Deliverable 1 Prompt: Theory for Particle in a Constant Magnetic Field

Schmitt Independent Study: Winter 2020

Cole Kampa

January 26, 2020

Problem 1: Calculate Track Parameter Theory (function)

Write a Python function that calculates features of a helical trajectory in a constant magnetic field, based on theory. The function should be of the following form:

```python
def B_theory(phi_0, theta_0, v_0, B_0):
    ...
    ...
    ...
    return R, pitch, T_circle, v_z, v_transverse
```

The function's parameters are ϕ_0 and θ_0 (angles of initial velocity vector in spherical coordinates), v_0 (initial velocity), and B_0 (strength of magnetic field). We will set the orientation of the field in the $\hat{z}$ direction: $\vec{B} = B_0\hat{z}$.

The function should return the following particle trajectory (or track) parameters: helix radius R, helix *pitch*, period of one helix revolution T_{circle}, magnitude of velocity in the direction of the magnetic field v_z, and magnitude of velocity perpendicular to the magnetic field $v_{transverse}$.

Comment on the units of your calculation.

Later on we will put this function in a standalone script that can be imported into any other code. With this we will be able to check if our trajectory code is behaving as we expect.

Problem 2: Random Sampling for Expectations

Using the `numpy` package, generate $N = 100,000$ initial conditions for the trajectory. Each trial should generate ϕ_0 and θ_0.

ϕ_0 should be sampled from a uniform (flat) probability distribution function from 0 to 2π. $\cos(\theta_0)$ should be sampled from a uniform probability distribution function from -1 to 1. This sampling configuration ensures uniform solid angle coverage. Comment on why this is true (hint: consider the differential of solid angle in spherical coordinates https://en.wikipedia.org/wiki/Solid_angle).

So, you should have two `numpy.array` objects `phi_0s` and `theta_0s` which both have length N.

Using your **B_theory** function, calculate track parameters for your N randomly generated initial conditions. Try passing your `phi_0s` and `theta_0s` arrays directly into the function. Does this result in the desired behavior? If not, adjust your function to make this work properly.

The result here should be the following length N `numpy.array` objects: `Rs`, `pitches`, `Ts_circle`, `vs_z`, `vs_transverse`.

Problem 3: Plotting Results

Be sure to properly format each plot (e.g. axis labels, legends, etc.).

0.1 Distributions (Histograms)

Using the input arrays and result arrays from Problem 2, make a histogram of each array. Along with each plot, print out the sample mean and sample standard deviation of the array.

0.2 Result vs. Input (Scatter/Line plots)

Calculate one more set of result arrays using the input `phi_0`$= 0$ and `theta_0` evenly spaced between 0 and π. All other input parameters should be the same ones used before. You should decide on what N is appropriate (hint: try $N = 10, 25, 50, 100$ and see how the plot changes).

For each result array, make a line plot *result* vs. `theta_0` to see relationships between the track parameters and the initial conditions.

0.3 Varying Other Inputs

Repeat part 0.2 but instead of varying `theta_0`, vary the other input parameters and make the corresponding scatter plots.

Note: vary only one input at a time. You should have two additional sets of plots: varying `v_0` and varying `B_0`.

Pick one or two plots from part 0.2/0.3 and explain why this is the expected shape from the theory.