2}$ times higher than the 1D standard deviation.

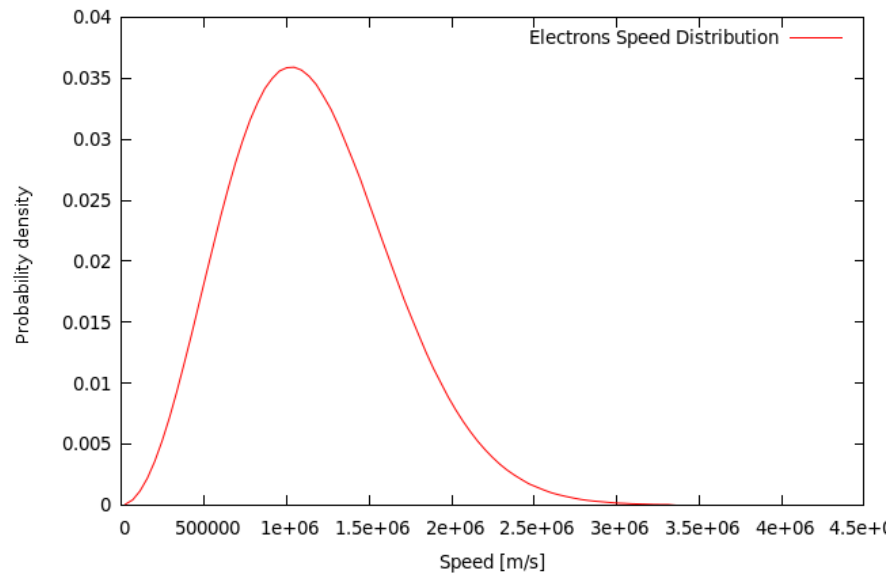


Figure 26: Initial Maxwell-Boltzmann speed distribution

6.3 Current density validation

To validate the current density algorithm, we choose an initial condition with particles having uniform spatial distribution and zero initial velocity, so that the current density could be obtained analytically and compared with the Particle Code results. The test case involved the application of an axial, radially uniform electric field, varying according to the the following equation:

$$E_z = |E_z| \cos(-\omega t + m\theta + kz + \Phi) \quad (96)$$

In which Φ , the phase angle, is imposed as zero, that is the field is a pure cosine wave. ω is kept constant, at a value of $2 * 10^8 \frac{rad}{s}$. The current density without the thermal motion can be computed analytically, having initial conditions of zero velocity for each particle. If $k = m = \Phi = 0$, the electric field has the form:

$$E_z = |E_z| \cos(-\omega t) \quad (97)$$

$|E_z|$ in the test case has a value of 100 V/m. The plasma density has been assumed to be $10^{14} m^{-3}$ and it is an input for the Particle Code. The current density grid has a size of 16x16x32, respectively for the radial, azimuthal and axial direction. The integration time is one wave period, thus the 1D part of the FFT, time-to-frequency, considers the first harmonic to be transformed with the 2D spatial FFT. Integrating analytically the equations of motion for electrons and ions under these hypotesis, the resulting magnitude of the analytical current density is $J_z = 1.405 A/m^2$, varying as a pure sine wave, constant along the radius. The result after the current density and FFT computation should be then a single harmonic having a constant imaginary value of 1.405. We present the plots showing a comparison between the imaginary part of the computed current density as a function of radius and the analytical value.

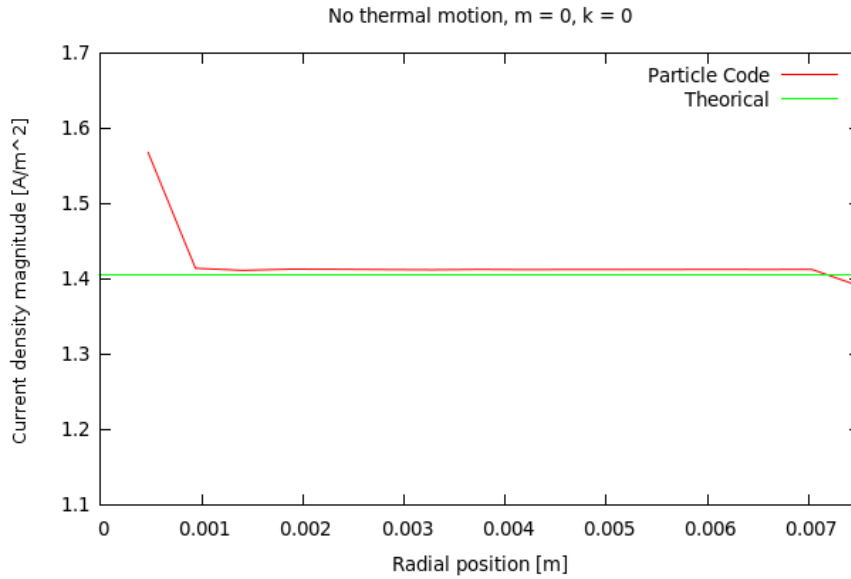


Figure 27: no thermal motion, $m = 0$, $k = 0$

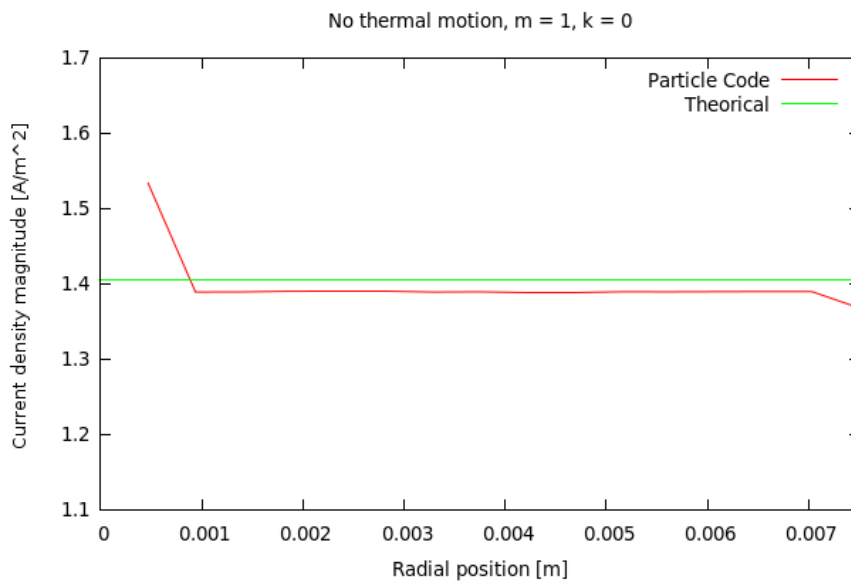


Figure 28: no thermal motion, $m = 1$, $k = 0$

For the $m = 1$ case is interesting to see also the 1D part of the FFT, that is only the part from time to frequency for all the grid points of a cylindrical shell in the current density grid. It can be seen that the partially transformed current density has different phase angles depending on the azimuthal position of the node on the

shell, showing a 2π total phase angle from the first to the last node, as it should be in the $m = 1$ case.

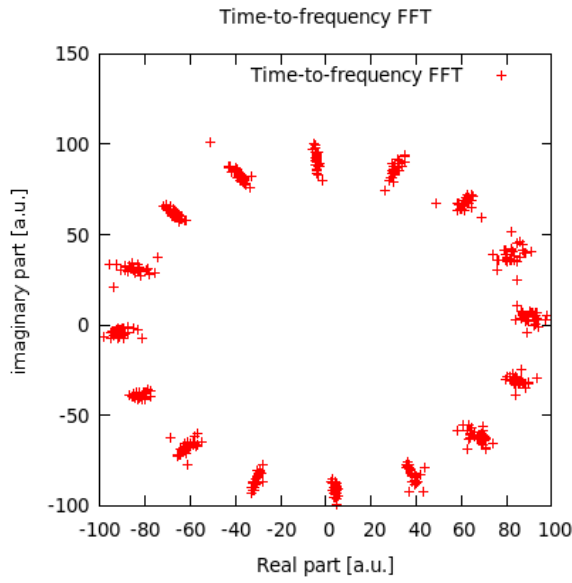


Figure 29: no thermal motion, $m = 1$, $k = 0$

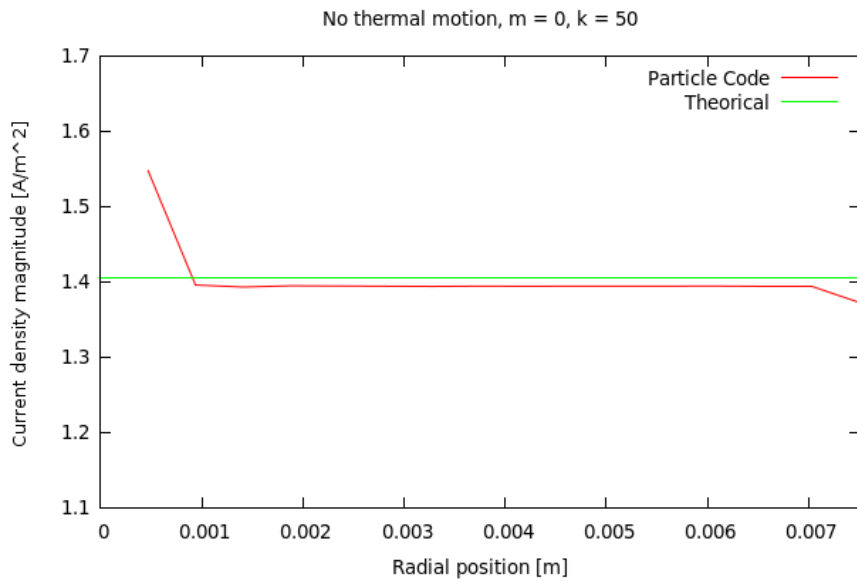


Figure 30: no thermal motion, $m = 0$, $k = 50$

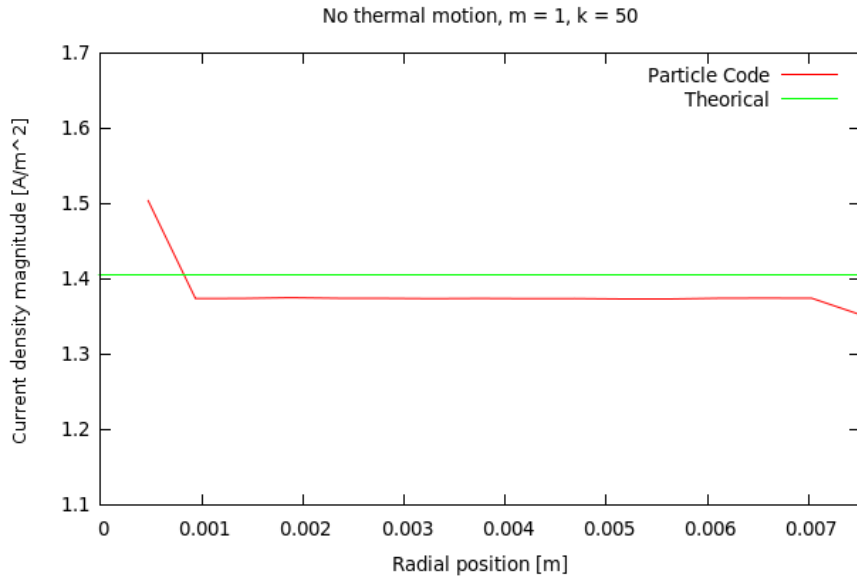


Figure 31: no thermal motion, $m = 1$, $k = 50$

7 Coupled Codes Results

The developed coupled codes (Wavecode and Particle Code) were used to investigate the plasma power absorption of several type of waves, with some of the physical parameters given by preliminary experimental measurements in the Helicon Micro Plasma experiment currently in development at University of Padua. The Particle Code runs with two species: Electrons and Argon+ ions. The Electron temperature is 34800 K, while the ions temperature is supposed to be 300 K, thus simulating a cold plasma. Initial electron density, supposed equal to initial ion density, is taken as $10^{17}m^{-3}$, from Langmuir probe measurements. With these parameters, the codes ran following the test matrix in the following table and the power absorbed by the plasma has been compared between the test cases.

Parameter	Values
Azimuthal wavenumber	-1 0 +1
Axial wavenumber [m^{-1}]	-25 25 50
Frequency [MHz]	6.78 13.56 27.12
Confinement magnetic field [T]	0.01 0.05

Table 3: Test matrix

7.1 Results

The power absorbed by the plasma has been analysed varying one parameter while keeping fixed the others, to identify the best modes configuration and then to obtain an antenna design. For every parametric analysis, the power absorbed has been normalized on the value of the higher result. The radial self-consistent electric field in the Particle Code has been turned off, so it is not included in the particles motion.

- *Effect of RF frequency*
azimuthal wavenumber = -1 ;
axial wavenumber = $25m^{-1}$;
confinement magnetic field = $0.01T$
- *Effect of azimuthal wavenumber*
RF frequency = $27.12MHz$;

axial wavenumber = $50m^{-1}$;
confinement magnetic field = $0.01T$

- *Effect of axial wavenumber*
azimuthal wavenumber = $+1$;
RF frequency = $13.56MHz$;
confinement magnetic field = $0.01T$
- *Effect of confinement magnetic field*
azimuthal wavenumber = -1 ;
axial wavenumber = $25m^{-1}$;
RF frequency = $13.56MHz$;

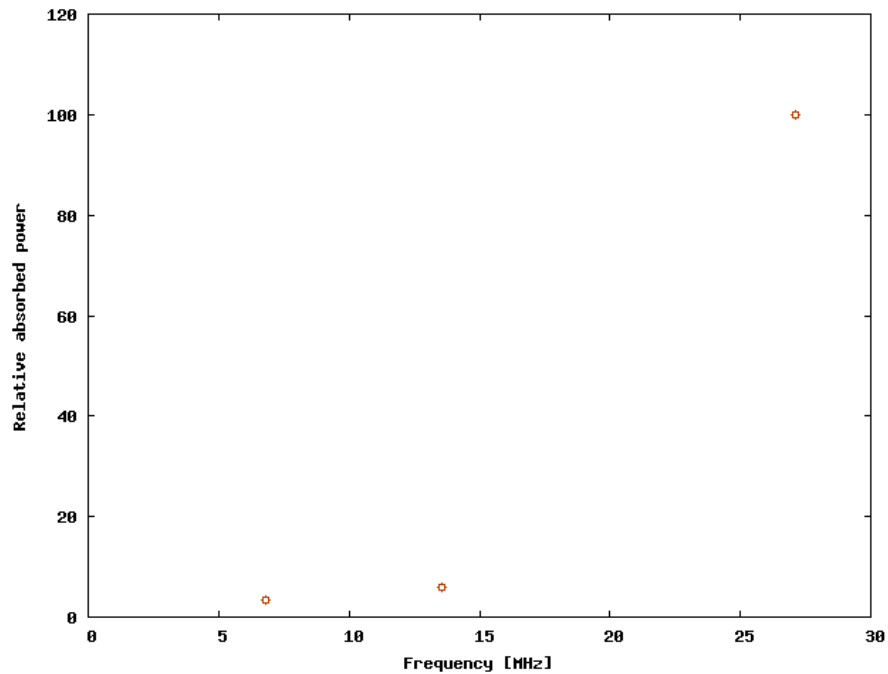


Figure 32: Effect of the RF frequency on power absorbed by plasma

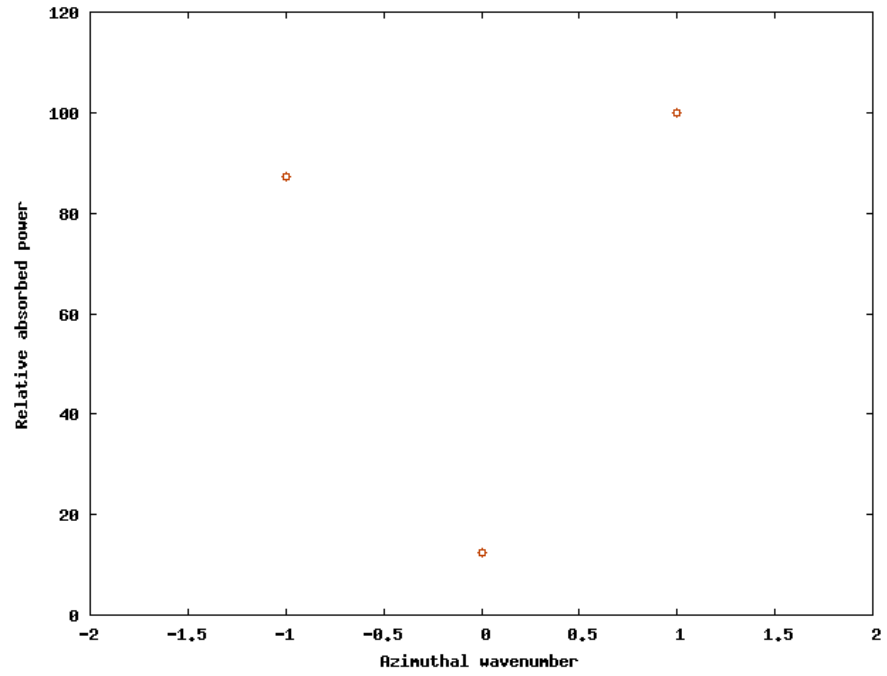


Figure 33: Effect of the azimuthal wavenumber on power absorbed by plasma

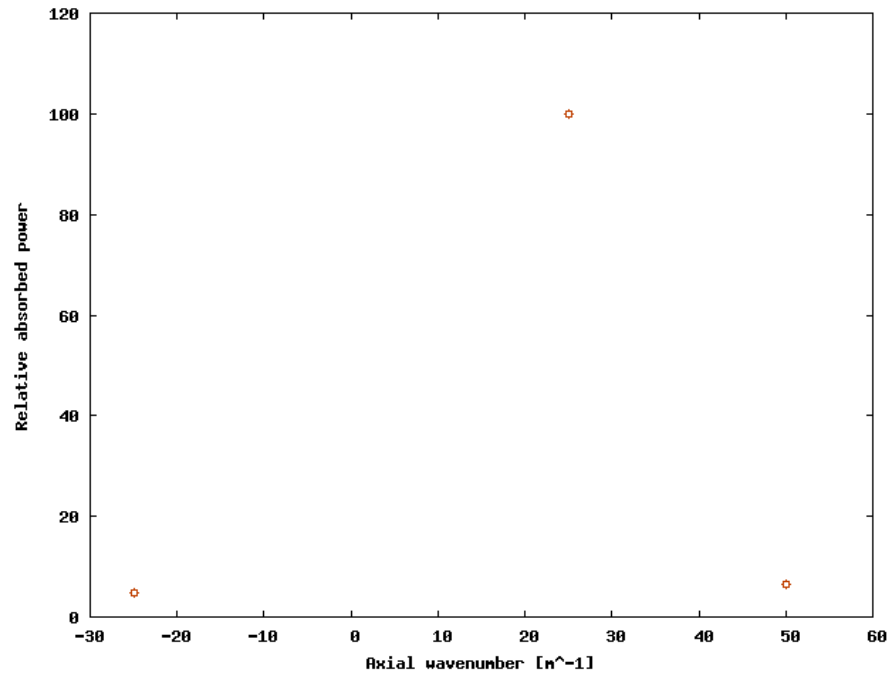


Figure 34: Effect of the axial wavenumber on power absorbed by plasma

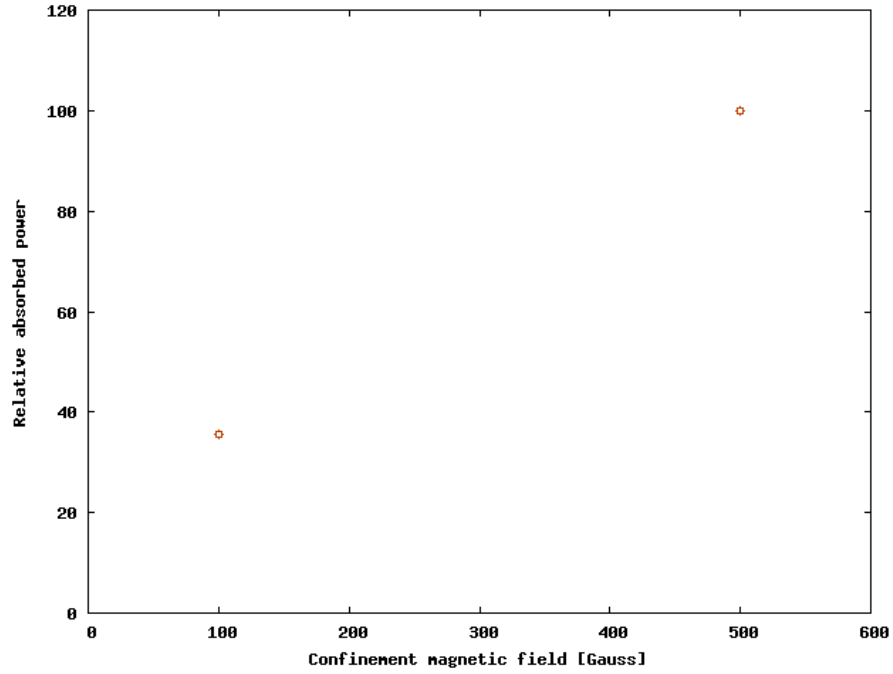


Figure 35: Effect of the confinement magnetic field on power absorbed by plasma

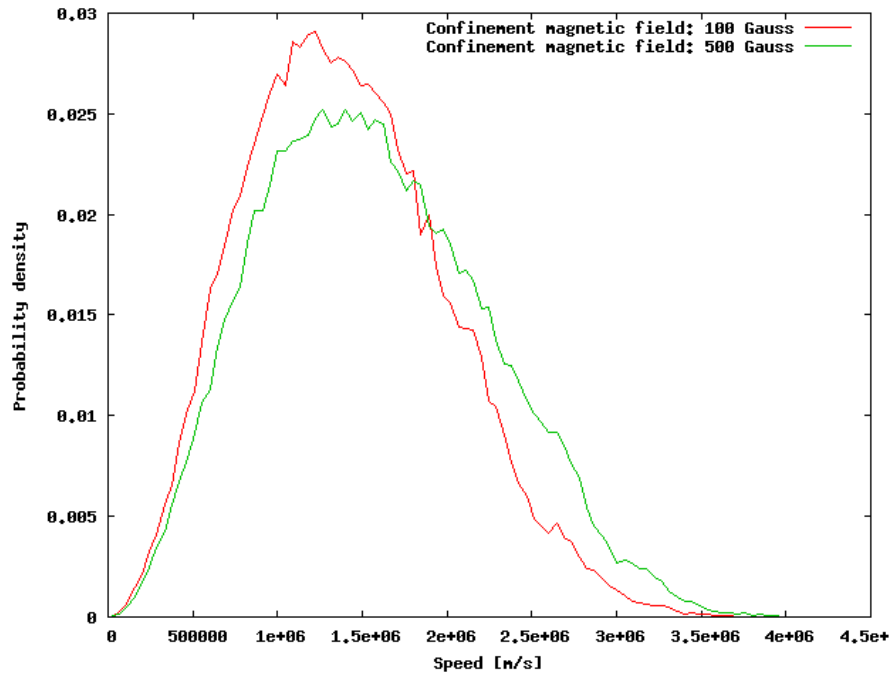


Figure 36: Speed distribution at the end of the simulation for the varying magnetic field case

In the paper "An evaluation of different antenna design for helicon wave excitation in a cylindrical plasma source" by I. V. Kamenski and G. G. Borg, a magnetohydrodynamic numerical model is employed to analyze the antenna radiated power coupled to the plasma in a cylindrical helicon wave driven plasma source; in particular, the frequency and mode numbers dependencies are investigated. Therefore, we have confirmed their numerical results and the well known experimental fact that the $m = +1$ mode is efficiently excited and that the power coupled to the plasma grows as the RF frequency is increased. Unlike them, we have also taken into account the effects of the confinement magnetic field, as shown in Fig. 35.

7.2 Results with self-consistent electrostatic field and collisions

A new test matrix has been completed with the codes, and the radial self-consistent electric field in the Particle Code has been turned on, so it is included in the particles motion. The effect of collisions has been considered too for electrons motion. We implemented in the Particle Code a collision system which rotates the particle velocity vector if a collision occurs for that electron. We assumed a 30 MHz collision frequency. For every parametric analysis, the power absorbed has been again normalized on the value of the higher result.

- *Effect of azimuthal wavenumber*
 RF frequency = $40.68MHz$;
 axial wavenumber = $25m^{-1}$;
 confinement magnetic field = $0.01T$

- *Effect of axial wavenumber*
 azimuthal wavenumber = $+1$;
 RF frequency = $13.56MHz$;
 confinement magnetic field = $0.01T$

- *Effect of confinement magnetic field*
 azimuthal wavenumber = 1 ;
 axial wavenumber = $37.5m^{-1}$;

RF frequency = $27.12MHz$;

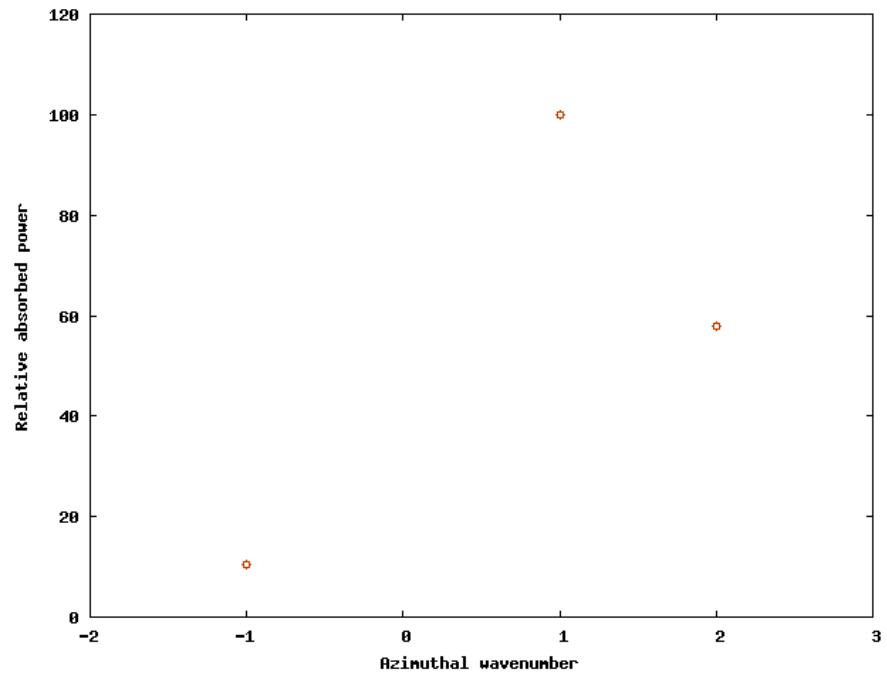


Figure 37: Effect of the azimuthal wavenumber on power absorbed by plasma

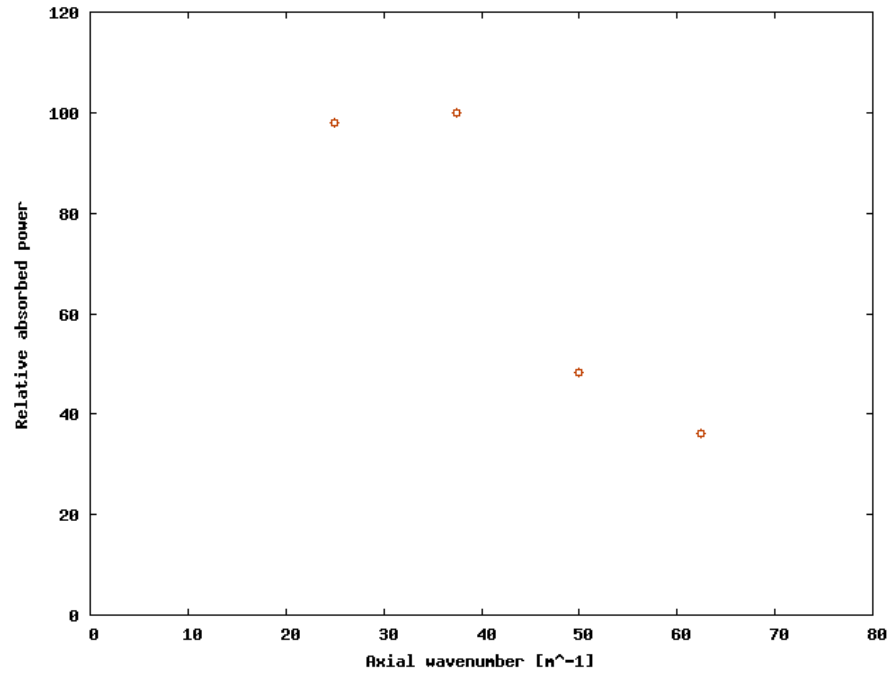


Figure 38: Effect of the axial wavenumber on power absorbed by plasma

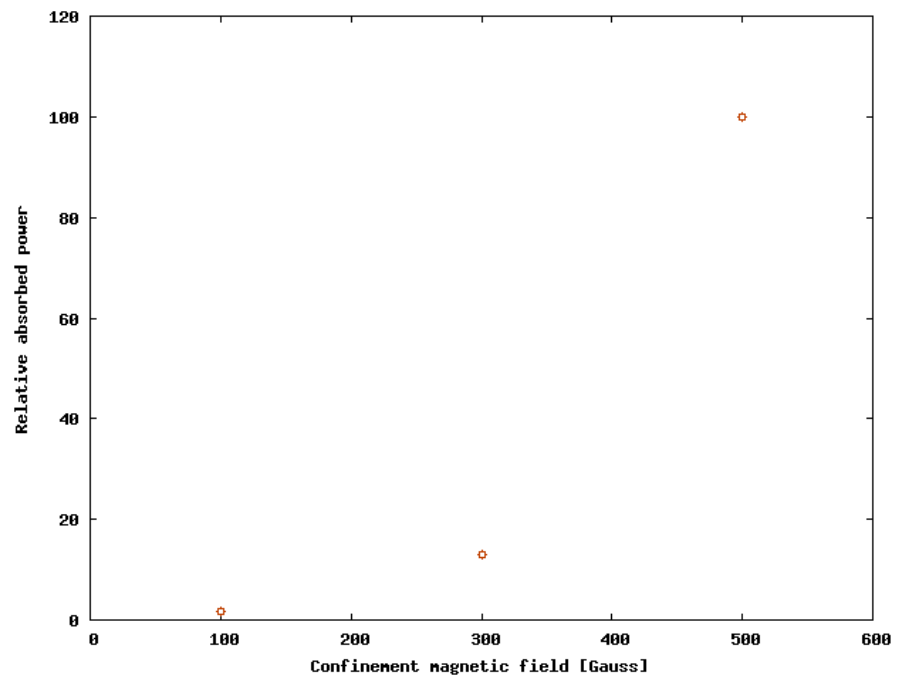


Figure 39: Effect of the confinement magnetic field on power absorbed by plasma

We can still see that the $m = +1$ mode is efficiently excited. High axial wavenumbers (which means low axial wavelengths) are not efficiently excited. We still observe an increase in power absorption with increasing static, axial magnetic field.

7.3 Procedures for Antenna Design

Once obtained the power results from the test matrix simulations, we can identify the best antenna parameters able to couple the maximum amount of power to the plasma. Due to the uniformity of the geometry in the θ and z directions, the Fourier transforms in these directions can be used, so that we will consider the $m - k_z$ spectrum of the antennas. Assuming the current density of the antennas to be solenoidal so that $\nabla \cdot \mathbf{J} = 0$, we need to determine only the azimuthal current density since the axial current density can be calculated from the $m - k_z$ transform of the continuity equation. That is:

$$J_z(m, k_z) = -\frac{m}{bk_z} J_\theta(m, k_z) \quad (98)$$

where b is the antenna inner radius. Therefore we get the current density vector $\mathbf{J}(r, m, k_z)$, since we know all its components.

Depending on the antenna modes we want to excite, we can pick different types of antenna, for which we also know the current density in the Fourier space. This current density depends not only on the antenna modes, but also on the antenna geometry, so that, once that we use the $J_\theta(r, m, k_z)$ value from Wavecode, we are able to design the antenna.

For instance, in case of Nagoya Type III, Half Helix and Half Turn, which are chosen to excite $m = +1$ mode (otherwise, in case of different m mode the current transform is different), we can consider the *fractional helix antenna* current distribution transform ¹, which is:

$$J_\theta(r, m, k_z) = \frac{2I_0}{\pi} \delta(r - b) \vartheta \left[\frac{\sin\left(\frac{k_z L}{2} + m\vartheta\right)}{\frac{k_z L}{2} + m\vartheta} \right] + \frac{iI_0}{m\pi} \delta(r - b) \cdot \left[e^{i(k_z L/2)} e^{im\vartheta} - e^{-i(k_z L/2)} e^{-im\vartheta} \right]. \quad (99)$$

On the LHS we put the $J_\theta(r, m, k_z)$ value used in Wavecode, while on the RHS we put the mode numbers $m - k_z$ and the antenna inner radius b used into Wavecode;

¹"Modeling of profile effects for inductive helicon plasma sources" by Yiannis Mouzouris and John E. Scharer.

in order to satisfy Eq. (99) we are bound to tune the current I_0 value, the antenna type parameter ϑ and the antenna length L . Once we get these values we have a specific antenna type, with determined geometry, which is capable of exciting the m and k_z modes chosen.

7.3.1 $m = 0$ case

For the $m = 0$ case, we get $J_\theta = \delta(r - b) I_0$, which is a simple loop antenna with the following parameters:

- I_0 amperes, corresponding to the current density used in the Wavecode;
- no restrictions on k_z values;
- inner radius $b = 7.5 \cdot 10^{-3}m$.

7.3.2 $m = +1$ case

For the $m = +1$ case, since the antenna geometry depends on the axial mode number k_z , we picked one of the values considered in the test matrix simulations such as $k_z = -157$. Three antenna types are investigated to excite this modes :

1. Nagoya Type III (N3 , $\vartheta = 0$);
2. Half-Helical (HH , $\vartheta = \pi/2$);
3. Half-Turn Helical (H1 , $\vartheta = \pi$).

Considering the current density value implemented along the azimuthal direction into the Wavecode $J_\theta(r, m, k_z)$, we obtain the following antennas.

Nagoya Type III The Nagoya Type 3 antenna has the following parameters:

- azimuthal wavenumber = +1;
- axial wavenumber = $25m^{-1}$;
- $b = 7.5 \cdot 10^{-3}m$;
- $I_0 = 7.0A$;
- $L = 0.2898m$;

as shown schematically in Fig. 40:

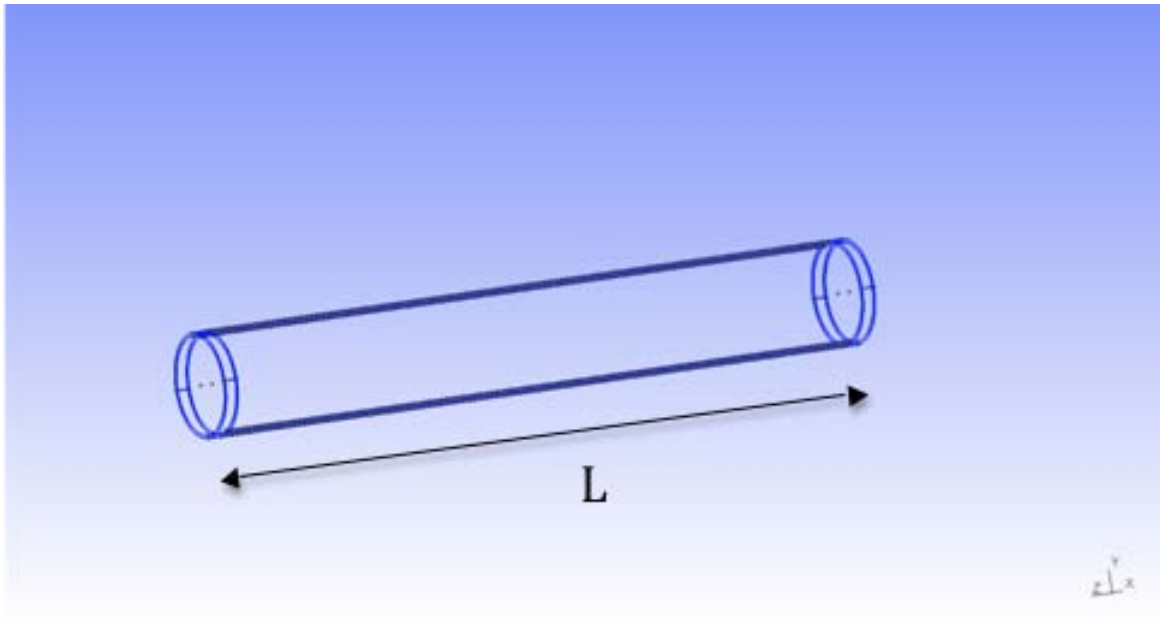


Figure 40: Nagoya Type III (N3 , $\vartheta = 0$).

Half Helical The Half Helical antenna has the following parameters:

- azimuthal wavenumber = +1;
- axial wavenumber = $25m^{-1}$;
- $b = 7.5 \cdot 10^{-3}m$;
- $I_0 = 7.0A$;
- $L = 0.1987m$;

as shown schematically in Fig. 41:

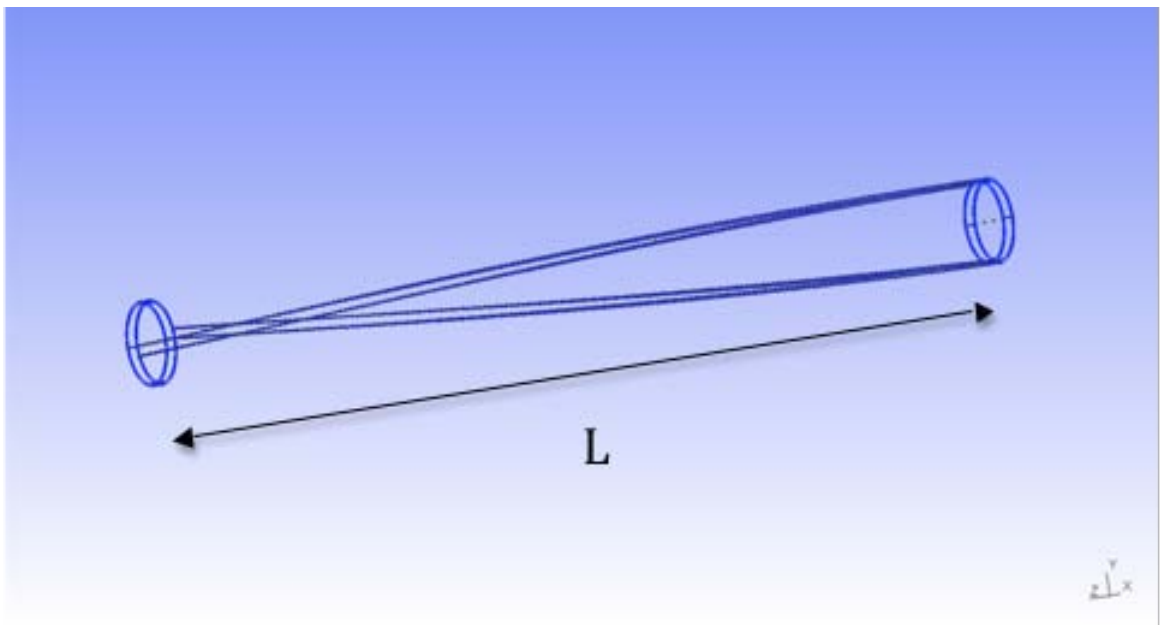


Figure 41: Half Helical (HH , $\vartheta = \pi/2$).

Half Turn The Half Turn antenna has the following parameters:

- azimuthal wavenumber = +1;
- axial wavenumber = $25m^{-1}$;
- $b = 7.5 \cdot 10^{-3}m$;
- $I_0 = 7.0A$;
- $L = 0.497m$;

as shown schematically in Fig. 42:

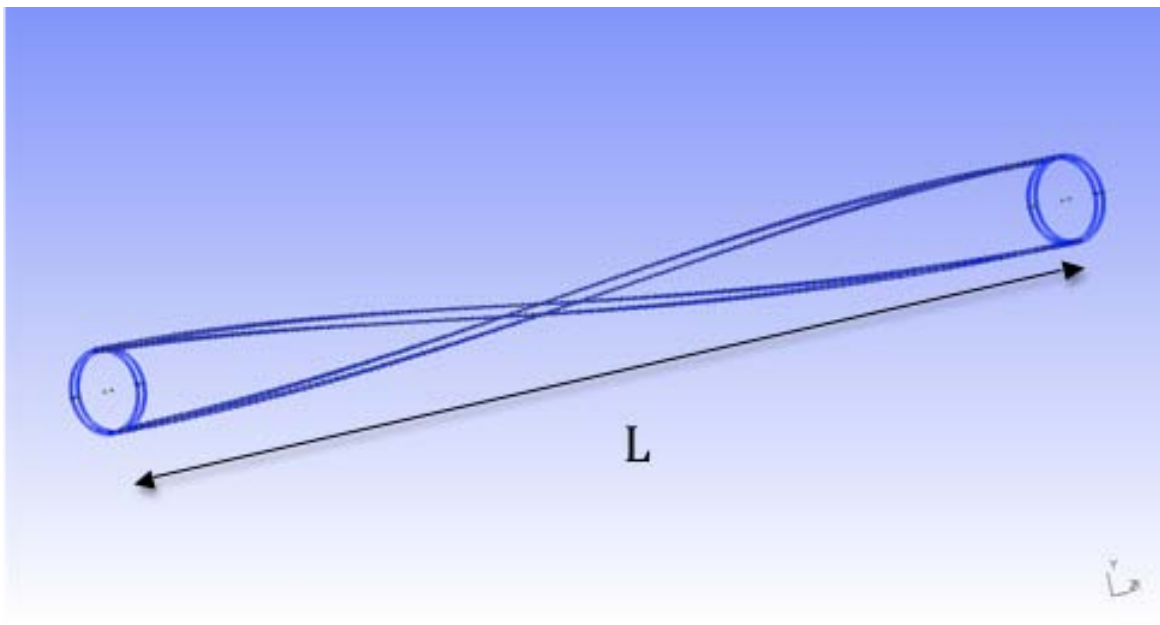


Figure 42: Half-Turn Helical (Ht , $\vartheta = \pi$).