PHYS2016 – ELECTROMAGNETISM HPO REPORT

V VIJENDRAN

AUSTRALIAN NATIONAL UNIVERSITY

This paper serves as a design document to explain the design philosophies and choices for the implementations for this HPO assignment. This HPO aims to visualize and plot electric and magnetic fields using Python. For the visualization of electromagnetic fields, a python library named `EMPY - Electromagnetic Python` was developed. `EMPY` can produce both 2D and 3D plots of static electric fields and 2D plot of static magnetic fields of any defined system.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

This HPO assignment aims to plot vector and scalar fields using Python to describe electromagnetic phenomena. To visualize and simulate these scenarios, a python library `EMPY - Electromagnetic Python` was developed to allow for a range of possible simulations..

Using the `EMPY` library, the following configurations were simulated:

- Electric Field Lines and Potentials for two point charges q_1 and q_2 such that
  - $q_1 = q_2 = +q$
  - $q_1 = 4q_2$
  - $q_1 = -q$
- Electric Field Lines(2D) and Potential for a Straight Wire
- Electric Field Lines(2D) and Potential for a Parallel Wire
- Electric Field Lines(2D) and Potential for a Circular Loop
- Electric Field Lines(2D) and Potential for a Square Plate
- Electric Field Lines(3D) of a Point Charge
- Electric Field Lines(3D) of a Dipole
- Electric Field Lines(3D) of an Infinitesimal Line Charge
- Electric Field Lines(3D) of an Infinitesimal Square Area
- Electric Field Lines(3D) of a Parallel Plate Capacitor
- Electric Field Lines(3D) of a Spring
- Electric Field Lines(3D) of a Circular Loop
- Magnetic Field Lines(2D) of a Straight Wire
- Magnetic Field Lines(2D) of a Circular Loop
- Magnetic Field Lines(2D) of Helmholtz Coils
- Magnetic Field Lines(2D) of Maxwell Coils
- Magnetic Field Lines(3D) of a Circular Loop

In the following sections, the design philosophies, choices and the implementations for the above configurations will be discussed thoroughly.

2 EMPY – ELECTROMAGNETIC PYTHON

Before discussing the simulations and visualizations for various configurations, it is essential to discuss the workings of the python library `EMPY - Electromagnetic Python` with which the simulations were produced. Instead of explaining the library in its entirety, I describe the general structure and properties and only explain specific algorithms that are crucial to the working of the library.

The python library `EMPY - Electromagnetic Python` is divided into two main parts which are as follows.

1. **Electrostatics**: This portion of the library contains the classes, objects, and functions that allow to instantiate charges and other electrically charged systems and enables us to plotParallel Wire the electric fields and potential in both 2D and 3D.
2. **Magnetostatics**: This portion of the library contains the classes, objects, and functions that allow to instantiate solid magnetic objects and other helper methods and enables us to visualize the system in 3D and plot the magnetic field lines 2D.

2.1 Electrostatics

The Electrostatics portion of the `EMPY` library consists of 4 python files.

1. `Charge.py`
2. `Charge3D.py`
3. `System.py`
4. `System3D.py`

The files `Charge.py` and `Charge3D.py` are similar to each other, with the only exception that `Charge3D.py` allows to calculate and simulate for an additional Z dimension. In `Charge.py`, the `Mathplotlib` package was used to generate the 2D plots of electric field lines and 3D plots of potential. For `Charge3D.py`, the `Mayavi` package was used to generate the 3D plots of electric field lines. This is also the case for the python files, `System.py` and `System3D.py`.

In the following subsections, I will explain the workings and algorithms for the above python files.

2.1.1 *Charge*

The `Charge.py` contains the Charge class that allows us to instantiate a point charge of a given magnitude at specified positions. It contains the following methods that allow calculating the electric fields, potential, and also to plot them.

1. **Electric Field:** The `electricField` algorithm takes in two parameters, the charge itself, and the position at which the electric field is to be calculated.

   To determine the position of the electric field relative to the point charge, we use the concept of separation vector. The separation vector specifies the distance and direction from the source to the point where we want to calculate the field.

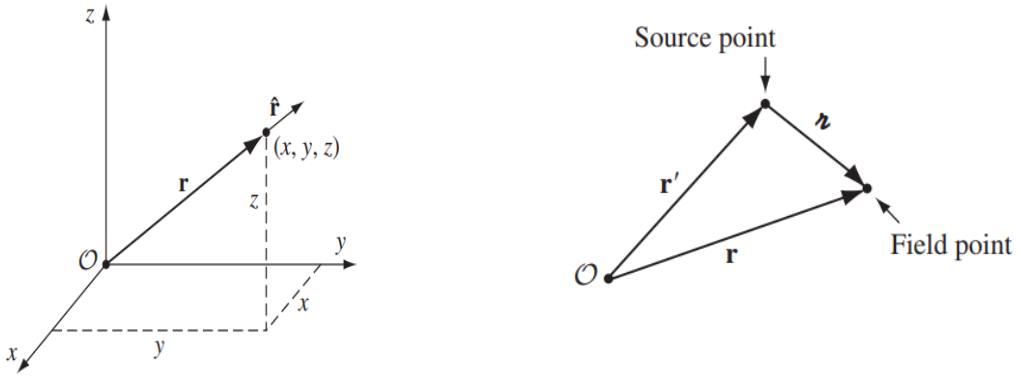


Figure 1: Position, Displacement and Separation Vectors

Given a position vector $\mathbf{r}$,

$$\mathbf{r} \equiv x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}} \tag{1}$$

and a source point, $\mathbf{r}'$

$$\mathbf{r} \equiv x'\hat{\mathbf{x}} + y'\hat{\mathbf{y}} + z'\hat{\mathbf{z}} \tag{2}$$

the separation vector can be calculated as follows:

$$\mathbf{R} \equiv \mathbf{r} - \mathbf{r}' \tag{3}$$

which can be written in Cartesian coordinates as,

$$\mathbf{R} = (x - x')\,\hat{\mathbf{x}} + (y - y')\,\hat{\mathbf{y}} + (z - z')\,\hat{\mathbf{z}} \tag{4}$$

With the charges q and separation vector $\mathbf{R}$, the electric field can be calculated as

$$\mathbf{E}(\mathbf{r}) \equiv \frac{1}{4\pi\epsilon_0}\frac{q}{R^2}\hat{\mathbf{R}} \tag{5}$$

where

$$R = \sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2} \tag{6}$$

$$\hat{\mathbf{R}} = \frac{(x - x')\,\hat{\mathbf{x}} + (y - y')\,\hat{\mathbf{y}} + (z - z')\,\hat{\mathbf{z}}}{\sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2}} \tag{7}$$

The following algorithm computes the electric field of a charged point object using the above steps.

Algorithm 1 Electric Field

```
procedure electricField(self, fieldPos)
    r := sqrt((fieldPos[0] - self.pos[0])^2 + (fieldPos[1] - self.pos[1])^2)
    r[r < 0.005] := 0.005
    E_x = self.q × (fieldPos[0] - self.pos[0]) ÷ r^3
    E_y = self.q × (fieldPos[1] - self.pos[1]) ÷ r^3
    E_z = self.q × (fieldPos[2] - self.pos[2]) ÷ r^3
    return E_x, E_y, E_z
```

2. **Potential Field:** We know that the electric field can be written as the gradient of a scalar potential.

$$\mathbf{E} = -\nabla V \tag{8}$$

From this expression, by setting the reference point at infinity, the potential of a point charge can be derived as,

$$V(\mathbf{r}) = \frac{1}{4\pi\epsilon_0}\frac{q}{R} \tag{9}$$

The following algorithm computes the electric potential using the above steps.

Algorithm 2 Electric Potential

```
procedure electricPotential(self, fieldPos)
    r := sqrt((fieldPos[0] - self.pos[0])^2 + (fieldPos[1] - self.pos[1])^2)
    r[r < 0.005] := 0.005
    V = self.q/r
    return V
```

3. **Plotting Electric Potential:** The `electricPotential` algorithm computes and returns the electric potential at a specified location. To get the electric potential on all the points of a two-dimensional surface, we instantiate a 2D mesh-grid and compute the electric potential on all the points. This primarily produces a contour plot of the electric potential.

   A better way to visualize this electronic terrain and to build physical intuition is to construct a 3D surface plot. This is done by creating a mapping from the x, y mesh of electric potential to the z dimension. I have chosen the mapping to be as

$$V_{i,j} = \begin{cases} -V_{i,j}^{1/9} & V_{i,j} < 0 \\ V_{i,j}^{1/9} & V_{i,j} > 0 \end{cases} \tag{10}$$

2.1.2 *System*

The file `system.py` contains the `System` class. This class is used to keep track and maintain a collection of `Charge` objects. The `System` class also contains functions similar to that of the `Charge` class; however, the only distinction is that it computes the sum of all-electric field and potential of all the charges present.

When dealing with a collection of charges, we have two types of charge distribution: discrete and continuous charge distribution. The `System.py` also contain two additional classes namely, `DiscreteSystem` and `ContinuousSystem`. These two classes are children of the `System` class, i.e. inherit the same properties of the parent class but also have additional properties and characteristics of their own. The reason for the development of `ContinuousSystem` class was to define charged systems using vector equations rather than adding point charges one by one.

2.2 Magnetostatics

The magnetostatics portion of the `EMPY` library is much smaller than that of the electrostatics due to the complexity of the expressions. Since the expression of describing the magnetic field of a system varied concerning the system's configuration, I decided to implement the functionality for two simple base cases; namely, a straight current-carrying conductor described by the `Wire` class and circular loop described by `Loop` and `Loops` classes. By creating multiple instances of the circular loop, I was able to plot the visualisation of the magnetic fields generated by Helmholtz and Maxwell coils.

2.2.1 *Wire*

The `Wire` class of `Magnetostatics` allows one to instantiate a straight current carry wire of the desired radius placed at a certain position and plot the visualisation of its magnetic field. The expression for the magnetic field generated by a straight current carry conductor is as follows.

Consider a straight conductor of length `l` along the zaxis carrying current I in the $\hat{z}$ direction. We desire the $\mathbf{B}$ in the $x-y$ plane at radial distance r.

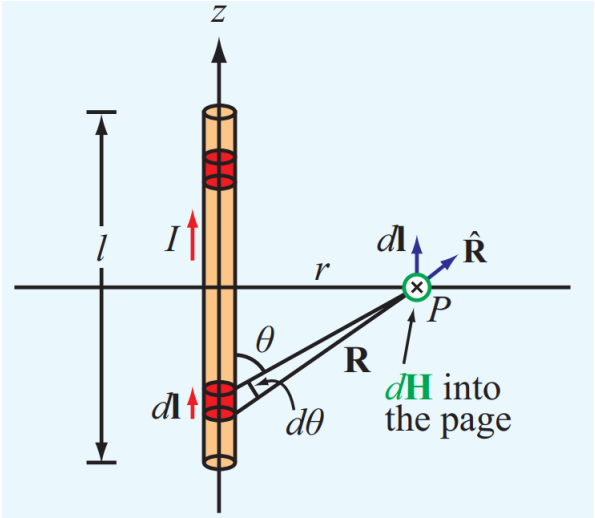


Figure 2: Finite length linear conductor carrying current I

Writing out $\mathbf{R}$, $\hat{\mathbf{R}}$, and $d\mathbf{l} \times \hat{\mathbf{R}}$ we have

$$\begin{aligned} \mathbf{R} &= (\hat{\mathbf{r}}r) - (\hat{\mathbf{z}}z) \\ \hat{\mathbf{R}} &= \frac{\hat{\mathbf{r}}r - \hat{\mathbf{z}}z}{\sqrt{r^2 + z^2}} \\ d\mathbf{l} \times \hat{\mathbf{R}} &= \hat{\mathbf{z}}dz \times \left(\frac{\hat{\mathbf{r}}r - \hat{\mathbf{z}}z}{\sqrt{r^2 + z^2}} \right) = \frac{\hat{\boldsymbol{\phi}}r dz}{\sqrt{r^2 + z^2}} \end{aligned} \tag{11}$$

since $\hat{\mathbf{z}} \times \hat{\mathbf{r}} = \hat{\boldsymbol{\phi}}$.

We are now ready to integrate over $-l/2 \leqslant z \leqslant l/2$

$$\begin{aligned} \mathbf{H} &= \frac{I}{4\pi} \int_l \frac{d\mathbf{l} \times \hat{\mathbf{R}}}{R^2} = \hat{\boldsymbol{\phi}} \frac{I}{4\pi} r \int_{-l/2}^{l/2} \frac{dz}{\left(r^2 + z^2\right)^{3/2}} \\ &= \hat{\boldsymbol{\phi}} \frac{Il}{2\pi r \sqrt{4r^2 + l^2}} \end{aligned} \tag{12}$$

Converting to magnetic flux, we get

$$\mathbf{B} = \mu_0 \mathbf{H} = \hat{\boldsymbol{\phi}} \frac{\mu_0 Il}{2\pi r \sqrt{4r^2 + l^2}} \tag{13}$$

When $l >> r$ the wire can be treated as infinite in length, thus giving the important result that

$$\mathbf{B} = \hat{\boldsymbol{\phi}} \frac{\mu_0 I}{2\pi r} \quad \text{(T) (for infinite length wire)} \tag{14}$$

This result parallels the infinite line charge by having the field inversely proportional to distance. The field, however, forms concentric circles around the wire.

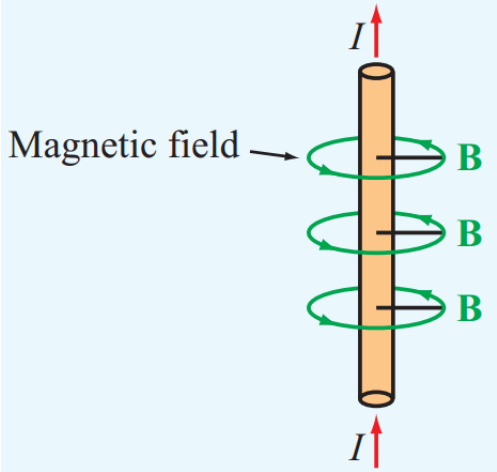


Figure 3: The field of a long linear conductor encircles the conductor and falls off as $\frac{1}{r}$

The magnetic field of the straight current-carrying conductor is calculated using the following algorithm.

Algorithm 3 Magnetic Field

procedure CALCULATEB(x, y)
    $\mu := 1.26 \times 10^{-6}$
    $mag := \frac{\mu}{2\pi} \times \frac{i}{\sqrt{x^2+y^2}}$
    $B_x := mag \times ((\arctan(\frac{y}{x})))$
    $B_y := mag \times (-\sin(\arctan(\frac{y}{x})))$
    return B_x, B_y

2.2.2 *Loops*

The file `Loops.py` consist of two classes: `Loop` and `LoopSystem`. The `Loop` class allows the instantiation of circular current carrying conductor, whereas `LoopSystem` maintains the collection of Loop objects, and evaluates the net magnetic field of the system.

The magnetic field of a current-carrying conductor is calculated as follows. Consider a circular loop of radius R in the xy plane carries a steady current I, as shown in the figure below. The field point is $P = (0, 0, z)$

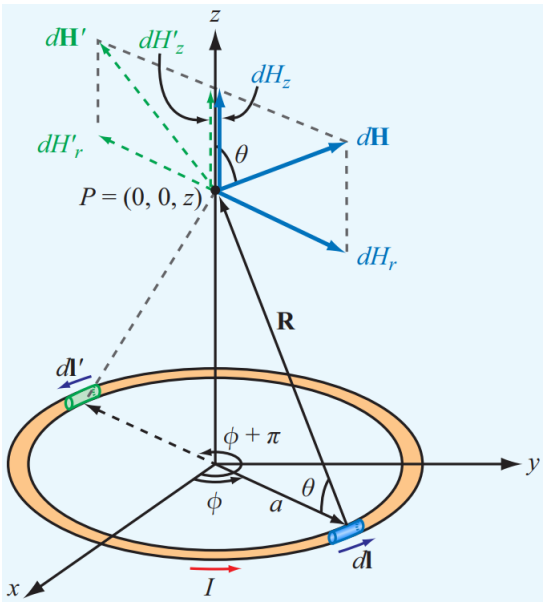


Figure 4: The field of a circular conductor encircles the conductor and falls off $\frac{1}{r}$

Writing out $\mathbf{R}$, $\hat{\mathbf{R}}$, $d\mathbf{l} \times \hat{\mathbf{R}}$, and $d\mathbf{H}$ we have

$$\begin{aligned} \mathbf{R} &= (\hat{\mathbf{z}}z) - (\hat{\mathbf{r}}a) \\ \hat{\mathbf{R}} &= \frac{-\hat{\mathbf{r}}a + \hat{\mathbf{z}}z}{\sqrt{r^2 + z^2}} \\ d\mathbf{l} \times \hat{\mathbf{R}} &= \hat{\phi}a d\phi \times \left(\frac{-\hat{\mathbf{r}}a + \hat{\mathbf{z}}z}{\sqrt{r^2 + z^2}} \right) = a d\phi \frac{\hat{\mathbf{x}}a + \hat{\mathbf{r}}z}{\sqrt{r^2 + z^2}} \\ d\mathbf{H} &= \frac{Ia}{4\pi R^2} \frac{\hat{\mathbf{z}}a + \hat{\mathbf{r}}z}{\sqrt{a^2 + z^2}} d\phi \end{aligned} \tag{15}$$

since $\hat{\boldsymbol{\phi}} \times -\hat{\mathbf{r}} = \hat{\mathbf{z}}$ and $\hat{\boldsymbol{\phi}} \times \hat{\mathbf{z}} = \hat{\mathbf{r}}$.

Finally, from Figure 4 we see that symmetry makes the radial component $H_r = 0$, and integrating over $0 \leqslant \phi \leqslant 2\pi$ gives

$$\begin{aligned} \mathbf{H} &= \hat{\mathbf{z}} \frac{Ia^2}{4\pi \left(a^2 + z^2\right)^{3/2}} \int_0^{2\pi} d\phi \\ &= \hat{\mathbf{z}} \frac{Ia^2}{2 \left(a^2 + z^2\right)^{3/2}} \quad (\mathrm{A/m}) \end{aligned} \tag{16}$$

The magnetic field of a circular current carrying conductor is calculated using the following algorithm.

Algorithm 4 Magnetic Field

```
procedure CALCULATEB(x, y)
    μ := 1.26 × 10^−6
    mag := (μ / 2π) × (i / sqrt(x^2 + y^2))
    B_x := mag × ((arctan(y / x)))
    B_y := mag × (−sin(arctan(y / x)))
    return B_x, B_y
```

3 2D ELECTRIC FIELD AND POTENTIAL VISUALISATION

This section discusses the implementation and visualization of 2 dimensional electrically charged objects by making use of the EMPY library.

Electric field lines provide a convenient graphical representation of the electric field in space. The properties of electric field lines may be summarized as follows:

- The direction of the electric field vector $\vec{\mathbf{E}}$ at a point is tangent to the field lines.
- The number of lines per unit area through a surface perpendicular to the line is devised to be proportional to the magnitude of the electric field in a given region.
- The field lines must begin on positive charges (or at infinity) and then terminate on negative charges (or at infinity).
- The number of lines that originate from a positive charge or terminating on a negative charge must be proportional to the magnitude of the charge.
- No two field lines can cross each other; otherwise the field would be pointing in two different directions at the same point.

3.1 Two Point Charge System

The electric field lines and potentials for two point charges q_1 and q_2 were visualised for the following configurations.

1. $q_1 = q_2 = +q$

   CODE

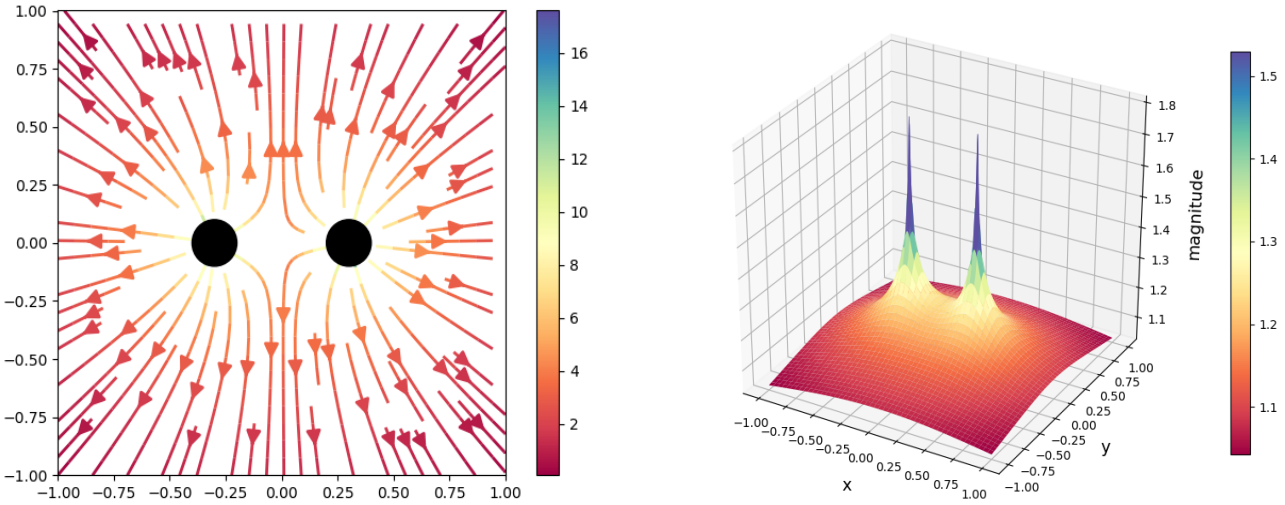
```
from EMPY.Electrostatics.Charge import Charge
from EMPY.Electrostatics.System import DiscreteSystem

Q = 1.6e-19
xs = ys = [-1, 1]

#System 1 - Q1 = Q2 = +q
c1 = Charge(1, [-0.3, 0])
c2 = Charge(1, [0.3, 0])

chargeDist1 = DiscreteSystem()
chargeDist1.add_Charge(c1)
chargeDist1.add_Charge(c2)
chargeDist1.plot_VectField(xs=xs, ys=ys,showEField=True, showEPot=False)
chargeDist1.plotPotential3D(xs,ys,6)
```

   OUTPUT

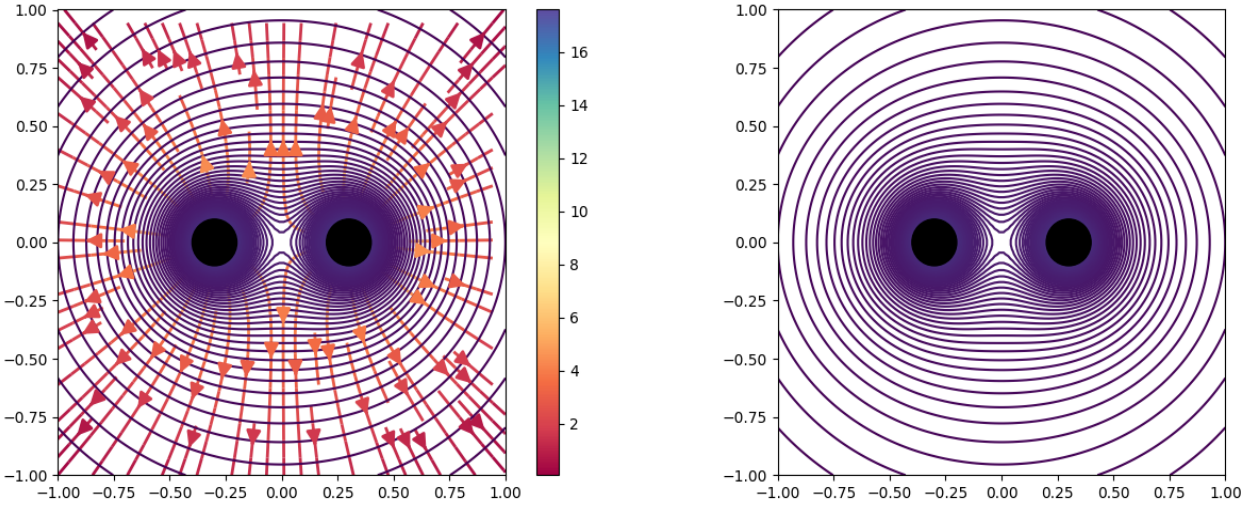


(a) Electric Field Lines

(b) Mapping of Electric Potential

Figure 5: Electric Field Lines and Potential Mapping of $q_1 = q_2 = +q$

We can also plot the electric potential along with the electric field lines. This can be done by setting the `showEPot` boolean value as True. By setting `showEPot` to True, we get the following plot.



(a) Electric Field and Potential

(b) Electric Potential

Figure 6: Electric Field Lines and Potential Mapping of $q_1 = q_2 = +q$

Figure 6a shows the pattern associated with two positive point charges and reveals that there is an absence of lines in the region between the charges. The absence of lines indicates that the electric field is relatively weak between the charges. The electric field lines are also curved in the vicinity of two identical charges. If the charges were both negative, the directions of the lines would be reversed. The colour of the field lines indicate the magnitude of the electric field; the electric field is stronger when closer to the charge and decreases as it gets farther away.

An electric potential is the amount of work needed to move a unit of charge from a reference point

to a specific point inside the field without producing an acceleration. Figure 5b shows the 3D mapping of the electric potential for the same system. The spikes at the position of the objects indicate the region of high potential, and it decreases rapidly as it moves farther away.

Figure 6b shows the electric potential as a two-dimensional contour plot. The density of the lines indicates the strength of the electric potential: the more dense, higher is the potential. Figure 6a is simply the electric field lines superimposed on top of the electric potential.

2. $q_1 = 4q_2$

CODE

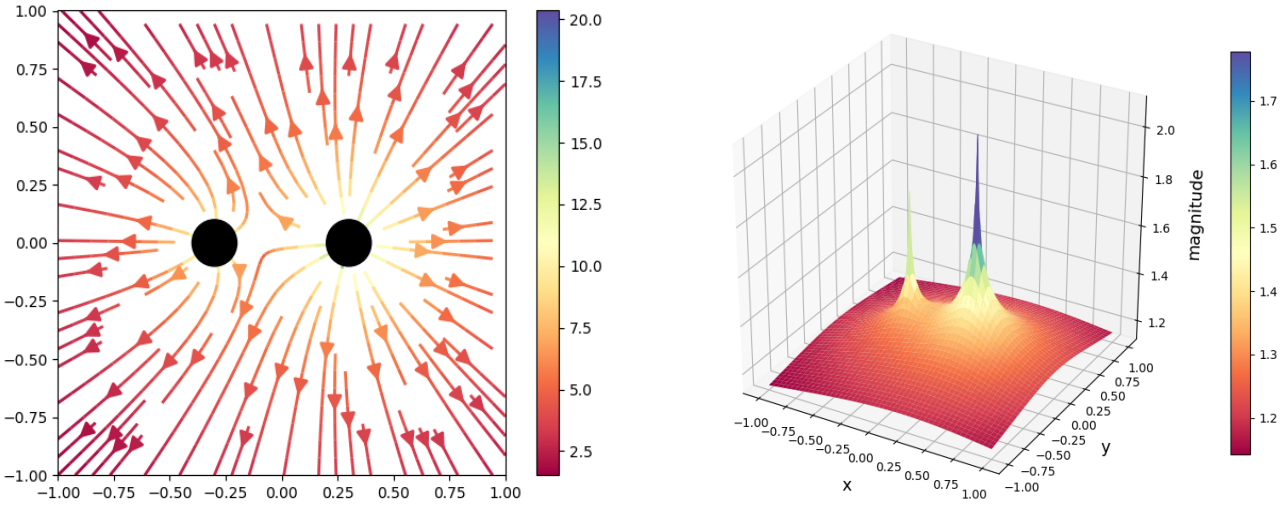
```
from EMPY.Electrostatics.Charge import Charge
from EMPY.Electrostatics.System import DiscreteSystem

Q = 1.6e-19
xs = ys = [-1, 1]

#System 2 - Q1 = 4Q2
c1 = Charge(1, [-0.3, 0])
c2 = Charge(4, [0.3, 0])

chargeDist1 = DiscreteSystem()
chargeDist1.add_Charge(c1)
chargeDist1.add_Charge(c2)
chargeDist1.plot_VectField(xs=xs, ys=ys,showEField=True, showEPot=True)
chargeDist1.plotPotential3D(xs,ys,6)
```

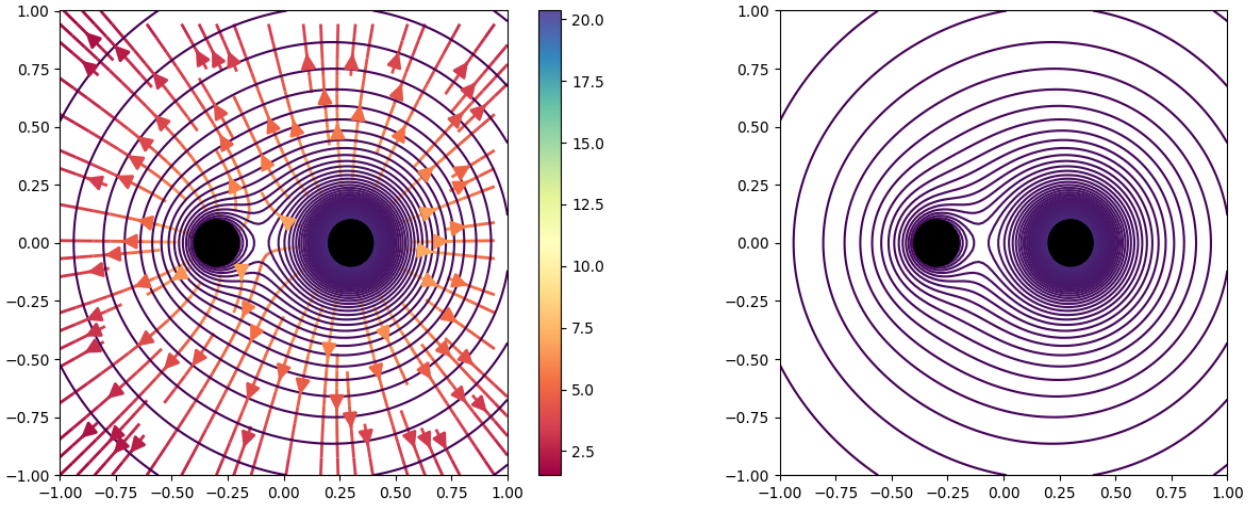
OUTPUT



(a) Electric Field Lines

(b) Mapping of Electric Potential

Figure 7: Electric Field Lines and Potential Mapping of $q_1 = 4q_2$



(a) Electric Field and Potential

(b) Electric Potential

Figure 8: Electric Field Lines and Potential Mapping of $q_1 = 4q_2$

The above figures 7 and 8 show the electric field and potential for $q_1 = 4q_2$. Since q_2 has four times the charge of q_1, it should have more number of electric field lines radially pointing outward and a much stronger electric potential. In figure 7a, the second charge has more number of electric field lines, and those of the second one deflect the field lines of the first charge. In fig 7b, the obvious difference in potential is visible by the difference in the height of the two spikes; this is also the case in figure 8b where the second charge has a much higher density of lines compared to the first one.

3. $q_1 = -q$

CODE

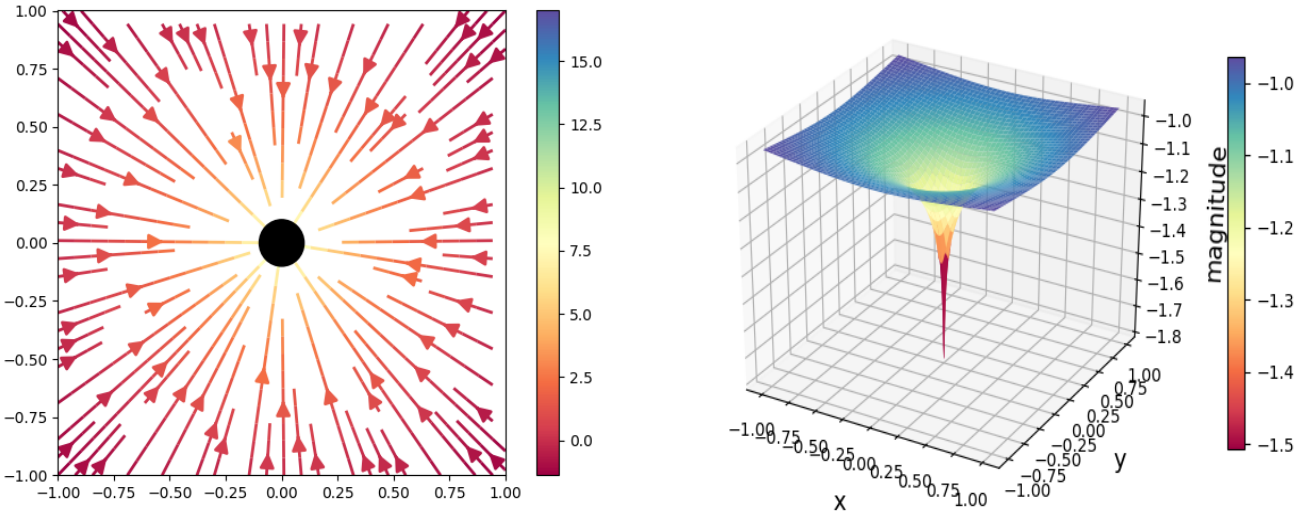
```
from EMPY.Electrostatics.Charge import Charge

Q = 1.6e-19
xs = ys = [-1, 1]

#System 3 - Q1 = -q
pos = [0,0]
c = Charge(-1, pos)

c.plot_VectField(xs=xs, ys=ys,showEField=True, showEPot=True)
c.plotPotential3D([-1, 1],[-1,1],6)
```

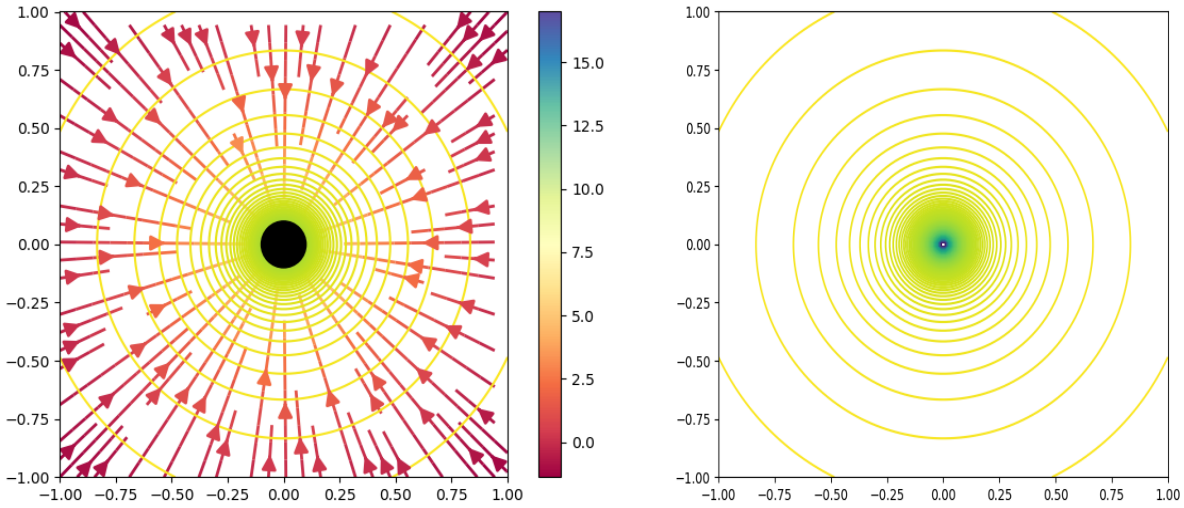
OUTPUT



(a) Electric Field Lines

(b) Mapping of Electric Potential

Figure 9: Electric Field Lines and Potential Mapping of $q_1 = -q$



(a) Electric Field Lines

(b) Electric Potential

Figure 10: Electric Field Lines and Potential of $q_1 = -q$

Figures 9 and 10 show the electric field lines and potential for an object with a negative unit charge. Notice that in figure 9a, the direction of the electric field is radially inward rather than outward. Similarly, in figure 9b, the height of the spike is in the negative direction.

3.2 Straight Wire

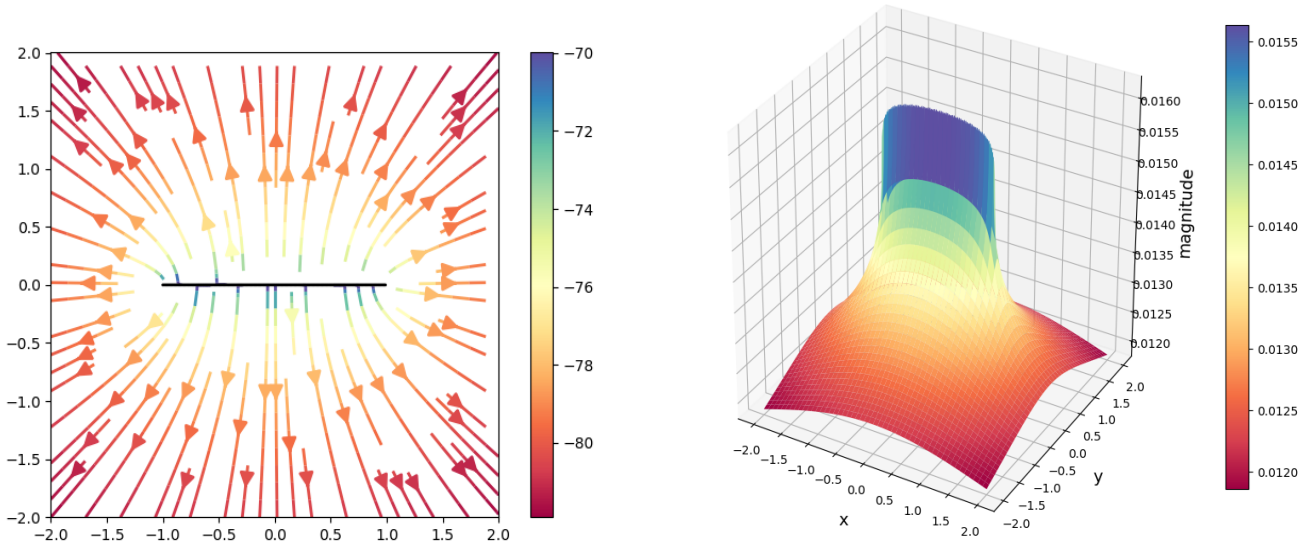
CODE

```
from EMPY.Electrostatics.System import ContinuousSystem

Q = 1.6e-19
xs = ys = [-1, 1]
```

```
#System 5 - Straight Wire
contSys2 = ContinuousSystem()
contSys2.straightWire([-1, 0], [1, 0], res=80, Q=1*Q)
contSys2.plot_VectField([-2,2],[-2,2], showEField=False, showEPot=True)
contSys2.plotPotential3D([-2,2],[-2,2],6)
```

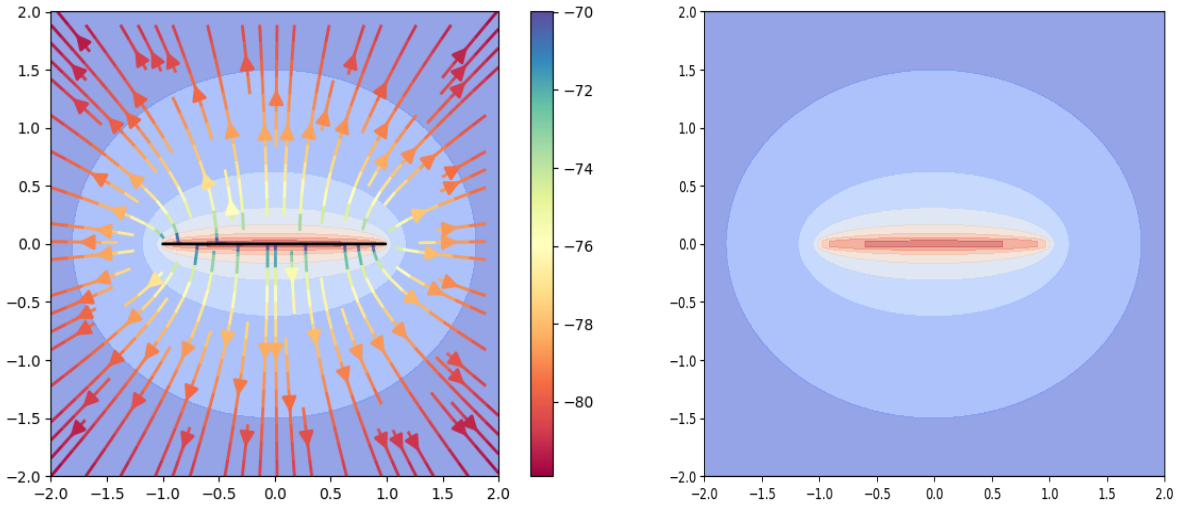
OUTPUT



(a) Electric Field Lines

(b) Mapping of Electric Potential

Figure 11: Electric Field Lines and Potential Mapping of a Straight Wire



(a) Electric Field Lines and Potential

(b) Electric Potential

Figure 12: Electric Field Lines and Potential of a Straight Wire

The above figures 11 and 12 show the electric field and potential for a straight current carry wire. Compared to the previous plots, this is a system consisting of continuous charge distribution. In figure 11a, the

electric field lines are perpendicular to the wire at the centre, and diverge outward towards the endpoints.

Figure 11b and 12b show the electric potential, where it has a maximum value at the position of the wire and gradually decreases as it moves farther away from the source.

3.3 Parallel Wires

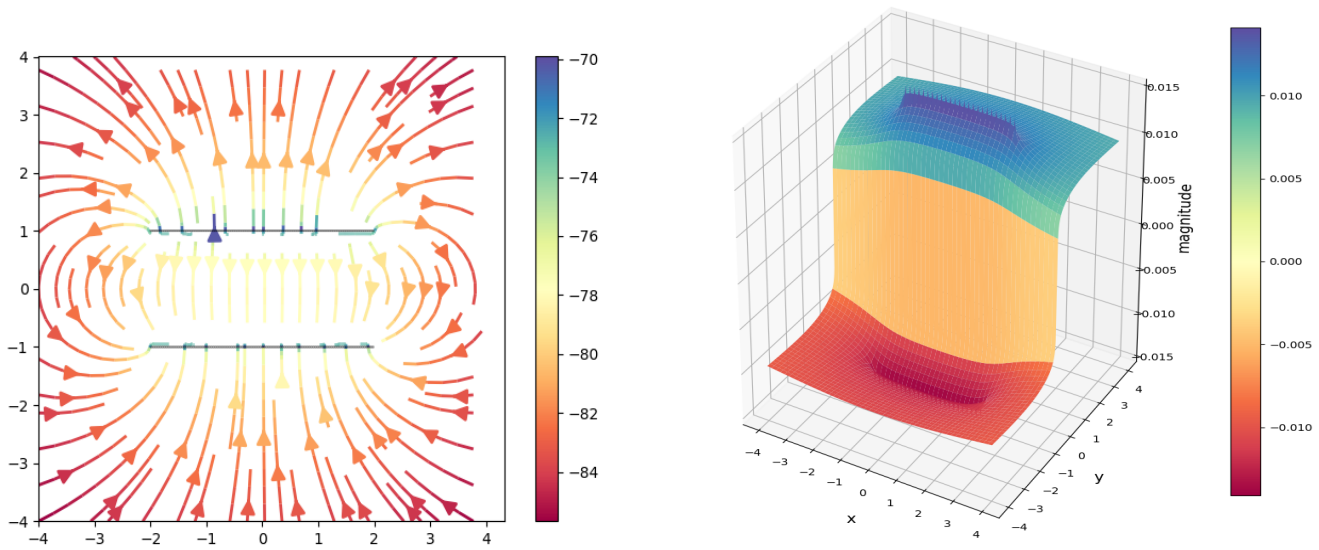
CODE

```
from EMPY.Electrostatics.System import ContinuousSystem

Q = 1.6e-19
xs = ys = [-1, 1]

#System 5 - Parallel Lines
contSys1 = ContinuousSystem()
contSys1.straightWire([-2, 1], [2, 1], res=80, Q=1*Q)
contSys1.straightWire([-2, -1], [2, -1], res=80, Q=-1*Q)
contSys1.plot_VectField([-4,4],[-4,4], showEField=True, showEPot=False)
contSys1.plotPotential3D([-4,4],[-4,4],6)
```

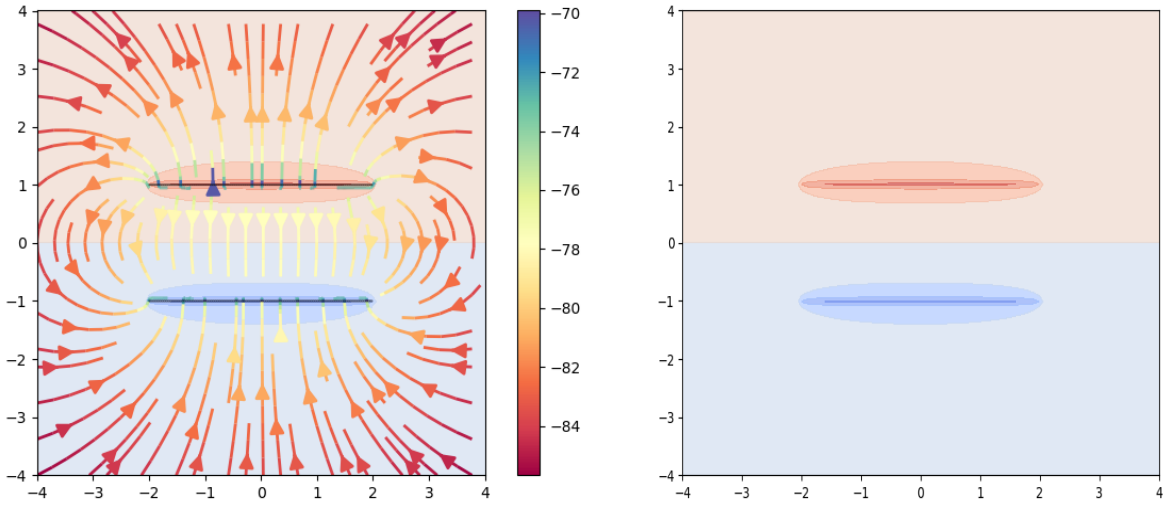
OUTPUT



(a) Electric Field Lines

(b) Mapping of Electric Potential

Figure 13: Electric Field Lines and Potential Mapping of Parallel Lines



(a) Electric Field Lines and Potential

(b) Electric Potential

Figure 14: Electric Field Lines and Electric Potential of Parallel Lines

The above figures depict the electric field and potential for a set of parallel wires; one has the opposite charge of the other. Figure 13a shows that the field lines between the parallel wires are parallel and equally spaced, except near the edges where they bulge outward. The equally spaced, parallel lines indicate that the electric field has the same magnitude and direction at all points in the central region of the capacitor.

Figure 13b show the electric potential where one can see that the electric potential inside the parallel wire increases linearly from the negative to the positive wire.

3.4 Circular Loop

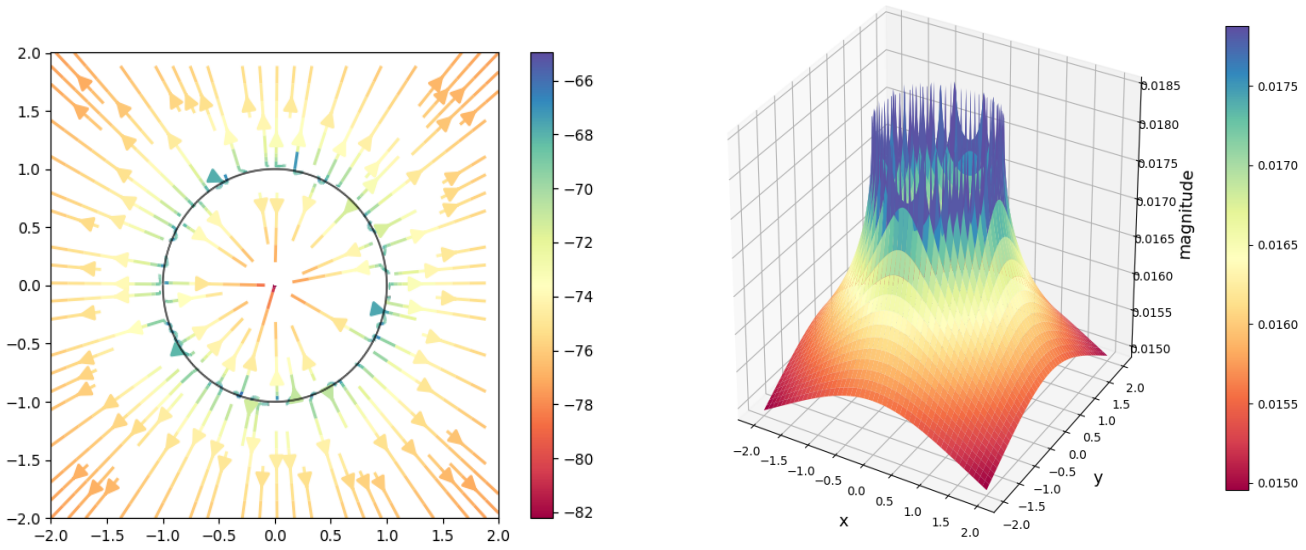
CODE

```
from EMPY.Electrostatics.System import ContinuousSystem

Q = 1.6e-19
xs = ys = [-1, 1]

#System 6 - Circular Loop
contSys3 = ContinuousSystem()
contSys3.circularWire([0,0], R=1, density=100, Q=100*Q)
contSys3.plot_VectField([-2,2],[-2,2], showEField=True, showEPot=False)
contSys3.plotPotential3D([-2,2],[-2,2],6)
```

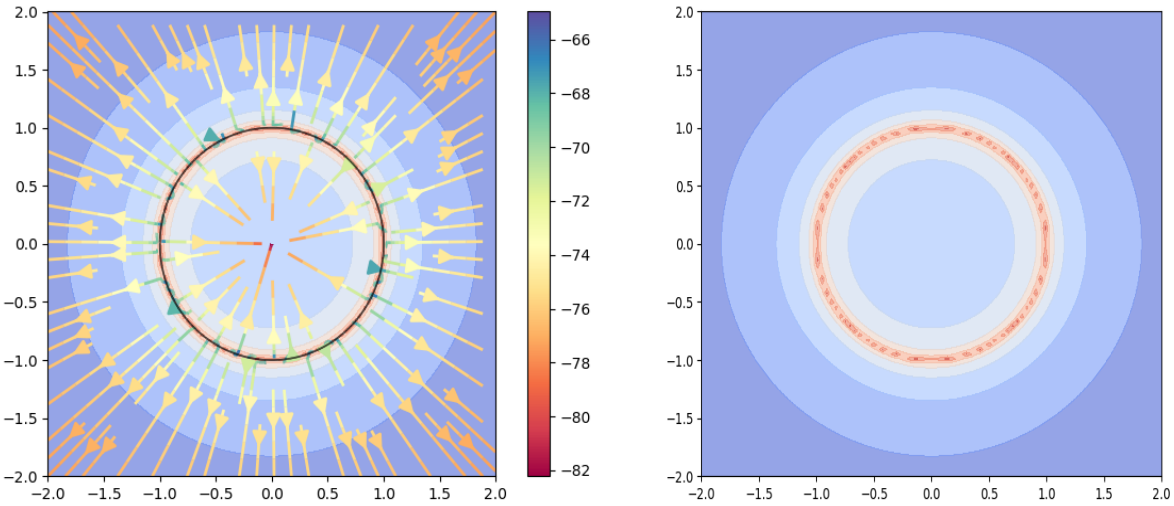
OUTPUT



(a) Electric Field Lines

(b) Mapping of Electric Potential

Figure 15: Electric Field Lines and Potential Mapping of a Circular Loop



(a) Electric Field Lines and Potential

(b) Electric Potential

Figure 16: Electric Field Lines and Potential of a Circular Loop

The above figures depict the electric field and potential for a circular loop of wire carrying current. From figure 15a and 16a, it is visible that outside the ring, the electric field lines are pointing outward and are all perpendicular to the surface. The field lines inside the ring are also perpendicular to the surface. However, the field lines now point to the centre.

Figure 15b and 16b portray the electric potential of the ring. The potential is maximum on where the ring is maximum and gradually decreases as it moves outward farther away. Notice that the electric potential inside the ring declines drastically. This is because, at the centre of the ring, all the electric field lines cancel each other out, causing the net electric field to be zero.

3.5 Square Plate

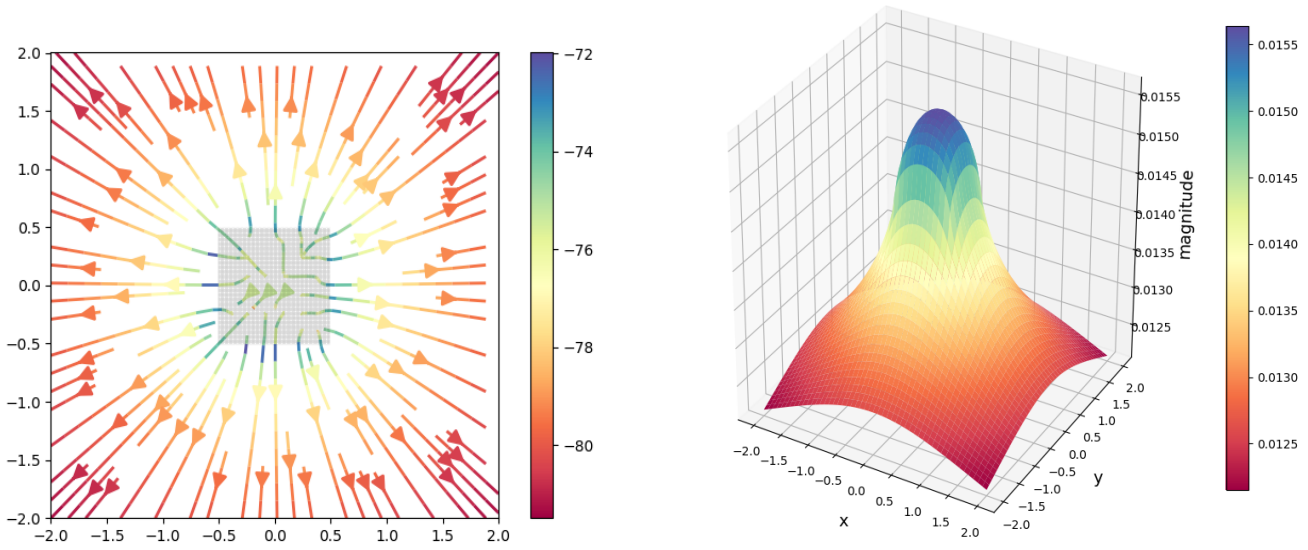
CODE

```
from EMPY.Electrostatics.System import ContinuousSystem

Q = 1.6e-19
xs = ys = [-1, 1]

#System 7 - Square Plate
contSys4 = ContinuousSystem()
contSys4.plate([1, 1], [-0.5, -0.5], density=80, Q=100*Q)
contSys4.plot_VectField([-2,2],[-2,2], showEField=True, showEPot=False)
contSys4.plotPotential3D([-2,2],[-2,2],6)
```

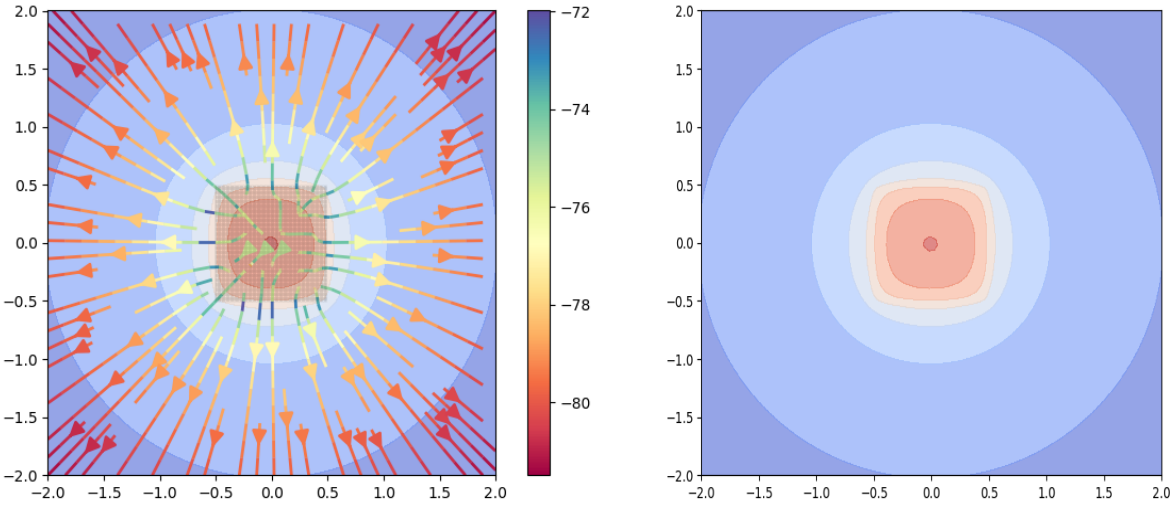
OUTPUT



(a) Electric Field Lines

(b) Mapping of Electric Potential

Figure 17: Electric Field Lines and Potential Mapping of a Square Plate



(a) Electric Field Lines and Potential

(b) Electric Potential

Figure 18: Electric Field Lines and Potential of a Square Plate

Figures 17a and 18a show the electric field generated by a charged plate. Due to the symmetry of the electric field intensity, the graph is also symmetric to the origin. We can observe that the vector of electric field changes direction in the centre of the plate, the electric intensity reaches negative values after crossing the centre of the plate. The electric field is uniform outside the plate.

Figures 17b and 18b show the electric potential of the electric plate. The electric potential is maximum at the centre of the square plate and gradually decreases as it gets farther away from the centre.

4 3D ELECTRIC FIELD VISUALISATION

4.1 Point Charge

CODE

```
from EMPY.Electrostatics.Charge3D import Charge3D

# 3D Plot of Electric Field of a Point Charge
charge3d = Charge3D(1, (0,0,1))
charge3d.plotField([-5,5],[-5,5],[-5,5])
```

OUTPUT

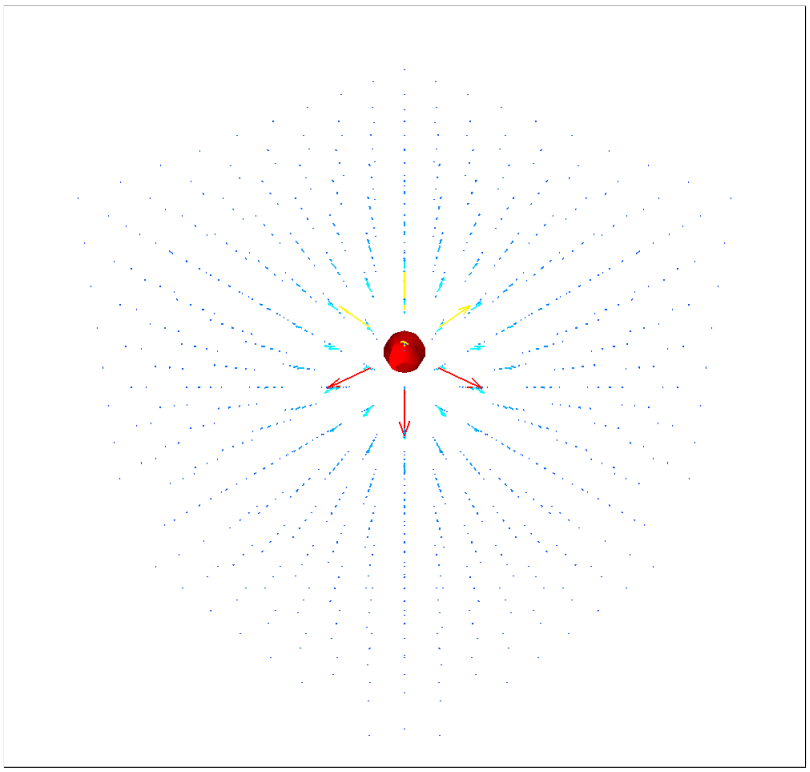


Figure 19: 3D Visualisation of Electric Field for a Point Charge

Figure 19 shows the three-dimensional electric fields generated by a unit positive charge. It is observable that the direction of field lines is radially outward for the positively charged particle. The length of the vector indicates the strength of the electric field. Similar to that of the two-dimensional plot, the magnitude decreases with distance.

4.2 Dipole

CODE

```
from EMPY.Electrostatics.System3D import DiscreteSystem3D
from EMPY.Electrostatics.Charge3D import Charge3D

# 3D Plot of Electric Field of a Dipole
charge3d_01 = Charge3D(-1, (-2,0,0))
charge3d_02 = Charge3D(1, (2,0,0))
discreteSys_01 = DiscreteSystem3D([charge3d_01, charge3d_02])
discreteSys_01.plotField([-5,5],[-5,5],[-5,5])
```

OUTPUT

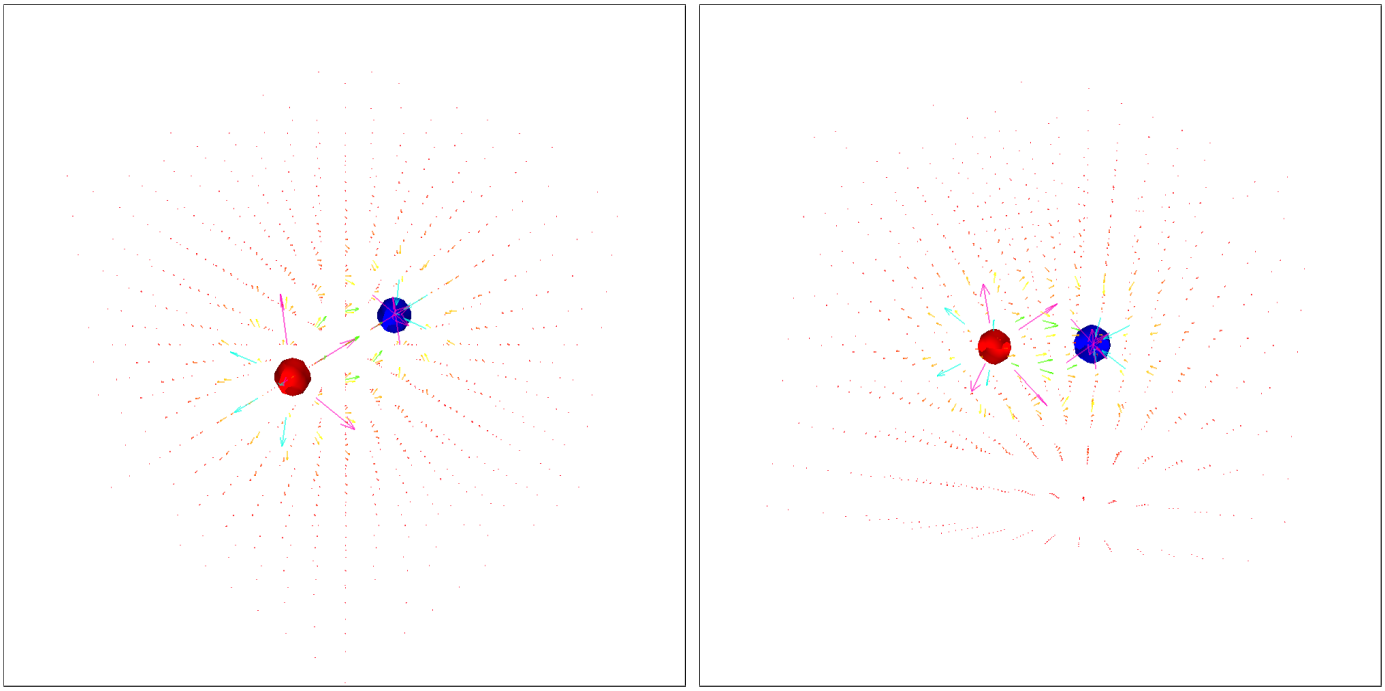


Figure 20: Electric Field Lines of a Dipole

Figure 20 shows the electric fields for a pair of charges of equal magnitude but the opposite sign (an electric dipole). The electric fields generated by an electric dipole have the following characteristics.

1. **Symmetry**: For every point above the line joining the two charges, there is an equivalent point below it. Therefore, the pattern must be symmetrical about the line joining the two charges.
2. **Near Field**: Very close to a charge, the field due to that charge predominates. Therefore, the lines are radial and spherically symmetric.
3. **Far Field**: Far from the system of charges, the pattern should look like that of a single point charge of value $Q = \sum_i Q_i$. Thus, the lines should be radially outward, unless $Q = 0$.

4.3 Infinitesimal Line Charge

CODE

```
from EMPY.Electrostatics.System3D import DiscreteSystem3D
from EMPY.Electrostatics.Charge3D import Charge3D

# Create a Infinitesimal Line Charge consisting of 5 3D Charge Objects
charge3d_03 = Charge3D(1, (-2,0,0))
charge3d_04 = Charge3D(1, (-1,0,0))
charge3d_05 = Charge3D(1, (0,0,0))
charge3d_06 = Charge3D(1, (1,0,0))
charge3d_07 = Charge3D(1, (2,0,0))
discreteSys_02 = DiscreteSystem3D([charge3d_03,charge3d_04,charge3d_05,charge3d_06,charge3d_07])
discreteSys_02.plotField([-5,5],[-5,5],[-5,5])
```

OUTPUT

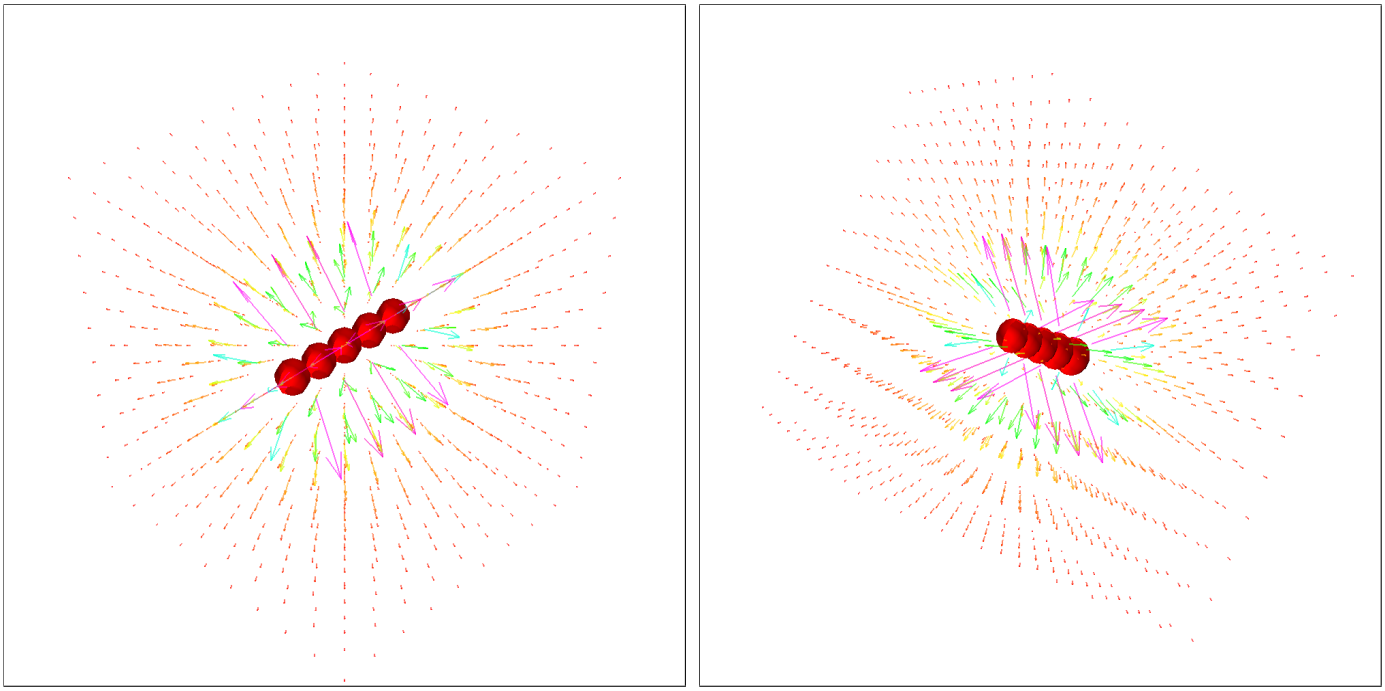


Figure 21: Electric Field Lines of an Infinitesimal Line Charge

4.4 Single Plate Charge

CODE

```
from EMPY.Electrostatics.System3D import DiscreteSystem3D
from EMPY.Electrostatics.Charge3D import Charge3D

# Create a Singe Plate in a Discrete Charge System
zo=0eo
dzo=2eo
chargeo=10eo
numptso = 9eo
qo = chargeo / numptso
chargeso = []
for x in range(-3,4):
    for y in range(-3,4):
        chargeso.append(Charge3D(qo, (x, y, zo)))
parallelPlate = DiscreteSystem3D(chargeso)
parallelPlate.plotField([-5,5],[-5,5],[-5,5])
```

OUTPUT

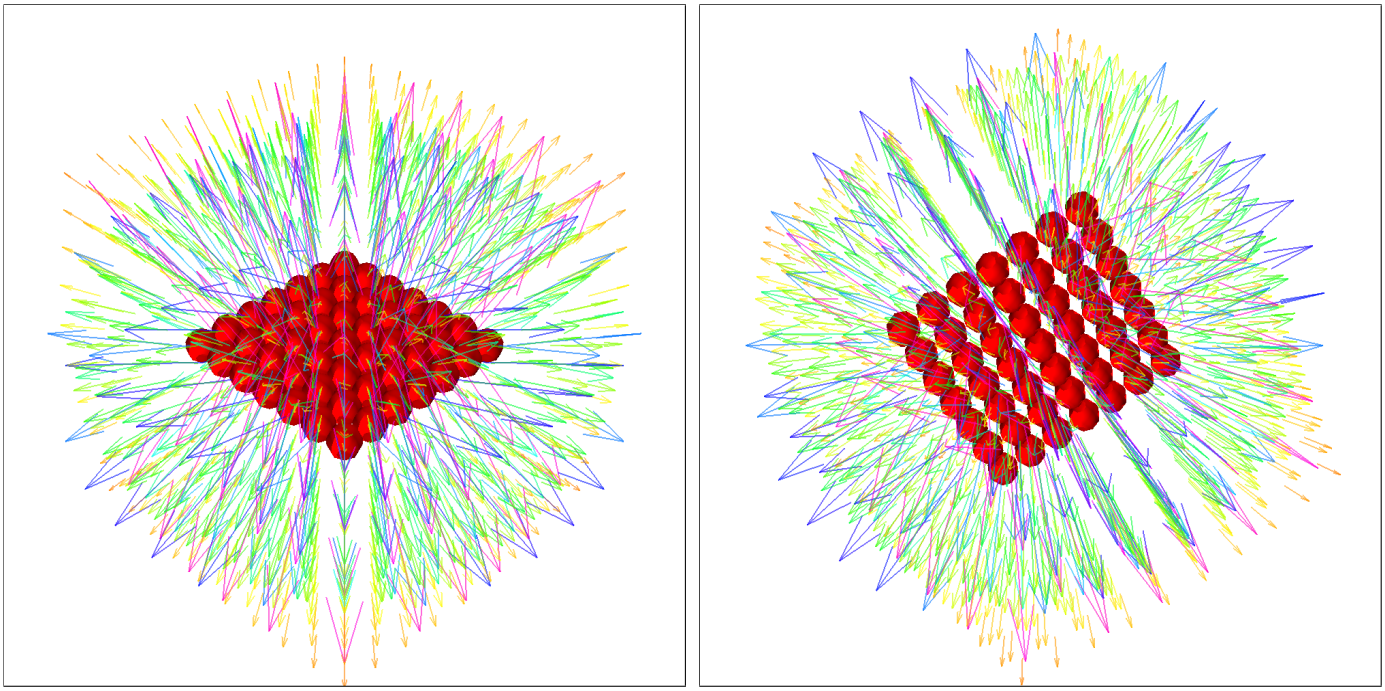


Figure 22: Electric Field Lines of a Single Plate Charge

Figure 22 represents the electric field lines generated by a positively charged plate formed by the collection of discrete points.

4.5 Parallel Plate Capacitor

CODE

```
from EMPY.Electrostatics.System3D import DiscreteSystem3D
from EMPY.Electrostatics.Charge3D import Charge3D

# Create a Parallel Plate Capacitor in a Discrete Charge System
z=0e0
dz=2e0
charge=10e0
numpts = 9e0
q = charge / numpts
charges = []
for x in range(-3,4):
    for y in range(-3,4):
        charges.append(Charge3D(q, (x, y, z + dz)))
        charges.append(Charge3D(-q, (x, y, z - dz)))
parallelPlate = DiscreteSystem3D(charges)
parallelPlate.plotField([-5,5],[-5,5],[-5,5])
```

OUTPUT

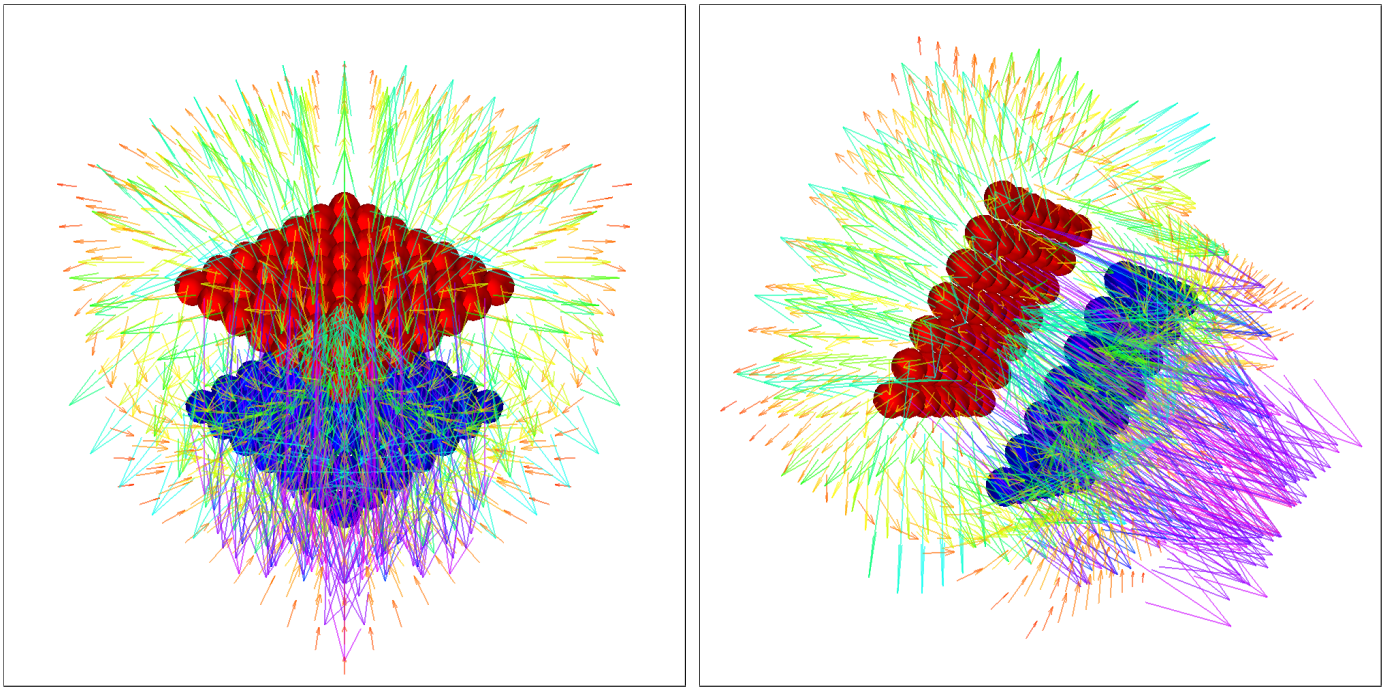


Figure 23: Electric Field Lines of a Parallel Plate Capacitor

Figure 23 depicts the electric field lines of a parallel plate capacitor. The red plate is positively charged, whereas the blue plate is negatively charged.

4.6 Spring

CODE

```
import numpy as np
from EMPY.Electrostatics.Charge3D import Charge3D
from EMPY.Electrostatics.System3D import ContinuousSystem3D

N=10
L=5e0
R=1.5e0
charge=50
numpts=10000
q = charge / numpts


charges = []
# Create a continuous charge distribution that forms a solenoid
z = np.linspace(0, L, numpts)
theta = 2 * np.pi * N / L * z
x = R * np.cos(theta)
y = R * np.sin(theta)

for i in range(len(z)):
    charges.append(Charge3D(q,[x[i],y[i],z[i]]))

solenoid = ContinuousSystem3D(charges)
solenoid.plotField([-6,6],[-6,6],[-6,6], (x,y,z))
```

OUTPUT

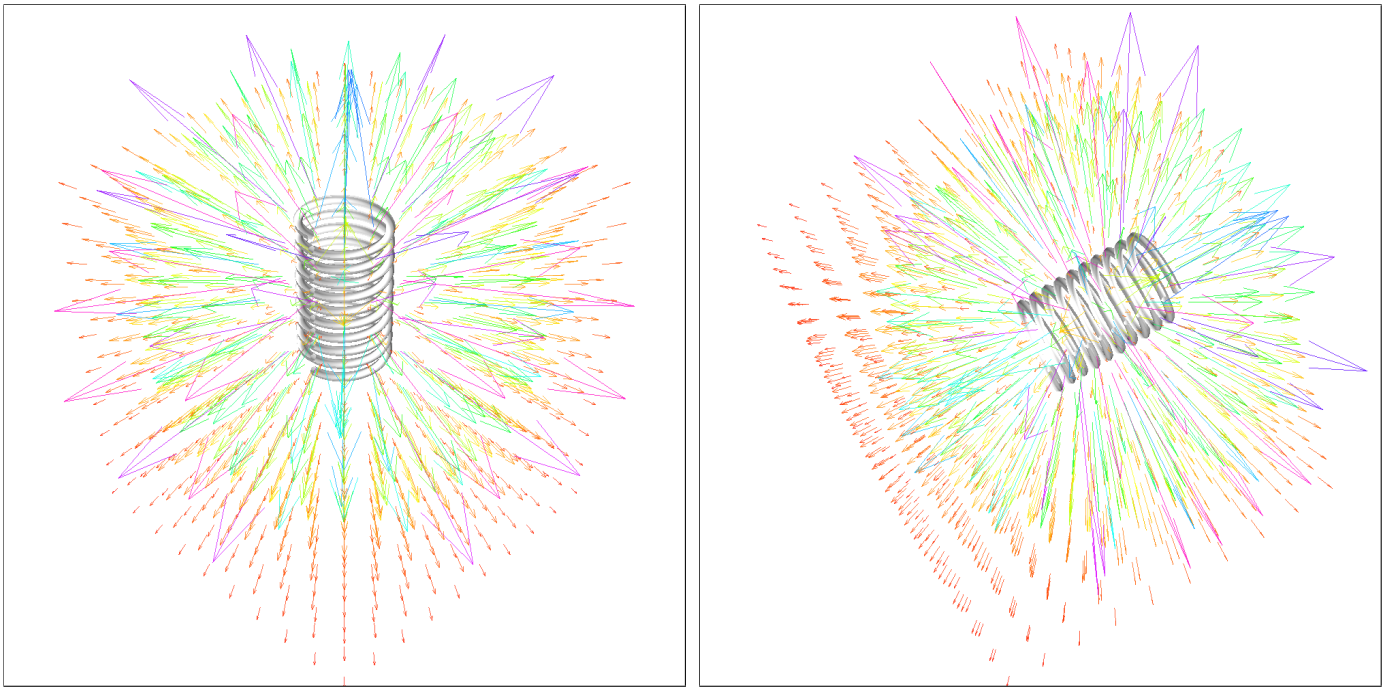


Figure 24: Electric Field Lines of a Spring

Figure 24 shows the electric field generated by a spring shaped object. This is similar to that of stacking circular loops with the charges evenly distributed among them.

4.7 Circular Ring

CODE

```
import numpy as np
from EMPY.Electrostatics.Charge3D import Charge3D
from EMPY.Electrostatics.System3D import ContinuousSystem3D

N=10
L=5e0
R=1.5e0
charge=50
numpts=10000
q = charge / numpts


charges = []

z = np.linspace(0, L, numpts)
theta = 2 * np.pi * N / L * z
x = R * np.cos(theta)
y = R * np.sin(theta)

for i in range(len(z)):
    charges.append(Charge3D(q,[x[i],y[i],0]))

circularLoop = ContinuousSystem3D(charges)
circularLoop.plotField([-6,6],[-6,6],[-6,6], (x,y,np.zeros(len(x))))
```

OUTPUT

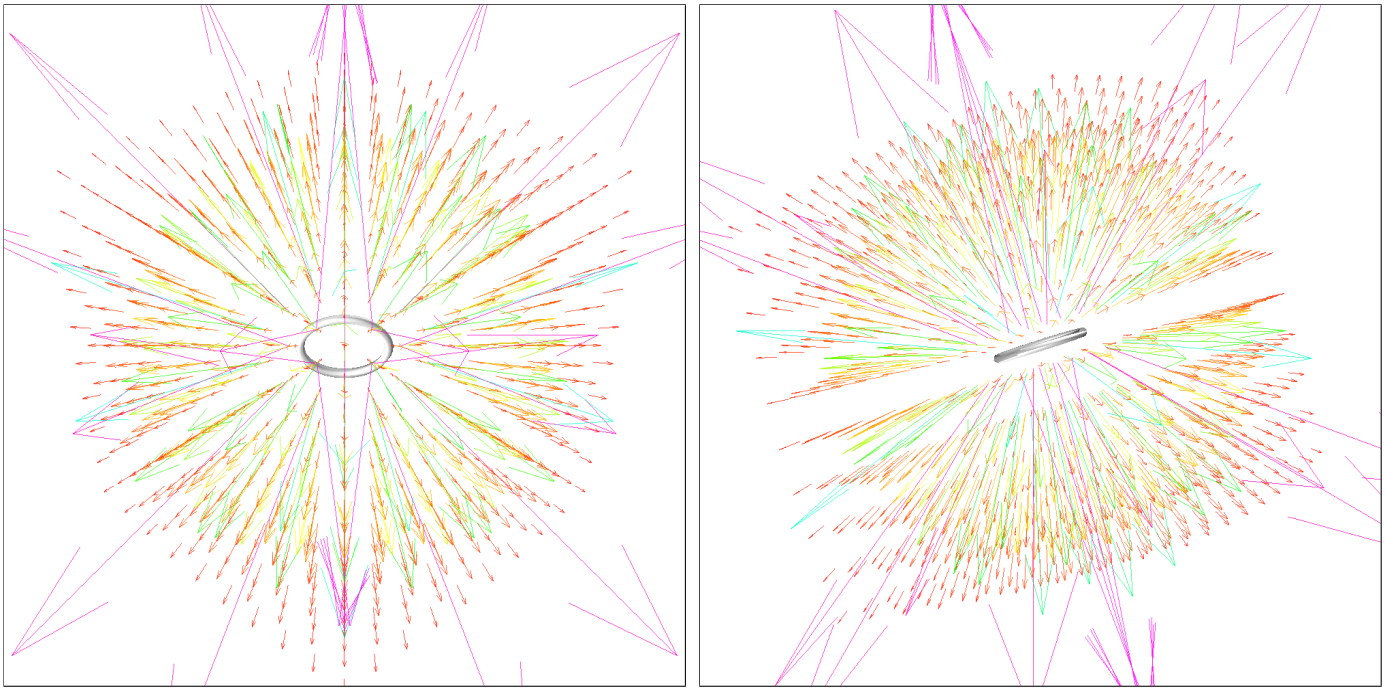


Figure 25: Electric Field Lines of a Circular Ring

Figure 25 shows the electric field generated by an electrically charged ring. It is observable that the electric fields in the xy plane are cancelled due to symmetry, and so only the z-components of the charge elements are added.

5 2D MAGNETIC FIELD VISUALISATION

5.1 Straight Wire

CODE

```
from EMPY.Magnetostatics import Wire

wire = Wire.Wire(0.5, [1,1])
wire.plotBField2D([-4,4],[-4,4])
```

OUTPUT

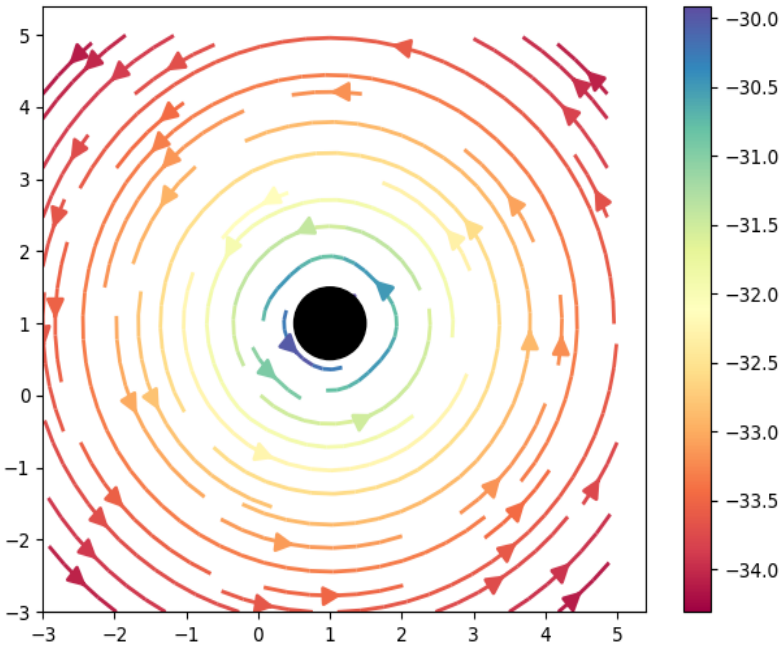


Figure 26: Magnetic Field Visualisation of a Straight Wire

The above figure, figure 26, depicts the magnetic field generated by a straight current-carrying conductor. Notice that in this, the system possesses cylindrical symmetry, and the magnetic field lines are circular. The right-hand rule can determine the direction of the magnetic field due to a long straight wire.

If you direct your right thumb along the direction of the current in the wire, then the fingers of your right hand curl in the direction of the magnetic field. In cylindrical coordinates (r, φ, z) where the unit vectors are related by $\hat{\mathbf{r}} \times \hat{\boldsymbol{\varphi}} = \hat{\mathbf{z}}$, if the current flows in the +z-direction, then, using the Biot-Savart law, the magnetic field must point in the φ-direction.

5.2 Single Circular Loop

CODE

```
from EMPY.Magnetostatics import Loops

# single-turn 10 cm x-oriented coil at origin
position = [0., 0., 0.]
normal = [1., 0., 0.]
radius = 10.
current = 1
```

```

coil = Loops.Loop(position, normal, radius, current)
cSystem = Loops.LoopSystem()
cSystem.addLoop(coil)

cSystem.plotBField(-15, 15, 101, -15, 15, 101)
```

OUTPUT

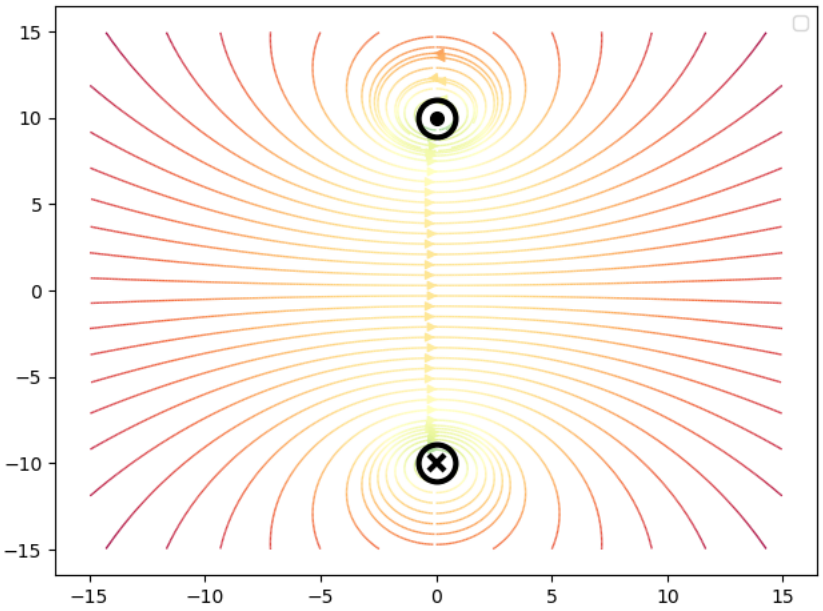


Figure 27: Magnetic Field Visualisation of a Circular Loop

The above plot is the magnetic fields generated by the current-carrying circular loop. It is noticeable that the electric current in a circular loop creates a magnetic field which is more concentrated in the centre of the loop than outside the loop. Stacking multiple loops concentrates the field even more into what is called a solenoid.

5.3 Helmholtz Coils

CODE

```
from EMPY.Magnetostatics import Loops

# single-turn 10 cm x-oriented coil at origin
# Helmholtz coil model with single current loops
R = 10.

# 2 windings
c1 = Loops.Loop([-R/2., 0, 0], [1, 0, 0], R, 1)
c2 = Loops.Loop([+R/2., 0, 0], [1, 0, 0], R, 1)


cSystem = Loops.LoopSystem()
cSystem.addLoop(c1)
cSystem.addLoop(c2)
cSystem.plotBField(-15, 15, 101, -15, 15, 101)
```

OUTPUT

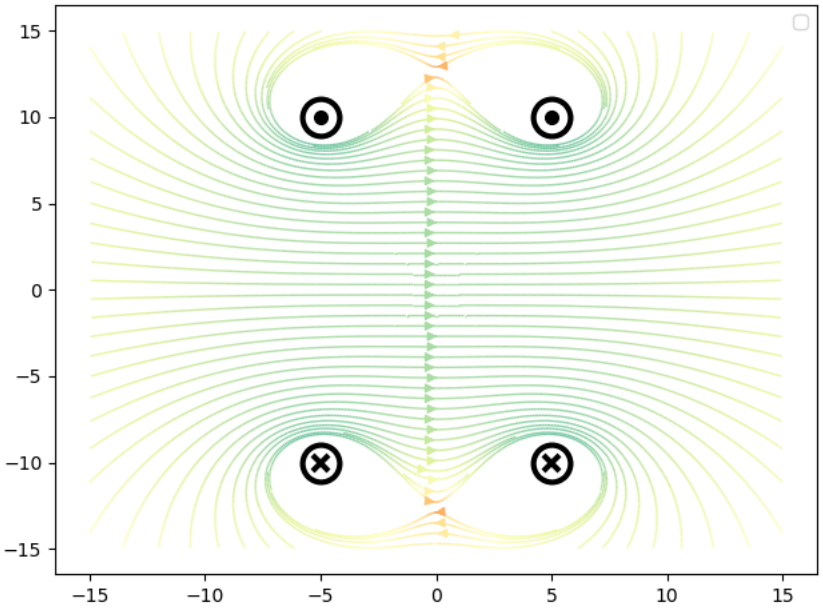


Figure 28: Magnetic Field Visualisation of a Helmholtz Coils

The graph in Figure 27 shows the magnetic field of the Helmholtz coils. In this configuration, the currents in the top and bottom coils flow in the same direction, with their dipole moments aligned. The magnetic fields from the two coils add up to create a net field that is nearly uniform at the centre of the coils. Since the distance between the coils is equal to the radius of the coils and remains unchanged, the force of attraction between them creates tension and is illustrated by field lines stretching out to enclose both coils. When the distance between the coils is not fixed, the two coils move toward each other due to their force of attraction.

5.4 Maxwell Coils

```python
import math
from EMPY.Magnetostatics import Loops

# Maxwell coil model with single current loops

R = 10

# center winding
c1 = Loops.Loop([0, 0, 0], [1, 0, 0], R, 64)

# outer windings
c2 = Loops.Loop([-R*math.sqrt(3./7.), 0, 0], [1, 0, 0], R*math.sqrt(4./7.), 49)
c3 = Loops.Loop([+R*math.sqrt(3./7.), 0, 0], [1, 0, 0], R*math.sqrt(4./7.), 49)


cSystem = Loops.LoopSystem()
cSystem.addLoop(c1)
cSystem.addLoop(c2)
cSystem.addLoop(c3)
cSystem.plotBField(-15, 15, 101, -15, 15, 101)
```

The above code produces the following visualisation of the magnetic field.

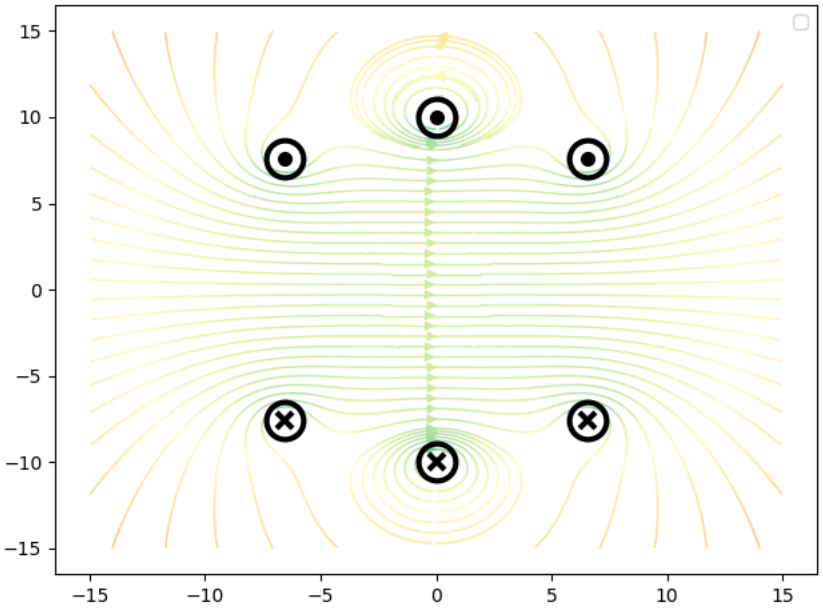


Figure 29: Magnetic Field Visualisation of a Maxwell Coils

Figure 29 is the visualisation of the magnetic fields generated by the system of Maxwell coil. A Maxwell coil is a device for producing a large volume of almost constant (or constant-gradient) magnetic field. It is visible from the above diagram that the magnetic field lines inside the coil set are uniform in nature.

6 3D MAGNETIC FIELD VISUALISATION

The following 3D visualisations generated are from configurations used in the 2D visualisations that had been explained earlier. The 3D visualisation of the circular loop was not generated using `EMPY`. It had been coded without any framework to satisfy my curiosity. More examples could not be developed for magnetostatics due to the increased complexity of its expressions.

6.1 Straight Wire

CODE

```
from EMPY.Magnetostatics import Wire

wire = Wire.Wire(0.5, [1,1])
wire.plotBField3D([-4,4],[-4,4], [-4,4])
```

OUTPUT

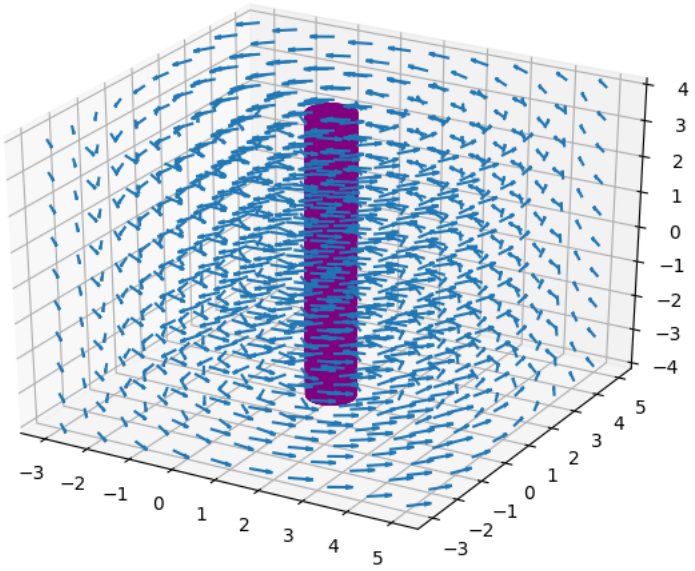


Figure 30: Magnetic Field Visualisation of a Straight Wire

6.2 Circular Loop

CODE

```
import numpy as np
from mayavi import mlab
from scipy import special

R = 1
I = 1
mewo = 4 * np.pi * 10. ** -7  #  o  constant

# Generating the 3D space
x, y, z = [i.astype(np.float32) for i in
                np.ogrid[-20:20:200j, -20:20:200j, -20:20:200j]]

r = np.sqrt(x ** 2 + y ** 2)
x_trans = x / r  # cos(a)
```

```
y_trans = y / r  # sin(a)

E = special.ellipe((4 * R * r) / ((R + r) ** 2 + z ** 2))  # special ellipse E
K = special.ellipk((4 * R * r) / ((R + r) ** 2 + z ** 2))  # special ellipse K
Bz = (I) / (2 * np.pi * np.sqrt((R + r) ** 2 + z ** 2)) * (
                K + E * (R ** 2 - r ** 2 - z ** 2) / ((R - r) ** 2 + z ** 2))

    # When r=0 there is a ZeroDivisionError for Br
try:
    Br = (I / (2 * np.pi * r)) * (z / (np.sqrt((R + r) ** 2 + z ** 2))) * (
                    -K + E * ((R ** 2 + r ** 2 + z ** 2) / ((R - r) ** 2 + z ** 2)))
except ZeroDivisionError:
    Br = 0
    # When the current I equals 0 there is no magnetic field
if I == 0:
    print('WARNING:', 'When I=0, \n There is no magnetic field generated')

else:
    mlab.close(all=True)
    Bx, By = x_trans * Br, y_trans * Br
    fig = mlab.figure(1, size=(500, 500), bgcolor=(1, 1, 1), fgcolor=(0, 0, 0))
    field = mlab.pipeline.vector_field(Bx, By, Bz)
    magnitude = mlab.pipeline.extract_vector_norm(field)
    contours = mlab.pipeline.iso_surface(magnitude,
                                         contours=[0.001, 0.8, 3.8, 4.0],
                                         transparent=True,
                                         opacity=0.6,
                                         colormap='YlGnBu',
                                         vmin=0, vmax=0.5)

    field_lines = mlab.pipeline.streamline(magnitude, seedtype='sphere',
                                           integration_direction='both',
                                           transparent=True,
                                           opacity=0.3,
                                           colormap='jet',
                                           vmin=0, vmax=0.5)

    field_lines.stream_tracer.maximum_propagation = 150.
    field_lines.seed.widget.radius = 5.5
    sc = mlab.scalarbar(field_lines, title='Field Strength [T]', orientation='vertical', nb_labels
    =4)
    sc.scalar_bar_representation.position2 = np.array([0.1, 0.8])
    sc.scalar_bar_representation.position = np.array([0.88374749, 0.14342105])

    mlab.show()
```

OUTPUT

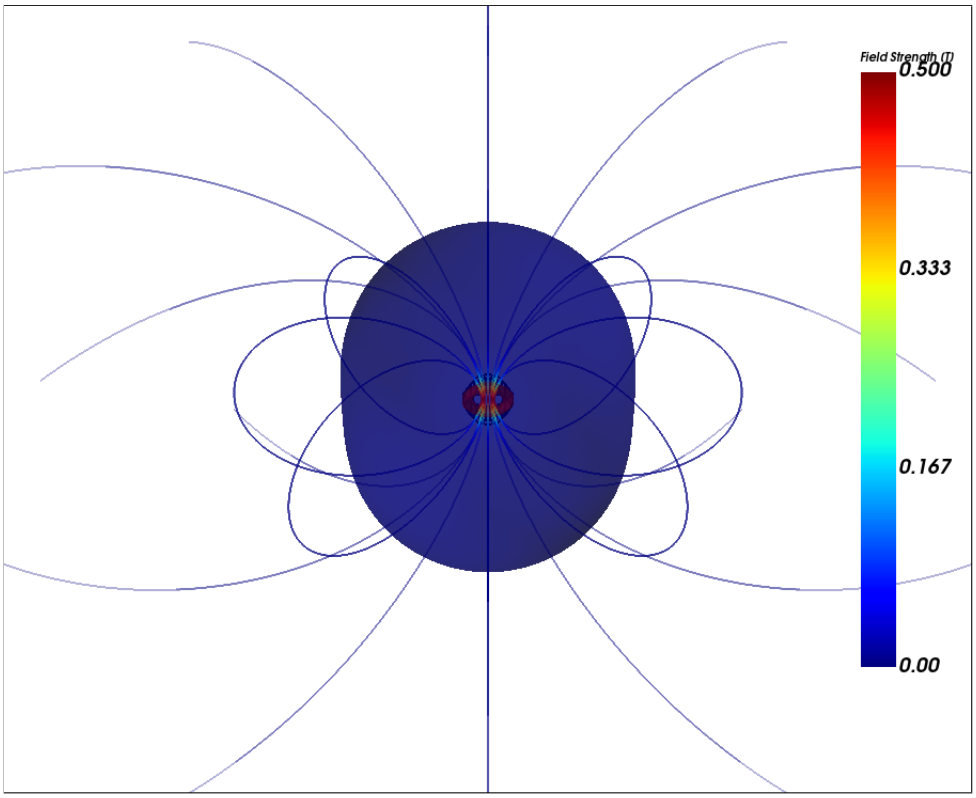


Figure 31: 3D Magnetic Field of Circular Loop

7 FURTHER REMARKS

For this HPO, a python library `EMPY - Electromagnetic Python` had been successfully developed. Visualisations of the electric field, electric potential and magnetic field were plotted for 18 different configurations of the system in 2D and 3D wherever possible. The graphs and plots obtained were in agreement with the numerical calculations and observations performed by various other physicists. While `EMPY` uses analytical results to generate plots of the electric and magnetic field, and may not be as accurate as those plots generated by solvers that employ techniques like FEM. However, its main strength is its efficient treatment of an extensive collection of charged systems that allows for faster visualisation and rendering of complex scenarios. Furthermore, its simplicity is a great advantage for the high-level python API as well as many tools for rapid data analysis and visualisation render standard simulations with ease. Scripts to reproduce all above-shown examples can be found within the `EMPY` library together with useful code comments.

Although `EMPY` does a great job, it could benefit from more features. For instance, the visualisations could also be constructed using numerical methods, and work can be done on numerical analysis and accuracy. To tackle the complexity of the expressions of magnetic fields, methods from `SYMPY` – Symbolic Python could be employed. They allow deriving the analytical expressions for any given configurations of the system. Other simple features such as integration of `PANDAS` for data analysis would also prove to be extremely useful for maintenance and analysis of the analytical and numerical data.

REFERENCES

[1] Rhett Allain. Use code to create sweet 3-d visualizations of electric fields, Jan 2016. URL https://www.wired.com/2016/01/use-code-to-create-sweet-3-d-visualizations-of-electric-fields/.

[2] Heiko Bauke, Aug 2015. URL https://www.numbercrunch.de/blog/2013/05/visualizing-vector-fields/.

[3] Bill, Rodrigo Cid Molina, Rodrigo Cid Molina, Rodrigo Cid Molina, Rodrigo Cid, and Andreas Hamre. Using mayavi to visualize electric fields, Sep 2010. URL https://elektromagnetisme.no/2010/09/25/using-mayavi-to-visualize-electric-fields/.

[4] Florian Le Bourdais. Charged particle trajectories in electric and magnetic fields. URL https://flothesof.github.io/charged-particle-trajectories-E-and-B-fields.html.

[5] David J Griffiths. *Introduction to electrodynamics; 4th ed.* Pearson, Boston, MA, 2013. doi: 1108420419. URL https://cds.cern.ch/record/1492149. Re-published by Cambridge University Press in 2017.

[6] Jianming Jin. *Theory and Computation of Electromagnetic Field*. John Wiley and Sons, 9 2010. ISBN 9780470533598. doi: 10.1002/9780470874257.

[7] G. Joos and I.M. Freeman. *Theoretical Physics*. Dover Books on Physics. Dover Publications, 1986. ISBN 9780486652276. URL https://books.google.com.au/books?id=vIw5m2XuvpIC.

[8] Bruce Knuteson, Eric Hudson, George Stephans, John Belcher, John Joannopoulos, Michael Feld, and Peter Dourmashkin. Electricity and magnetism. URL https://ocw.mit.edu/courses/physics/8-02t-electricity-and-magnetism-spring-2005/.

[9] Mic. Biot-savart law: magnetic field of a straight wire. URL http://firsttimeprogrammer.blogspot.com/2015/05/biot-savart-law-magnetic-field-of.html.

[10] James Simpson, John Lane, Christopher Immer, and Robert Youngquist. Simple analytic expressions for the magnetic field of a circular current loop. 06 2003.