Python for physics assignment 2

Axel Wohlin

June 2022

UPPSALA
UNIVERSITET

1 Introduction

In this assignment it is showcased how to describe particle motion trajectories using python. We will be dealing with charged particles in magnetic fields, and therefore we will be using the Lorentz force equation

$$F = q(E + v \times B)$$

We will use this relationship to numerically solve the differential equation corresponding to the particles motion in a homogenous magnetic field. This will be done using the scipy $solve_ivp$ differential equation solver, which is an adaptive step length Runge-kutta based solver.

Q1

1.1 Problem Introduction

For this problem we have a proton moving in the x-direction exposed to a homogenous magnetic field in the z-direction. We will follow this proton for $1\mu s$ and plot its trajectory.

1.2 solution

Using the initial value problem solver from matplotlib, this was the solution I obtained from 1 micro second. As we can see it's only accelerated in the Y direction, which is just what we except since the only force is a cross product between the velocity in the x direction and a magnetic field in the Z direction. It is also expected for it to accelerate in the negative y direction as the cross product between the x and z unit vectors is the negative y unit vector.

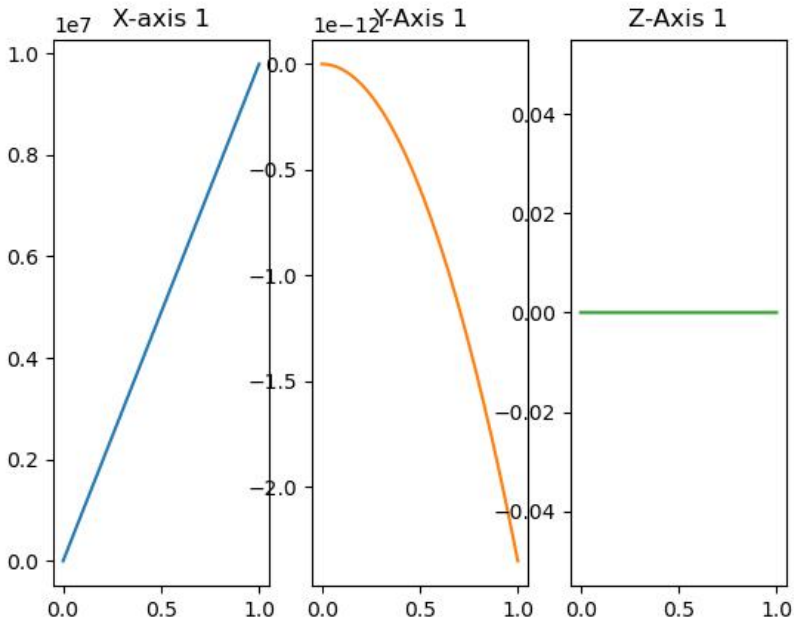

Figure 1:

Q2

1.3 Problem Introduction

For this problem we have a similar proton, but now also having an initial velocity in both the x and z-direction. We will follow this proton for $1\mu s$ and plot its trajectory.

1.4 solution

This question was solved in a similar fashion. We see here no real difference in how the proton was accelerated in the Y direction, this is because a cross product between the velocity's Z component and the magnetic field in the Z direction will be 0. So only the X velocity impacts it for this short time span. If we were to follow the proton for a longer timespan we would be able to see a difference in the x-trajectory as the cross product between $(1, -1, 1) \times (0, 0, 1)$ is $(-1, -1, 0)$ (not to scale just showcasing geometry). However the y-velocity isnt really changed much during only 1 microsecond, so it's not really noticeable in the x-plot.

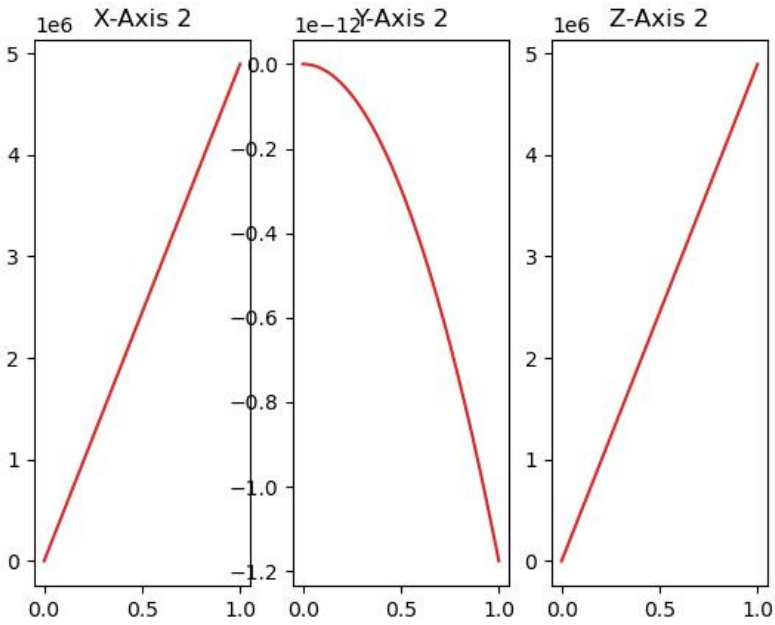

Figure 2: Figure 2

Q3

Below is a 3D figure created using Axes3D. We see that the proton is following the expected trajectory for being in such a strong (3 Tesla) magnetic field, with it starting in the xz direction, but quickly "bending" its path to the x,-y,z direction. If we were to follow it longer it would also make the famous Helical-path shape of the Lorentz force due to the y-velocity cross product with the z-direction B-field resulting in an x-acceleration.

[

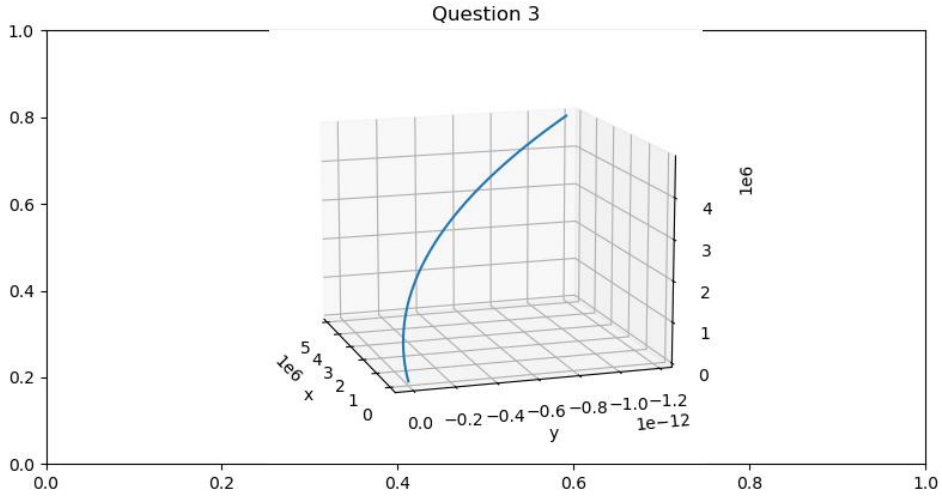

Figure 3: Figure 3

2 Python Code

```
#Axel Wohlin's assignment 2

import numpy as np
import math
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
import scipy.constants as const
from mpl_toolkits.mplot3d import Axes3D

#Since the proton has kinetic energy 1MeV we know E = mv^2/2
#So its velocity will be equal to sqrt(2E/m)


times = (0, 1)
time_span = np.linspace(0, 1, 1000000)
#Setup initial position and velocity vector. We use const.eV to get SI units
XV = np.array([0.0, 0.0, 0.0, math.sqrt(1e6*const.electron_volt/const.proton_mass), 0.0 ,0.0


def PosVel_to_VelAcc(t, y):
    #F = q*V cross B and a = F/m, so we can find the acceleration now.
    z = np.zeros(6)           # Initialise the vector to return
    z[:3] = y[3:]
    z[3:] = const.elementary_charge*np.cross(z[:3],np.array([0.0,0.0,3.0]))
    +np.array([0.0,9.81*const.proton_mass,0.0])
    return z

results_1 = solve_ivp(PosVel_to_VelAcc, times, XV, t_eval=time_span, rtol=1e-10, atol=1e-10)

#I understood question 2 as them having total 1MeV kinetic energy so it's half in x and half
XV_2 = np.array([0.0, 0.0, 0.0, 0.5*math.sqrt(1e6*const.electron_volt/const.proton_mass), 0.
results_2 = solve_ivp(PosVel_to_VelAcc, times, XV_2, t_eval=time_span, rtol=1e-10, atol=1e-1

#Plots and save figures for report.

fig, axs = plt.subplots(1, 3)
axs[0].plot(results_1.t, results_1.y[0])
axs[0].set_title('X-axis 1')
axs[1].plot(results_1.t, results_1.y[1], 'tab:orange')
axs[1].set_title('Y-Axis 1')
axs[2].plot(results_1.t, results_1.y[2], 'tab:green')
axs[2].set_title('Z-Axis 1')
plt.savefig("assignment_2_fig_1.jpg")
```

```
fig2, axes = plt.subplots(1,3)
axes[0].plot(results_1.t, results_2.y[0], 'tab:red')
axes[0].set_title('X-Axis 2')
axes[1].plot(results_1.t, results_2.y[1], 'tab:red')
axes[1].set_title('Y-Axis 2')
axes[2].plot(results_1.t, results_2.y[2], 'tab:red')
axes[2].set_title('Z-Axis 2')
plt.savefig("assignment_2_fig_2.jpg")


fig3 = plt.figure(3)
plt.title("Question 3")
ax = fig3.add_subplot(projection='3d')
ax.plot3D(results_2.y[0], results_2.y[1], results_2.y[2], color='C0')
plt.savefig("assignment_2_fig_3.jpg")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```