DTU Physics
Department of Physics

Special course assignment

Advanced 3D Simulation of Electromagnetic Field Effects on Tungsten ion Transport in a Fusion Reactor

By

Lars Thor Sørensen (s194073)

Supervisors

Anders Henry Nielsen
Alexander Simon Thrysøe

Special Course
Technical University of Denmark
04/06-2024

Contents

1 Introduction

The next generation tokamak, ITER, which is currently under construction, has decided that all plasma facing components (PFC) inside the tokamak be made of tungsten [1]. Tungsten (W) is a favorable PFC because it has an extremely highly melting point, high heat conductivity, low tritium retention rate, and low erotion rate [2].

Impurity particles appear in the tokamak due to interactions between the plasma and the plasma facing components, such as physical sputtering or chemical erotion [2]. Therefore tungsten particles appear in the plasma, and some are transported toward the core of the plasma which can deteriorate plasma performance [3, 2]. Tungsten being a heavy atom with a high atomic number $Z = 74$ is only partially ionized in the plasma, even for central core temperatures of $T_\mathrm{e} \sim [10, 30]$ keV [1]. This causes large radiative power losses from the plasma by means of line emission from the partially ionized tungsten atoms excited by electron impact [1]. Further on, the tungsten particles can cause fuel dilution [3, 2]. Therefore the accumulation of tungsten impurity in the core needs to be limited, e.g. the concentration in ITER is expected to be limited to a few 10^{-5} [1, 3].

For this reason the understanding of impurity transport, and in particular tungsten transport, has become one of the most critical issues for future tokamak operation [3]. However actual measurements of impurity transport in the plasma edge of tokamaks is considered a great experimental challenge, which has yet to be overcome, and the field is still considered in its infancy [4]. Fortunately though, a lot of theoretical and computational work has been done [4].

This paper will first present the equations of motion for a particle moving in 3-dimensional electromagnetic fields, and how to numerically solve the first order initial value problem via a fourth order Runge-kutta method. A cylindrical transformation of these equations of motion will then be derived, due to the toroidal geometry of a tokamak. Then some bench-marking tests addressing the performance of the code will be presented. Finally a simulation on an advanced 3-dimensional tungsten ion transport was done using data from the FELTOR code.

2 Theory

Notation

Vectors are given in bold font, i.e. $\boldsymbol{x}$ is the position vector, so using the Einstein summation notation $\boldsymbol{x} = x_i \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ are the basis vectors of unity length and mutual orthogonal, x_i is the i'th coordinate or component. The L2 length of a vector e.g. $|\boldsymbol{v}| = \sqrt{v_i v_i}$ is simply denoted as v. Furthermore, $(\nabla)_i = \boldsymbol{e}_i \partial_i$.

2.1 Equations of motion and Runge-kutta

The governing equation for the particle motion inside a tokamak builds on newtons second law, that momentum $\boldsymbol{p}(\boldsymbol{r}(t), t) = m(t)\boldsymbol{v}(t) = m(t)\partial_t \boldsymbol{r}(t)$ is changed by the total force acting on the particle i.e. $\partial_t \boldsymbol{p} = \partial_t (m\boldsymbol{v}) = \boldsymbol{F}_\mathrm{tot}$, where $\boldsymbol{r}(t)$ denotes the time dependent position vector, $\boldsymbol{v}(t) = d_t \boldsymbol{r}(t) = \partial_t \boldsymbol{r}(t)$ the velocity of the particle and $\boldsymbol{F}_\mathrm{tot} = m\partial_t^2 \boldsymbol{r}(t)$ the total force on the particle. To ease notation $\boldsymbol{r}(t)$ is written as $\boldsymbol{r}$. In a tokamak gravity can safely be ignored, and we assume only the electromagnetic force, i.e. the Lorentz force is relevant $\boldsymbol{F}_\mathrm{tot} = \boldsymbol{F}_\mathrm{L}$ and that our particle is non-relativistic, and finally that mass is constant

$$\begin{aligned} \boldsymbol{F}_\mathrm{tot}(\boldsymbol{r}, t) &= \partial_t \boldsymbol{p}(t) \\ m\partial_t^2 \boldsymbol{r} &= q\big(\boldsymbol{E}(\boldsymbol{r}, t) + \boldsymbol{v} \times \boldsymbol{B}(\boldsymbol{r}, t)\big) \end{aligned} \tag{1}$$

This vector equation is second order differential equation in $\boldsymbol{r}$ and is typically reduced to first order by introducing [5]:

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Where $\boldsymbol{y}_1 = \boldsymbol{r}$ and $\boldsymbol{y_2} = \dot{\boldsymbol{r}} = d_t \boldsymbol{r} = \partial_t \boldsymbol{r}$ and both are written in Cartesian basis i.e. $\boldsymbol{r} = x\,\boldsymbol{e}_x + y\,\boldsymbol{e}_y + z\,\boldsymbol{e}_z$, so $\dot{\boldsymbol{r}} = (\dot{\boldsymbol{r}})_i \boldsymbol{e}_i$ (because the cartesian unit vectors have no time dependence), where Einstein summation notation has been used and $\boldsymbol{e}_i$ denote mutually orthogonal unit vectors, and $(\dot{\boldsymbol{r}})_i$ is the i'th component of $\dot{\boldsymbol{r}}$. The velocity vector is $\boldsymbol{v} = \dot{\boldsymbol{r}} = \boldsymbol{y}_2$ and the acceleration is $\boldsymbol{a} = \ddot{\boldsymbol{r}} = m^{-1}\boldsymbol{F}_L$. Then

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}) = \begin{bmatrix} \boldsymbol{y}_2 \\ \frac{1}{m}\boldsymbol{F}_L \end{bmatrix} \tag{2}$$

Where the notation is kept to match that of [5], i.e. denoting $\dot{\boldsymbol{y}}$ as $\boldsymbol{f}$. Now $\boldsymbol{f}$ only depends on $\boldsymbol{y}$ and t, so Eq. (2) is an initial value problem that is first order in $\boldsymbol{y}$. We can evaluate $\boldsymbol{f}(t, \boldsymbol{y})$ at a given $(t, \boldsymbol{r}, \boldsymbol{v})$ if we know $\boldsymbol{E}(t, \boldsymbol{r})$, $\boldsymbol{B}(t, \boldsymbol{r})$ and then integrate in time to obtain $\boldsymbol{y}$. There is some choice as to which numerical time integration method one chooses. In this project the fourth order Runge-kutta method is used as the finite difference formula to evaluate $\boldsymbol{y}_{j+1}$ from the $\boldsymbol{y}_j$ at time t_j. Runge-kutta of order 4 is typically denoted RK4, and the difference formula is given as [5]

$$\boldsymbol{y}_{j+1} = \boldsymbol{y}_j + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{3}$$

where

$$\boldsymbol{k}_1 = k\boldsymbol{f}_j$$
$$\boldsymbol{k}_2 = k\boldsymbol{f}(t_j + \frac{k}{2}, \boldsymbol{y}_j + \frac{1}{2}\boldsymbol{k}_1)$$
$$\boldsymbol{k}_3 = k\boldsymbol{f}(t_j + \frac{k}{2}, \boldsymbol{y}_j + \frac{1}{2}\boldsymbol{k}_2)$$
$$\boldsymbol{k}_4 = k\boldsymbol{f}(t_j + k, \boldsymbol{y}_j + \boldsymbol{k}_4)$$

where a fixed time step k has been used, so $t_j = jk$ for $j = 0, 1, 2, ..., M$ and $k = t_{j+1} - t_j$.

In this project a numerical framework for solving the equation of motion by using a Runge-kutta-4 approach was implemented in python. To benchmark this code some simple and analytical choices for $\boldsymbol{E}$ and $\boldsymbol{B}$ were used. The goal was then to use simulated fields for $\boldsymbol{E}$ and $\boldsymbol{B}$ from the FELTOR code. In that case $\boldsymbol{E}$ and $\boldsymbol{B}$ would be given on some discretized grid, and an interpolation-scheme should be used to interpolate $\boldsymbol{E}, \boldsymbol{B}$ between grid points in space. In this project the fields were taken as stationary in time, but one could also let $\boldsymbol{E}, \boldsymbol{B}$ depend on time and interpolate in a time grid as well. The FELTOR data is given in cylindrical coordinates, due to the toroidal symmetry of a Tokamak, so it would be sensible to interpolate in a cylindrical coordinate system. If one visualizes the discretization of the 3 cylindrical variables it would be a rectangle in cylindrical coordinate space, whereas the same grid would be cylindrical in a Cartesian coordinate system. When one wants to interpolate a field that is given in cylindrical coordinates it would therefore be advantageous to interpolate in this cylindrical coordinate space, such that there is equal spacing between grid points in order to avoid an error or bias of interpolating between points on a curved grid. Therefore a transformation of the equation of motion into cylindrical was needed.

Originally the plan was to define the equations of motion in a cylindrical basis, then the simulation could be integrated in time, entirely in cylindrical coordinates. Alternatively one could keep the integration of the equations of motion in Cartesian coordinates, but then have an intermediate step where the cylindrical coordinates of $\boldsymbol{r}$ were calculated, then an interpolation of $\boldsymbol{E}$ and $\boldsymbol{B}$, and then finally step forward in time using Runge-kutta on the Cartesian equation of motion.

These approaches should ideally output the same ion trajectory for a given simulation, and it was decided to go with the latter approach first. However the derivation of the equation of motion in cylindrical coordinates was already done, so it will be presented here as well. For the future the cylindrical equation of motion should probably be implemented in the particle trajectory code, as it is expected that the simulation time would be faster, because one would cut out the intermediate step of calculating the cylindrical coordinates for each calculation step.

2.2 Cylindrical transformation

In order to transform the coordinate system from Cartesian (x, y, z^{cart}) to cylindrical coordinates $(s, \phi, z^{\text{cyl}})$ we need their $s(x, y, z)$, $\phi(x, y, z)$, and $z^{\text{cyl}}(x, y, z^{\text{cart}})$, s being the radial coordinate from the z-axis, ϕ the azimuthal angle (angle around from x-axis) and z the same as in Cartesian.

$$
\begin{aligned}
s(x, y, z) &= (x^2 + y^2)^{\frac{1}{2}} \\
\phi(x, y, z) &= \tan^{-1}(\frac{y}{x}) \\
z^{\text{cyl}}(x, y, z^{\text{cart}}) &= z^{\text{cart}}(s, \phi, z^{\text{cyl}})
\end{aligned}
$$

Likewise to transform from cylindrical coordinates back to Cartesian coordinates one would use

$$
\begin{aligned}
x(s, \phi, z) &= s\cos\phi \\
y(s, \phi, z) &= s\sin\phi \\
z^{(\text{cart})}(s, \phi, z) &= z^{(\text{cyl})}
\end{aligned}
$$

Since the z coordinate is unchanged we will just drop the superscript. In cartesian coordinates the position vector is given by $\boldsymbol{r}(t) = r_i(t)\boldsymbol{e}_i$, where $(r_x, r_y, r_z) = (x, y, z)$, again where we have used Einstein's index notation (same index imply summation), $i = x, y, z$, and it has been made explicit that the unit vectors are not time dependent. Any vector $\boldsymbol{A}$ can be written in its components A_i for a given mutually orthogonal basis $\boldsymbol{e}_i$

$$\boldsymbol{A} = A_i\boldsymbol{e}_i \tag{4}$$

For the cylindrical coordinates a new basis $\boldsymbol{e}_k$ with $k = s, \phi, z$ and the position vector is given by

$$\boldsymbol{r}(t) = s(t)\boldsymbol{e}_s(\phi(t)) + z(t)\boldsymbol{e}_z$$

Where $\boldsymbol{e}_s$, $\boldsymbol{e}_\phi$ and $\boldsymbol{e}_z$ are the cylindrical unit vectors for the radius, azimuthal angle and the z component. To ease notation the time dependence in $s(t), \phi(t), z(t)$ will be dropped, so we will just write s, ϕ, z. So we see that $(\boldsymbol{r})_\varphi = 0$, however now the unit vector has a ϕ dependence.

The direction of the unit vectors are defined by how the position vector changes due to that parameter while keeping the other parameters constant

$$\boldsymbol{x}_k = \partial_k\boldsymbol{r}$$

Normalizing these vectors to unity gives the unit vectors

$$\boldsymbol{e}_k = \frac{\boldsymbol{x}_k}{|\boldsymbol{x}_k|}$$

So the cylindrical unit vectors can be found

$$
\begin{aligned}
\boldsymbol{x}_s &= \partial_s\boldsymbol{r} = \boldsymbol{e}_s(\phi) = \cos\phi\boldsymbol{e}_x + \sin\phi\boldsymbol{e}_y \\
\boldsymbol{x}_\phi &= s\partial_\phi\boldsymbol{e}_s(\phi) = s(-\sin\phi\boldsymbol{e}_x + \cos\phi\boldsymbol{e}_y) \Rightarrow \\
\boldsymbol{e}_s &= \cos\phi\boldsymbol{e}_x + \sin\phi\boldsymbol{e}_y \\
\boldsymbol{e}_\phi &= -\sin\phi\boldsymbol{e}_x + \cos\phi\boldsymbol{e}_y \\
\boldsymbol{e}_z &= \boldsymbol{e}_z
\end{aligned}
$$

In the Cartesian coordinate system the time derivatives of the unit vectors vanish, since they are all constant, however in the cylindrical coordinate system they depend on the azimuthal angle $\phi(t)$ meaning they also have time dependence, $\boldsymbol{e}_s(t), \boldsymbol{e}_\phi(t)$.

$$d_t \boldsymbol{e}_s = (-\sin\phi\, \boldsymbol{e}_x + \cos\phi\, \boldsymbol{e}_y)\partial_t\phi = \boldsymbol{e}_\phi \partial_t\phi = \dot{\phi}\, \boldsymbol{e}_\phi \tag{5}$$

$$d_t \boldsymbol{e}_\phi = -(\cos\phi\, \boldsymbol{e}_x + \sin\phi\, \boldsymbol{e}_y)\partial_t\phi = -\boldsymbol{e}_s \partial_t\phi = -\dot{\phi}\, \boldsymbol{e}_s \tag{6}$$

Now looking at the governing equation Eq. (1), expressing $\boldsymbol{E}, \boldsymbol{B}$ fields in cylindrical coordinates is just as in Eq. (4), so we just need the velocity and acceleration, i.e. $\dot{\boldsymbol{r}}, \ddot{\boldsymbol{r}}$ expressed in cylindrical coordinates. The position $\boldsymbol{r}(t)$ only has s, z components i.e. $r_\phi = 0$, but two of the unit vectors have $\phi(t)$ dependence, and by using Eq. (5) and Eq. (6) we get the equations of motion in cylindrical coordinates

$$\begin{aligned}
&\boldsymbol{F}(\boldsymbol{r}, t) = m\partial_t^2 \boldsymbol{r} = q\left(\boldsymbol{E}(\boldsymbol{r}, t) + \partial_t \boldsymbol{r} \times \boldsymbol{B}(\boldsymbol{r}, t)\right) \Leftrightarrow \\
&a_s = \ddot{s} - s\dot{\phi}^2 = \frac{q}{m}(E_s + \dot{\phi} s B_z - \dot{z} B_\phi) \\
&a_\phi = 2\dot{s}\dot{\phi} + s\ddot{\phi} = \frac{q}{m}(E_\phi + \dot{z} B_s - B_z \dot{s}) \\
&a_z = \ddot{z} = \frac{q}{m}(E_z + \dot{s} B_\phi - \dot{\phi} s B_s)
\end{aligned}$$

Where we have used that

$$\begin{aligned}
&\boldsymbol{v} = \partial_t \boldsymbol{r} = \partial_t(s\boldsymbol{e}_s + z\boldsymbol{e}_z) = \dot{s}\boldsymbol{e}_s + \dot{\boldsymbol{e}}_s s + \dot{z}\boldsymbol{e}_z = \dot{s}\boldsymbol{e}_s + \dot{\phi} s \boldsymbol{e}_\phi + \dot{z}\boldsymbol{e}_z \\
&\boldsymbol{a} = \partial_t \boldsymbol{v} = (\ddot{s}\boldsymbol{e}_s + \dot{s}\dot{\boldsymbol{e}}_s) + (\ddot{\phi} s \dot{\phi} + \dot{\phi}\dot{s}\boldsymbol{e}_\phi + \dot{\phi} s \dot{\boldsymbol{e}}_\phi) + \ddot{z}\boldsymbol{e}_z = (\ddot{s} - s\dot{\phi}^2)\boldsymbol{e}_s + (2\dot{s}\dot{\phi} + s\ddot{\phi})\boldsymbol{e}_\phi + \ddot{z}\boldsymbol{e}_z \\
&\ddot{\boldsymbol{e}}_s = \partial_t(\dot{\phi}\boldsymbol{e}_\phi) = -\dot{\phi}^2 \boldsymbol{e}_s + \ddot{\phi}\boldsymbol{e}_\phi \\
&\boldsymbol{v} \times \boldsymbol{B} = (\dot{s}\boldsymbol{e}_s + \dot{\phi}\boldsymbol{e}_\phi s + \dot{z}\boldsymbol{e}_z) \times (B_s \boldsymbol{e}_s + B_\phi \boldsymbol{e}_\phi + B_z \boldsymbol{e}_z) \\
&= (\dot{\phi} s B_z - \dot{z} B_\phi)\boldsymbol{e_s} + (\dot{z} B_s - B_z \dot{s})\boldsymbol{e_\phi} + (\dot{s} B_\phi - \dot{\phi} s B_s)\boldsymbol{e_z}
\end{aligned}$$

The derived $\boldsymbol{a}$, $\boldsymbol{v}$ seems to match that of a classical mechanics book [6]. This cylindrical coordinate system is the most traditional one, and used by e.g. Griffiths, however FELTOR uses data in the (R, Z, φ) format, which probably is to enhance to the programming speed of accessing φ coordinates. The order of the azimuthal angle and z coordinates are thus permuted and in order to keep the coordinate system right handed, φ must go in the negative direction (out of plane, clockwise viewed from above). It also seems that φ is taken from the y axis instead of the x-axis (i.e. $\varphi = \frac{\pi}{2} - \phi$). We therefore have that

$$\begin{aligned}
R &= \sqrt{(x^2 + y^2)} \\
\varphi &= \tan^{-1}\left(\frac{x}{y}\right) \\
Z &= z \\
x &= R\sin(y) \\
y &= R\cos(y)
\end{aligned}$$

Then

$$\begin{aligned}
\boldsymbol{e}_R &= \sin\varphi\, \boldsymbol{e}_x + \cos\varphi\, \boldsymbol{e}_y \\
\boldsymbol{e}_Z &= \boldsymbol{e}_z \\
\boldsymbol{e}_\varphi &= \cos\varphi\, \boldsymbol{e}_x - \sin\varphi\, \boldsymbol{e}_y
\end{aligned} \tag{7}$$

We see then that $\partial_t \boldsymbol{e}_R = \dot{\varphi}\, \boldsymbol{e}_\varphi$ and $\partial_t \boldsymbol{e}_\varphi = -\dot{\varphi}\, \boldsymbol{e}_R$, exactly like before, so then we already know the expression for $\boldsymbol{v}$ and $\boldsymbol{a}$. The cross product expression is however a bit different due to the reversed order of Z, φ. The equations of motion are then

$$a_R = \ddot{R} - R\dot{\varphi}'^2 = \frac{q}{m}(E_R + \dot{Z}B_\varphi - \dot{\varphi}' R B_Z) \tag{8}$$

$$a_Z = \ddot{Z} = \frac{q}{m}(E_Z + \dot{\varphi}' R B_R - \dot{R}B_\varphi) \tag{9}$$

$$a_\varphi = 2\dot{R}\dot{\varphi} + R\ddot{\varphi}' = \frac{q}{m}(E_\varphi + B_Z\dot{R} - \dot{Z}B_R) \tag{10}$$

Implementing these into the code, one could then solve the governing equation of motion Eq. (1) in the cylindrical grid, where interpolation is on a rectangular equally spaced grid.

2.3 Charged particle motion inside a Tokamak

In a uniform $\boldsymbol{B}$ field the charged particle is confined to a circular motion transverse to the $\boldsymbol{B}$ field, while not experiencing any field in the direction parallel to the $\boldsymbol{B}$ field. This is seen from the equation of motion

$$m d_t \boldsymbol{v} = q\boldsymbol{v} \times \boldsymbol{B} \tag{11}$$

The characteristic time scale is then seen to be

$$\omega_\mathrm{c} = \frac{|q|B}{m} \tag{12}$$

Where ω_c is the cyclotron angular frequency. The solution, if $\boldsymbol{B}$ is aligned along the z axis, is

$$\boldsymbol{x} = x_{\mathrm{gc}0} + r_\mathrm{L} \sin(\omega_c t + \gamma_0) \tag{13a}$$

$$\boldsymbol{y} = y_{\mathrm{gc}0} \mp r_\mathrm{L} \cos(\omega_c t + \gamma_0) \tag{13b}$$

$$z = z_{\mathrm{gc}0} + v_\parallel t \tag{13c}$$

where $v_\perp$ and the phase γ_0 come from initial conditions, and $(x_{\mathrm{gc}0}, y_{\mathrm{gc}0}, z_{\mathrm{gc}0})$ are initial positions. The motion is thus a spiral or helical trajectory since the magnetic field only influences the motion perpendicular to $\boldsymbol{B}$. The gyro-period (for one circular revolution) is $\tau_\mathrm{gyro} = f_\mathrm{c}^{-1} = \frac{2\pi}{\omega_\mathrm{cs}}$ and the radius of this circular motion is the Larmor radius

$$r_\mathrm{L} = \frac{v_\perp}{\omega_\mathrm{cs}} \tag{14}$$

When an arbitrary force $\boldsymbol{F}$ acts on the particle the ion will drift with a velocity

$$\boldsymbol{v}_\mathrm{F} = \frac{\boldsymbol{F} \times \boldsymbol{B}}{qB^2} \tag{15}$$

If the force fields inside a tokamak are smooth, i.e. vary on a much larger length scale than the larmor radius one finds that all the drifts are the $\boldsymbol{E} \times \boldsymbol{B}$ drift, $\boldsymbol{\nabla} B$ drift, curvature drift, gravitational drift and polarization drift are all of the possible drifts. In a fusion plasma gravity is usually neglected. Due to the toroidal symmetry in a tokamak the toroidal magnetic field decays as r^{-1} so the particles will experience a $\boldsymbol{\nabla} B$ force. Because of the torus configuration a curvature drift will also be experienced. We can use the expected drift velocity Eq. (15) later on to test if the simulation runs as predicted in an analytical $B \propto r^{-1}$ field, and by inserting electric fields in the simulation. In the time stationary FELTOR data electrical fields are also present, which would also give rise to a $\boldsymbol{E} \times \boldsymbol{B}$ drift.

3 Simulation

3.1 The Impurity particle trajectory code

The code framework computes the trajectory of point particles that have a given mass m, charge q, initial velocity $\boldsymbol{v}_0$ and position $\boldsymbol{r}_0$ in a given $\boldsymbol{E}$ and $\boldsymbol{B}$ and is therefore called Impurity particle trajectory or IPT. For all the simulations in this report the impurity particle has been given a mass of $m_{\mathrm{W}} = 183.84$ amu and only singly ionized $q = q_e = 1.602\ 10^{-19}$ C, the initial velocity and position is chosen to be some specific value for each simulation. In the future one could implement that the particles be initialized from e.g. a maxwellian distribution and that they be allowed to be ionized by e.g. using a monte carlo method.

3.1.1 Code specifics

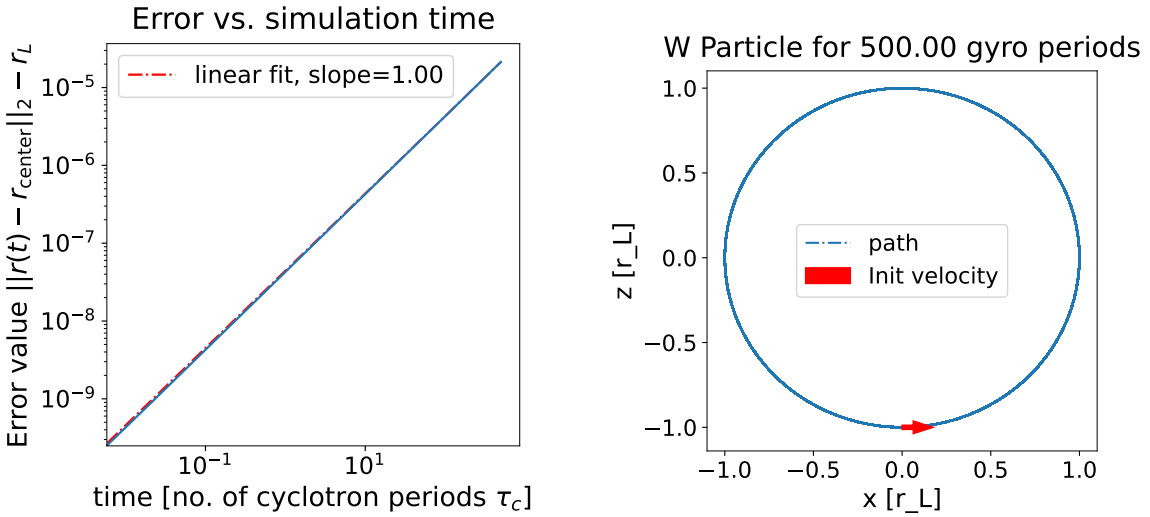
The code was written in python in an object oriented programming manner. The four main objects (classes in python) are the "Particle" object, the "ParticleSystem" object, the "f" object, and finally the "RK4" object as found in "IPT.py". The Particle is an object which contains the instantaneous position, velocity, charge at time t, while the "ParticleSystem" is simply a list of these Particle objects. The "f" object is an object that can calculate $\boldsymbol{f}$ in Eq. (2) for different analytical $\boldsymbol{B}$ and $\boldsymbol{E}$ fields and interpolate on a $\boldsymbol{B}$ and $\boldsymbol{E}$ grid if represented in cylindrical coordinates. The "RK4" object uses "f" and "ParticleSystem" as inputs and is able to calculate the particle trajectory by stepping $\boldsymbol{y}$ forward in time by means of the fourth order Runge-kutta method Eq. (2.1). Currently the "IPT.py" contains the code used for the performance tests and sanity plot, whereas the "IPT_3.py" has is an upgraded version which has the interpolation method as well.

3.1.2 cylindrical coordinates

The current version solves the equation of motion in Cartesian coordinates and then interpolates $\boldsymbol{E}$ and $\boldsymbol{B}$ in cylindrical coordinates by expressing $\boldsymbol{r}$ in the (R, Z, φ) basis. The cylindrical coordinate system has a singularity at $R = 0$, but for simulations in a tokamak the particles will not be near $R = 0$ anyways. The azimuthal angle φ was found by using $\varphi = \tan^{-1}(\frac{x}{y})$ and three if statements ensured that the angle be kept within $\varphi \in [0, 2\pi]$.

3.2 Performance test

Before beginning to simulate using advanced $\boldsymbol{E}$ and $\boldsymbol{E}$ and on cylindrical coordinates, it was useful to check that the code was implemented properly and would yield known theoretical results as well as yield numerical errors in the correct order of magnitude. The numerical truncation error at some time t of using a Runge-Kutta method of fourth order should decrease as $\mathcal{O}(k^4)$ where k is the time step. To first check if the numerical error had a dependence on the time t at which this error was evaluated a plot as seen in Fig. 1a was made. Fig. 1b shows the simplified case where the magnetic field is uniform in a Cartesian space, $\boldsymbol{B} = B_0 \boldsymbol{e}_y$ with $B_0 = 1$ T and with $v_\perp = 10^3$ m s^{-1}, meaning $\tau_c = 1.2 \cdot 10^{-5}$ s. The axes are normalized by the Larmor radius $r_{\mathrm{L}} = 1.92$ mm, and the graph has been centered around the gyro-center $(x_{\mathrm{gc0}}, z_{\mathrm{gc0}})$. One would expect a circular motion in the transverse directions, i.e. in the $\boldsymbol{e}_x$ and $\boldsymbol{e}_z$ directions, as is also evident in Fig. 1b, where trajectory is projected on these directions for each time-step from $t = 0$ to $t = 500\ \tau_c$ where the time-step $k = 10^{-2}\ \tau_c$. From visual inspection of the graph the 500 circular trajectories lie almost directly on top of each other. In reality the radius shrinks a little bit throughout the simulation, and this is exactly what is measured as the error value in Fig. 1a. We know the radius shouldn't change analytically, and the L2 norm deviation from a radius of r_L is then a measure of the simulation error. It is evident from the linear increase with a slope of 1 in a double-logarithmic plot in Fig. 1a that the error depends linearly on the time at which it was evaluated. So then in principle it would



(a) (b)

Figure 1: In (a): Error value associated with the numerical integration (fourth order Runge-Kutta method) $|\boldsymbol{r}(t) - r_{\text{center}}| - r_{\text{L}}$, where $r_{\text{L}} = 1.92$ mm, as a function of simulation time t in units of gyro-periods τ_{c} in a double-logarithmic plot. In (b) the trajectory of the Tungsten ion is shown and since the magnetic field is uniform the analytical solution is a circle which has a constant radius r_{L}. So the distance to (0,0) should be r_{L} throughout the entire simulation. The deviation from that value is a measure for the numerical error. Note that there are $500 \cdot k^{-1} = 5 \cdot 10^4$ points plotted, but since they more or less overlap, only a a single circle is visible.

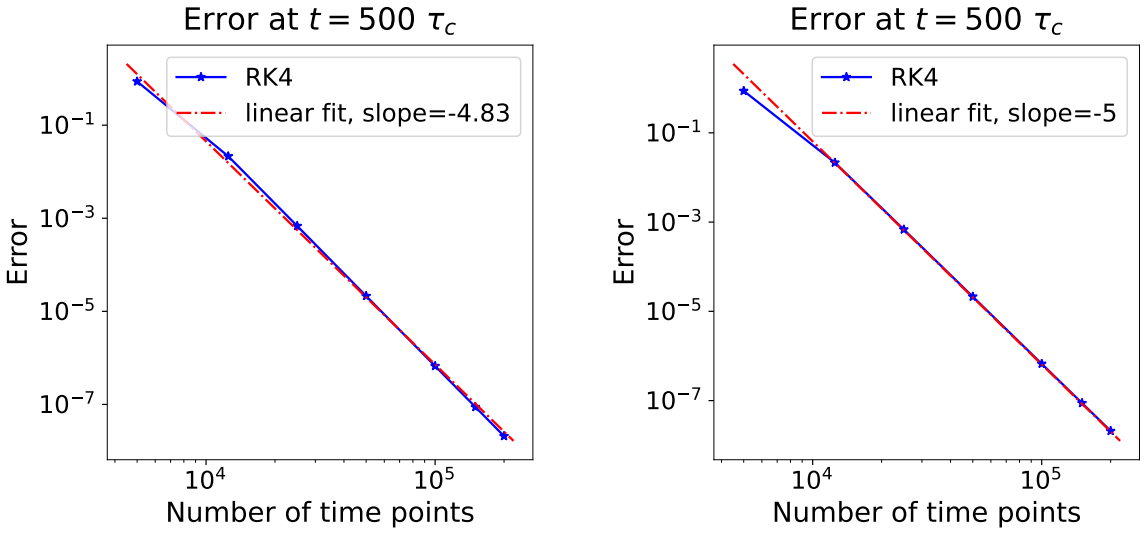
not really matter at which time t one evaluates the truncation error when trying to increase the number of time points, therefore $t = 500\tau_c$ was chosen. A plot of the truncation error versus number of time points is seen in Fig. 2a and Fig. 2b. All other parameters besides the number of time points used, and hence k, is kept constant. The decrease should in principle go as $n = 4$ for a Runge-Kutta [5], but it does however seem to be even better, and go as $n \sim 5$, which probably has to do with the high symmetry in a circular motion. Fig. 2a is plot with a standard linear fit where the residual sum squares was minimized, however as it seems the line starts out by overestimating, then underestimating and finally overestimating the error as the number of points is increased. Instead a manual fit was done by excluding the very first point and choosing the slope to be -5, which is seen in Fig. 2b, and it seems to fit the asymptotic slope better. In any case Fig. 2 seems to indicate that the code works as a fourth order Runge-Kutta method, since the decreasing slope is greater than 4 (expected to be due to the circular motion). Further on we see that the error is down to 10^{-5} for $5 \cdot 10^4$ time points, meaning $k = 10^{-2}\ \tau_c$. This k value is used in later simulations.

3.3 Sanity checks

Sanity check 1: ∇B drift

Now that the numerical truncation error has confirmed that the Runge-Kutta method is implemented correct some sanity checks would be nice, to be sure that the simulation can reproduce intuitive analytical results. It was desired to try and validate the ∇B drift velocity

$$\boldsymbol{v}_{\nabla B} = -\mu \frac{\nabla B \times \boldsymbol{B}}{qB^2} \tag{16}$$



(a) (b)

Figure 2: Truncation error (same error value as in Fig. 1) at $t = 500\ \tau_c$ versus number of time pounts in a double-lorarithmic plot. A linear fit in this double-logarithmic plot was done, and the slope is shown in the legend. A Runge-kutta method should decrease as $\mathcal{O}(4)$ i.e. a slope of -4.

where the grad B force $\boldsymbol{F} = -\mu \boldsymbol{\nabla} B$ has been inserted in Eq. (15), with the magnetic moment

$$\mu = \frac{m v_\perp^2}{2B} \tag{17}$$

The analytical magnetic field used in this simulation was still in the $\boldsymbol{e}_y$ direction but now had a x^{-1} dependence. This Cartesian magnetic field would thus imitate the expected R^{-1} dependence in a cylindrical system, i.e. the plot should looks as a poloidal projected trajectory (R, Z), with purely toroidal magnetic field in the $\boldsymbol{e}_\phi$ or $-\boldsymbol{e}_\varphi$. The analytical s^{-1} and the x^{-1} Cartesian version are

$$\boldsymbol{B}(s) = \boldsymbol{e}_\phi \frac{B_0 s_0}{s} \tag{18a}$$

$$\boldsymbol{B}(x) = \boldsymbol{e}_y \frac{B_0 x_0}{x} \tag{18b}$$

In Eq. (18b) x is used instead of the traditional major radius from the z axis $s = s_0 + r$, where B_0 is the magnetic field strength at some characteristic distance s_0 or x_0 (typically the magnetic axis), and r the radius from s_0. The drift velocity was then estimated from the simulation by computing the guiding center of the motion by the radius of curvature R_{curv} [7]:

$$R_{\text{curv}} = \frac{\left[1 + (\partial_x z)^2\right]^{\frac{3}{2}}}{\partial_x^2 z} \tag{19}$$

This would so to speak give the length R_{curv} to the local center of the radius of curvature. Then by taking the 90° rotating (in the positive out of plane direction) the unit vector parallel with the tangent (gotten from the derivative and inital position) the position of the guiding center for each position $\boldsymbol{r}_i$ $(x_{\text{gc}}, z_{\text{gc}})$ could be calculated. Then the velocity can be estimated for this guiding center by estimating the slope of $z_{\text{gc}}(t)$.

To start of with, a uniform $\boldsymbol{B}$ field was used, and the curvature should then just stay in the center of the circular motion, such a scenario is plotted in Fig. 3 using $B = 1\text{T}$ and $k = 0.01\tau_c$. In total 100 data-points are plotted in the figure, meaning that there is the initial point $\boldsymbol{r}_1$ and 99 calculated

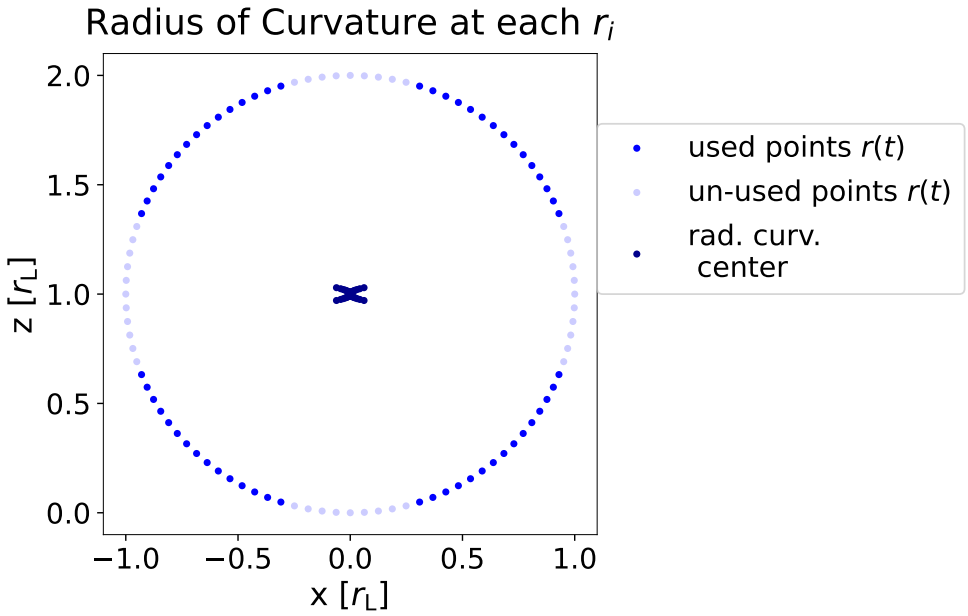


Figure 3: One gyro period, using $k = 10^{-2}\tau_c$, i.e. including the first initial datapoint $\boldsymbol{r}_1$ then 99 positions were simulated using a uniform $\boldsymbol{B}$ field. In this plot 3 types of coloured dots are seen, with a legend matching each. The transparent dots were not used when calculating the radius of curvature, and hence the center of the circular motion. This is because the numerical derivative diverges here. The 40% most extreme gradients were removed, i.e. only the gradients between quantile [0.1,0.4] and [0.6,0.9] were used. This effectively removed the numerical diverging places where the derivative would be extremely big or very close to zero. The filled blue dots are then the points used to calculate the center of the circular motion (dark-blue coloured dots in the center). One sees that there is some error (it is not directly in the center of the circle, but looks like a cross), but this cancels when calculating the mean value. The mean location $(x_{\mathrm{gc}0}, z_{\mathrm{gc},0}) - (0, r_{\mathrm{L}}) = (7.04 \cdot 10^{-9}, 1.34 \cdot 10^{-7})\, r_{\mathrm{L}}$ i.e. in principle a numerical 0.

points, so $\boldsymbol{r}_i$ with $i = 1..100$. When calculating the gradient $\partial_x z$ for this circular motion, especially the four corners of the circle become problematic. The numerical derivative will be enormous at the right/left parts of the circular motion, because lie directly vertically over each other. Furthermore the top and bottom part are problematic because the numerical gradient goes toward 0, since here the points lie directly horizontal to one another. Therefore only 60% of the data was used in the calculation of the radius of curvature, i.e. only the gradients between quantile [0.1,0.4] and [0.6,0.9] were used. These points that were used in this data-analysis are marked as solid blue dots in the figure, as indicated in the legend. The unused points are marked transparent and the dark-blue dots indicate the calculated position of the guiding center, by using the radius of curvature (i.e. the 60 somewhat-overlapping dark blue points in the center). As can be seen the calculated guiding center has some errors (looks like a cross, but should be a single dot in the center $(0, 1)$), but they seem to be systematic. Centering the data about $(0, 1)$ r_L i.e. subtracting it from the mean value for the guiding center a very small deviation was found $(x_{\mathrm{gc}0}, z_{\mathrm{gc},0}) - (0, r_\mathrm{L}) = (7.04 \cdot 10^{-9}, 1.34 \cdot 10^{-7})\, r_\mathrm{L}$ i.e. in principle a numerical 0. This is also what we would expect if the axes are placed in the center of the circular motion in a uniform field, because there would be no drifts and the guiding center should then stay at $(x_\mathrm{gc}, z_\mathrm{gc}) = (0, 0)$. Being confident that this data-analysis method would then be somewhat precise (down to a precision of 10^{-7} r_L) the same analysis was done for a gradient B drift scenario.

Using the analytical magnetic field Eq. (18b), for $B_0 = 1$ T at $x_0 = 1$ m and now choosing $v_\perp = v_x = 10^4$ m s^{-1} produced the plot in Fig. 4. The velocity was increased, to increase the $\boldsymbol{\nabla} B$ drift velocity, so now the larmor radius is 10 times larger. The colours are like before, and we again see some systematic deviation of the guiding center, (it is not a straight line). This is probably again due to the numerical errors in the radius of curvature calculation. Since we are using a Cartesian grid, no time dependence, and no $\boldsymbol{E}$ field in this simulation, only the $\boldsymbol{\nabla} B$ drift is present. Using Eqs. (19) and (16) the expected change in the height for a time $t = M\ k = 5\tau_c$ where $M = 500$ was $(\Delta z)_\mathrm{analytical} = v_{\boldsymbol{\nabla} B} M k = 0.359\ r_\mathrm{L}$ (i.e. the change in height dz after 5 gyro-periods). Estimating the simulated dz due to the gradient B drift, by the z-values of the end-points the value found was $(\Delta z)_\mathrm{sim} = 0.354\ r_\mathrm{L}$. This estimate comes quite close to the analytical one, and belief is that the simulation works properly, but that the deviation is due to the numerical errors in the gradient and exclusion of some of the points. For example, as can be seen in Fig. 4, the two endpoints are excluded in the start and end of the trajectory. Another way of calculating the guiding center and hence estimate the curvature drift could be to make a sliding average for a single gyro-period (τ_c would however depend on B). One should also note that the analytical expression for $v_{\boldsymbol{\nabla} B}$ comes from the first order guiding center equations, i.e. taylor expanding with $r_\mathrm{L} \ll \frac{B}{|\boldsymbol{\nabla} B|}$. Therefore the simulation might actually find the more correct Δz deviation due to the drift. In any case the simulation seems to work as it should.

Sanity check 2: $\boldsymbol{E}$ cancelling $\boldsymbol{v} \times \boldsymbol{B}$

From the equations of motion Eq. (1) it is evident that $\boldsymbol{E}$ and $\boldsymbol{v} \times \boldsymbol{B}$ can cancel each other. As can be seen in Fig. 5 this special case also appears in the simulation, where a straight motion is observed. This simulation or computation is done with a uniform $\boldsymbol{B} = B_0 \boldsymbol{e}_y$ with $B_0 = 1$ T, $\boldsymbol{v} = 10^3$ m s^{-1} $\boldsymbol{e}_x$ and uniform $\boldsymbol{E} = E_0 \boldsymbol{e}_z$ with $E_0 = -10^3$ V m^{-1} meaning $\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B} = \boldsymbol{0}$. There is then no gyration or deflection and just a straight motion is observed, this is the principle in a mass spectrometer. Again this builds our confidence that the simulation is working properly.

3.4 Advanced 3D electromagnetic fields: FELTOR

The FELTOR data contained the magnetic field components B_R, B_Z in the usual co-variant form, but B_φ component was in a contravariant form. From a double-logarithmic plot it was found that multiplying the coordinate with $R = \sqrt{g}$, where g is the determinant of the metric tensor, would yield the correct R^{-1} behavior, so then $B_\varphi = B^\varphi R$ where the superscript indicates a contra-variant

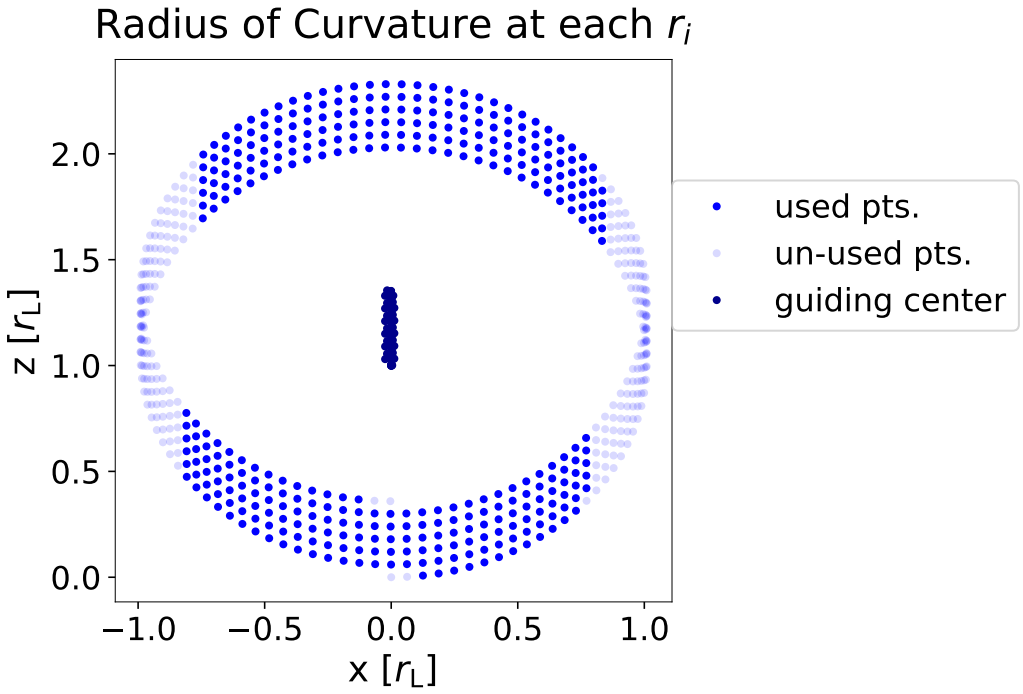


Figure 4: This plot is the same as in Fig. 3 now, however, using a magnetic field with an x dependence, $\boldsymbol{B} = x^{-1}B_0x_0\boldsymbol{e}_y$, using $B_0 = 1$ T at $x_0 = 1$ m, $v_\perp = 10^4$ m s^{-1}, and hence obtaining a curvature drift in the $\boldsymbol{e}_z$ direction, as seen from Eq. (15). The transparrent dots are used in the simulation, but not when computing the guiding center location, in order to avoid bad numerical gradients. The full blue dots (60% of data) was used in computing the dark-blue guiding center dots, there is one guiding center for each used point. The data used here was the gradients between quantile [0.2,0.8] i.e. the central 60% part of data. The ion is started at (0,0) and ends after $M = 600$, i.e. after 6 gyro-period (if the gyro-period is calculated at the initial point).

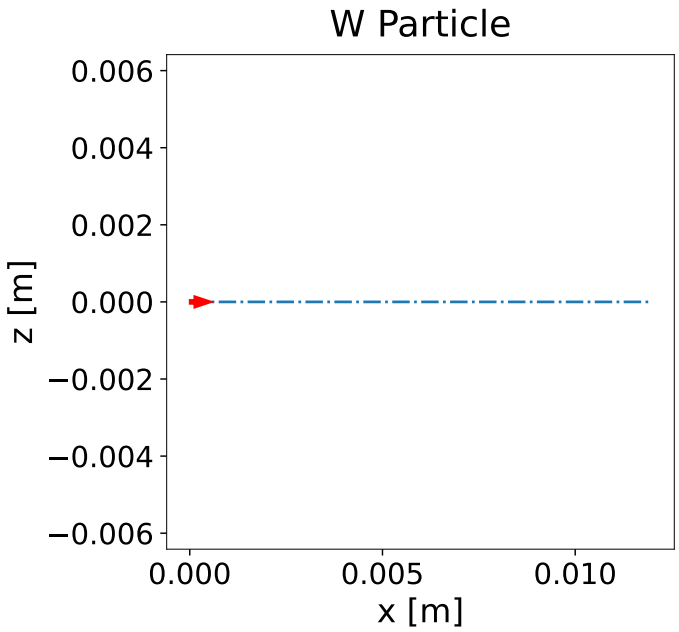


Figure 5: Special case with uniform $\boldsymbol{B} = B_0\boldsymbol{e}_y$ with $B_0 = 1$ T, $\boldsymbol{v} = 10^3$ m s^{-1} $\boldsymbol{e}_x$ and uniform $\boldsymbol{E} = E_0\boldsymbol{e}_z$ with $E_0 = -10^3$ V m^{-1}, then the $\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}$ cancels, there is no gyration and just a straight motion is observed, this is the principle in e.g. a mass spectrometer.

form. The $\boldsymbol{E}$ field had to be obtained by a gradient of the electrostatic potential. However it was not known how to express the gradient in contra-variant form, and it was not the scope of this project to go into the details of contra- and co-variant forms. Therefore only the $\boldsymbol{B}$ field data from FELTOR was considered. Any interpolation from an $\boldsymbol{E}$ field should however be straight-forward to implement in "f" object of the code. Looking at Fig. **??** we get a sense of the magnetic field in the poloidal plane at $\varphi = 1.56$ rad i.e. roughly at the starting position $\varphi = 1.57$ rad (roughly the x axis) which is where the simulated particle was initialized (black dot in figure). To be clear the particle was initialized at $\boldsymbol{r} = 1$ m $\boldsymbol{e}_x + 0.001$ m $\boldsymbol{e}_y = 1000$ mm $\boldsymbol{e}_R(\varphi = 1.57$ rad), where the units of R and Z are mm in the FELTOR data and φ is in radians. Note that in the current version of the IPT code the interpolation is a linear interpolater that will output a null-type if the point to be interpolated is outside the grid. This would make physical sense in the (R, Z) coordinates if the boundary is also e.g. the wall of the tokamak, but not in the φ direction. It is quite unfortunate in the φ grid, because the current data has a φ grid which goes between $[0.008, 6.29]$ rad. There will therefore be a gap in which the particle will be converted into a null-type. This should be debugged in the future, but a quick fix was to initialize the particle away from the problematic coordinates and set the $v_{\parallel} = 0$ avoiding the particle to reach to problematic null-type space. In Fig. 7 the calculated trajectory for $M = 2000$ time-steps of $k = 5.986 \cdot 10{-}8$ s is projected onto the poloidal plane, i.e. the 2000 (R, Z) values are shown. The red dot indicates the starting position and the blue is where the simulation ends. This plot shows that the simulation is for the most parts working properly. The Larmor radius from the initial position is $r_{\mathrm{L}} = 2.12$ mm, and will slightly decrease when R increases, but we see that the diameter quite well matches $2r_{\mathrm{L}}$. Further on, it is seen that ion gyrates in a left handed motion as it should, because $B_\varphi\boldsymbol{e}_\varphi$ is out of the plane. Finally the trajectory does drift in the vertical $\boldsymbol{e}_Z$ direction, which is expected for an charged particle in a $\boldsymbol{\nabla}B$ drift, but seeing as $\boldsymbol{\nabla}B$ is in the $-\boldsymbol{e}_R$ direction and $\boldsymbol{B} \approx B_\varphi\boldsymbol{e}_\varphi \approx B_\varphi(-\boldsymbol{e}_y)$ it should actually drift in the $-\boldsymbol{e}_Z$ direction. This is in contrast with the Cartesian version of this $\boldsymbol{\nabla}B$ drift in Fig. 4 were $\boldsymbol{B}$ was in the $\boldsymbol{e}_y$ direction so that ions should go in the $\boldsymbol{e}_z$ direction. There should therefore be some error in the code where the interpolation happens, since the Cartesian calculation worked properly. The interpolated magnetic field at the initial position is however $\boldsymbol{B}_{(R,Z,\varphi)} = (0.00182, 0.15075, 0.89655)$ T and is computed to

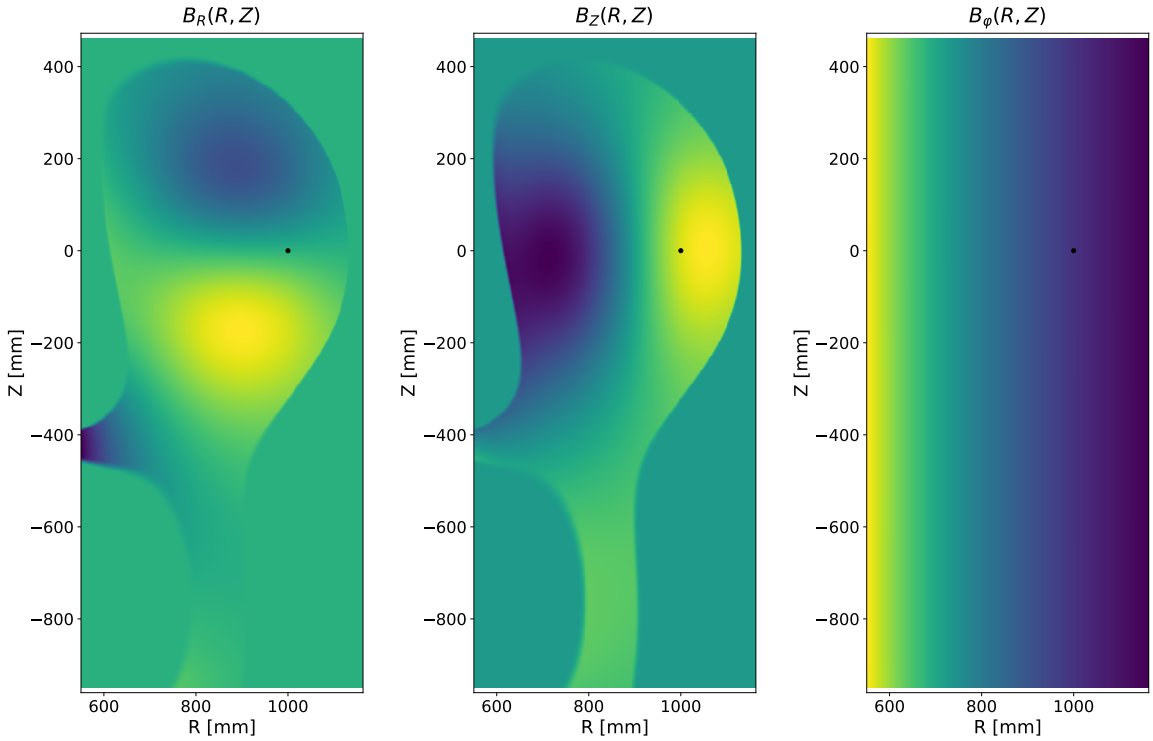


Figure 6: Plots of the magnetic field strength of each component, B_i with $i = (R, Z, \varphi)$ for the poloidal plane with $\varphi = 1.562$ rad, i.e. closest grid-point to the $\varphi = 1.57$ rad starting position. The black dot indicates the starting position of the simulation.

be $\boldsymbol{B}_{(x,y,z)} = (0.0027, -0.8965, 0.15075)$ in the Cartesian basis, which makes good sense seeing as $\boldsymbol{e}_\varphi \approx -\boldsymbol{e}_y$ there. It is still a bit unclear why there seems to be sign flip in the $\boldsymbol{\nabla} B$ drift.

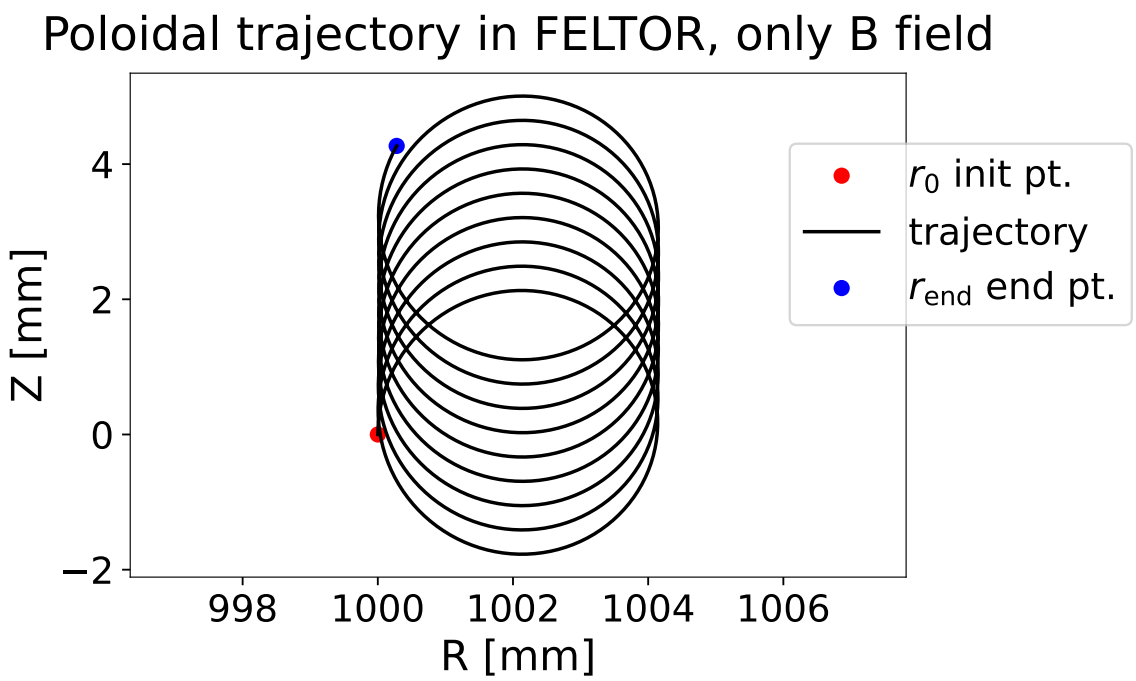


Figure 7: Particle trajectory for a singly ionized Tungsten atom, initialized at $\boldsymbol{r} = 1\text{ m}\ \boldsymbol{e}_x + 0.001\text{ m}\ \boldsymbol{e}_y = 1000\text{ mm}\ \boldsymbol{e}_R(\varphi = 1.57\text{ rad})$ (red). The $\boldsymbol{E}$ field from the FELTOR simulation has been avoided due to the electromagnetic field being expressed in contra-variant form. At the initial point $\boldsymbol{B} = 0.00193\text{ T}\ \boldsymbol{e}_R + 0.15\text{ T}\ \boldsymbol{e}_Z + 0.897\text{ T}\ \boldsymbol{e}_\varphi$ so $B = 0.909$ T, meaning the far dominating field is in the $\boldsymbol{e}_\varphi$ direction. This is reassuring because $\boldsymbol{B}_\varphi(R) \sim R^{-1}$ and we again see the a vertical $\boldsymbol{\nabla} B$ drift like observed in Fig. 4, however now it seems to go in the opposite direction, because the $\boldsymbol{B}$ field is flipped. The Larmor radius will vary throughout the circular Larmor motion, but will start out being $r_\text{L} = 2.12$ mm since $v_\perp = 10^3\text{m s}^{-1}$. This does very well agree with the graph. Furthermore also note that the ion gyrates in a left handed motion as it should, because $B_\varphi \boldsymbol{e}_\varphi$ is out of the plane.

4 Conclusion

Impurity particles are very important within fusion energy because they deteriorate plasma performance if they reach the core of the plasma. The particle motion and the trajectories these impurity particles undergo inside a tokamak are therefore very important. Describing the equations of motion of these particles by newtons second law and the Lorentz force, the trajectories were found by numerical integration using a Runge-kutta of fourth order. The equations of motion were presented in the basis for a cylindrical system (s, ϕ, z) and also for a cylindrical system with the order of the azimuthal angle and z axis interchanged (R, Z, φ). The motion or trajectories of ionized particles in a tokamak are composed of a fast gyration or Larmor motion, while the average position the guiding center may drift due to various forces inside the tokamak. Such forces and hence drifts include the $\boldsymbol{\nabla} B$ drift, $\boldsymbol{E} \times \boldsymbol{B}$ drift, the curvature drift, and polarization drift. A code was developed in this project in order to eventually simulate how impurity particles moved within advanced electromagnetic fields from a simulation of a tokamak called FELTOR. Before using the developed code to simulate particle trajectories on advanced data, some sanity checks and performance tests were made to check that the code was working properly. The decrease of the truncation error had a $\mathcal{O}(k^5)$ dependence, where $\mathcal{O}(k^4)$ is expected from a fourth order Runge-kutta integration method. The even better performance in the simulation is thought to be because of the circular motion. It was therefore concluded that code had a properly working Runge-kutta fourth order time integration method.

Next the motion in two different analytical magnetic fields was investigated where the local radius of curvature was used to estimate the guiding center location. The motion transverse to a uniform B field was expected to be circular, so by placing the axes in the theoretical guiding center, the guiding center should be located at $(0, 0)$ and not move throughout the simulation. Then by simulating a single gyro-motion by 100 time points, i.e. a time-step $k = 0.01\ \tau_c$, with $B = 1$ T and $v_\perp = 10^3$ m s^{-1} a circular motion around $(0, 0)$ should be observed with a Larmor radius $r_\mathrm{L} = 1.9$ mm. When calculating the numerical gradient $\partial_x z$ some problematic data-points was filtered out and the guiding center was calculated for 60 data-points in the simulation, the mean value was then found to be $(x_g c, z_g c) = (7.04 \cdot 10^{-9}, 1.34 \cdot 10^{-7})\ r_\mathrm{L}$, essentially a numerical 0, as it should be. Then by using a $\boldsymbol{B}$ field in a Cartesian coordinate system with a x^{-1} dependence a vertical z drift was expected, only due to the $\boldsymbol{\nabla} B$ drift, which the simulation also showed. The change along z, Δz, due to a $\boldsymbol{\nabla} B$ drift was calculated both analytical and estimated from the guiding center location in the simulation, these were $(\Delta z)_\mathrm{sim} = 0.354\ r_\mathrm{L}$ and $(\Delta z)_\mathrm{analytical} = v_{\boldsymbol{\nabla} B} M k = 0.359\ r_\mathrm{L}$. The deviation is expected to be due to the systematic errors that seemed to be occurring in the calculation of the numerical gradient, i.e. the simulation-code is probably working as it should. Another check that the simulation was working properly was to make the $\boldsymbol{E}$ and $\boldsymbol{v} \times \boldsymbol{B}$ terms equal and pointing in opposite directions, from which a straight line motion was observed. This therefore again confirmed that the simulation was working properly.

Being confident in the simulation, a method for interpolating an advanced $\boldsymbol{B}$ field was implemented in the code, along with a transformation from (x, y, z) to (R, Z, φ). The Equations of motion were kept in the basis of Cartesian coordinates, while the data for $\boldsymbol{B}$ was in cylindrical (R, Z, φ), so an intermediate step of interpolating $\boldsymbol{B}$ in cylindrical coordinates was needed. The current version of the code uses linear interpolation in (R, Z, φ), where any coordinate outside the grid will be made into a null-type. This is would make physical sense in the (R, Z) coordinates if the boundary is also e.g. the wall of the tokamak, but not in the φ direction. This is quite unfortunate, because the current data has a φ grid which goes between $[0.008, 6.29]$ rad, so there is a gap in which the particle will be converted into a null-type. This should be debugged in the future, but a quick fix was to initialize the particle away from the problematic coordinates and set the $v_\parallel$ very small avoiding the particle to reach to problematic null-type space. A trajectory of $M = 2000$ time points for the tungsten ion in the advanced 3-dimensional $\boldsymbol{B}$ field was shown, and it in general seemed to work properly. The trajectory looked much like that of the analytical $\boldsymbol{\nabla} B$ drift simulation in an analytical x^{-1} dependent field, which is nice because the largest component in the FELTOR data was the B_φ component which had a R^{-1} dependence. The diameter of the Larmor motion seemed to

match twice the calculated analytical expression of the Larmor radius calculated in the initial point, which is sensible. The Ion also followed a left handed motion, but for some reason the $\boldsymbol{\nabla} B$ drift was observed in the $\boldsymbol{e}_Z$ direction, when it should be in the $-\boldsymbol{e}_Z$ direction for a positively charged ion. This should be debugged in the future.

References

[1] A. Loarte, M. Reinke, A. Polevoi, M. Hosokawa, M. Chilenski, N. Howard, A. Hubbard, J. Hughes, J. Rice, J. Walk, *et al.*, *Tungsten impurity transport experiments in alcator c-mod to address high priority research and development for iter*. Physics of Plasmas **22**(5) (2015).

[2] Z. Wen, Y. Chen, and L. Zhang, *Modeling of tungsten impurity transport and distribution in east based on multi-fluid and kinetic monte carlo simulations*. AIP Advances **14**(5) (2024).

[3] S. Yamoto, Y. Homma, K. Hoshino, M. Toma, and A. Hatayama, *Impgyro: The full-orbit impurity transport code for sol/divertor and its successful application to tungsten impurities*. Computer Physics Communications **248**, 106979 (2020).

[4] P. Stangeby and D. Moulton, *A simple analytic model of impurity leakage from the divertor and accumulation in the main scrape-off layer*. Nuclear Fusion **60**(10), 106005 (2020).

[5] M. H. Holmes, *Introduction to numerical methods in differential equations* (Springer) (2007).

[6] D. Gross, W. Hauger, J. Schröder, W. A. Wall, and S. Govindjee, *Engineering Mechanics 3* (Springer) (2014).

[7] E. W. Weisstein, *Radius of curvature*. `https://mathworld.wolfram.com/RadiusofCurvature.html` [Accessed: 03.06.2024].