Magnetic Mirror Effect in Magnetron Plasma:

Modeling of Plasma Parameters

January 26, 2022

GitHub repository:

https://github.com/18BME2104/MagneticMirror

Required functionalities and Files

1. Particle - **particle.ipynb**
2. Electric and Magnetic fields - **field.ipynb**
3. Particle initialization - **sampling.ipynb**
4. Updating the particles - **step.ipynb**
5. Batches of updates - **run.ipynb**
6. Plotting - **plot.ipynb**

1. Particle - particle.ipynb

Define state of a particle and update it

1. Particle State:
   Position and Velocity
   name, mass, charge

2. Update State:
   Boris Update
   … other updates strategies

```python
class Particle:
    #... other things

    def Boris_update(self, afield, argsE, argsB):
        # Define q_prime
        q_prime = (self.charge / self.mass) * (dt /
            2)

        # Get E and B fields from the afield
            argument by passing in the current
            position of the particle
        argsE = V, center
        E = afield.get_E_field(self.r, V, center)
        argsB = n, I, R, B_hat, mu_0
        B = afield.get_B_field(self.r, n, I, R,
            B_hat, mu_0)
```

```
1            #Boris velocity update
2            v_minus = self.v + q_prime * E
3            v_plus = v_minus + q_prime * 2 * np.cross(
                 v_minus, B)
4            v_new = v_plus + q_prime * E
5
6            self.v = v_new
7
8
9            #could have also done:
10           #self.v += (2 * q_prime) * (E + np.cross( (
                 self.v + q_prime * E), B))
11
12           #update position
13           self.r += v_new * dt
14
15       #... some other things
```

2. Electric and Magnetic fields - field.ipynb

Define field configurations

- Electric Field:
    1. Uniform Electric Field
    2. Radial Electric Field
- Magnetic Field:
    1. Uniform Magnetic Field
    2. Magnetic Field due to Helmholtz Coil
    3. Magnetic Field due to 2 Helmholtz Coils

... other field configurations:

1. Analytic field expression
2. Apparatus like coils

```python
1   class Field:
2       #... something
3
4       def uniform_E_field(self, E):
5           return E
6
7       def radial_E_field(self, r, V, center =
            [0,0,0]):
8           #Get the distance vector of the particle
                from the electrode
9           dr = [(r[0] - center[0]), (r[1] - center
                [1]), 0]
10          #Get the electric field
11          E = V * dr
12          return E
```

```python
1   class Field:
2       def uniform_B_field(self, B):
3           return B
4
5       def helmholtz_coil_B_field(self, n, I, R, B_hat
            , mu_0):
6           return ( (4/5)**1.5 * ( (mu_0 * n * I) / (R
                ) ) * B_hat)
7
8       def two_helmholtz_B_field(self, n1, I1, R1,
            B1_hat, n2, I2, R2, B2_hat, mu_0):
9           B1 = helmholtz(self, n1, I1, R1, mu_0,
                B1_hat)
10          B2 = helmholtz(self, n2, I2, R2, mu_0,
                B2_hat)
11          B_hat = B1_hat + B2_hat
12          return B_hat
```

3. Particle initialization - sampling.ipynb

Sample initial positions and velocities
- Initial Position Samping:
    1. Same position for all: eg: valve position
    2. Same distance from a point: eg: from an electrode
       ... other distributions
- Initial Velocity Sampling:
    1. Same velocity for all: eg: injected by pump
    2. Same speed for all
    3. Speeds based on same K.E. :
       eg: accelerated through same potential
    4. Maxwellian distributed speeds
    5. Maxwellian distributed velocities
    6. Parabolic distribution
       ... other distributions
- Writing to files
    csv format

```python
class Sampler:
    #... something

    def sample_same_given_position(self, r, n):
        positions = []
        for i in range(n):
            positions.append(r)

        return np.array(positions)
    def sample_same_given_speed_all_random_direction(self, s, n):
        velocities = []
        for i in range(n):
            velocities.append(s *
                uniform_random_unit_vector())
        return np.array(velocities)
```

```python
1   class Sampler:
2       #... something
3       def
            sample_Maxwellian_velocity_all_random_direction
            (self, v_median, K, T, m, n):
4           speeds = sample_Maxwellian_speed(self,
               v_median, K, T, m, n)
5           velocities = []
6           for i in range(n):
7               velocities.append(speeds[i] * np.array(
                   uniform_random_unit_vector()))
8
9           return np.array(velocities)
10
11      def sample_parabolic_velocity(self):
12          pass
```

```
 1    class Sampler:
 2        #... something
 3        def write_to_csv_file(self, ...):
 4
 5            # write array to csv file
 6
 7        def write_file_name(self, ...):
 8
 9            # write the file name to the list of
                 available files
10
11        #... something else
```

4. Updating the particles - step.ipynb

Evolve particles under influence of fields

1. Initialize particles: hold states
   - Use initial positions and velocities directly
   - Read initial positions and velocities from saved files
2. Initialize fields: call field configurations
3. Update particles: call updates

```python
1  class Step:
2
3      # ... something
4
5  ### PARTICELS INTIALIZATION SECTION
6      def initialize_particles(self, names, q_s, m_s,
           r_0_s, v_0_s, a_0_s, n):
7
8          for i in range(n):
9              self.particles.append(Particle(names[i
                  ], q_s[i], m_s[i], r_0_s[i], v_0_s[
                  i], a_0_s[i] ))
10
11  ### FIELDS INITIALIZATION SECTION
12      def initialize_fields(self):
13          self.fields = Field()
```

```python
### TIME STEP SECTION
    def update_particles(self, dt, argsE, argsB):

        for particle in self.particles:
            particle.update(self.fields, dt, argsE,
                argsB)
```

```python
class Step:

    # ... something
    def read_r_or_v_file(self, index):
        # ...
        return rows

    def reshaper(self, array_from_file):
        # ...
        return reshaped_array

    def read_r_or_v_file_and_reshape(self, index):
        # ...
        return reshaped_array

    # ... something else
```

5. Batches of updates - run.ipynb

Define dynamic plasma system

1. Create particles
2. Update particles
3. Update fields
4. Create new particles
5. Remove particles

```python
class Run:
    def create_particles(self):
        pass

    def update_particles(self):
        pass

    def remove_particles(self):
        # This might include particles moving
            outside the chamber (the Electric and
            Magnetic fields or
        # simply positions of interest) or
            particles being absorbed for example in
             a coating process
        pass
```

```python
1    def create_fields(self):
2        pass
3
4    def change_fields(self):
5        pass
6
7    #... something else
```

6. Plotting - plot.ipynb

Plot functionalities

References

Qin, H., Zhang, S., Xiao, J., & Tang, W. M. (April, 2013). *Why is Boris algorithm so good?*. Princeton Plasma Physics Laboratory, PPPL-4872.