# 26 Sep 23 - Activity: Electrostatic Fields

As we saw the electric field is described by the 4 Maxwell equations. But, in the electrostatic situation where the charges do not move, we find the electric field is described by the following equations:

These linear partial differential equations can be solved analytically for a number of situations. In this activity we will explore the electric field for a number of situations.

## The del operator

The del operator () is a differential vector operator that can act in the same way that vectors do (i.e., through a dot or cross product). In addition, it can operate on a scalar function to produce a vector function (i.e., ). The operator itself is given by:

where , , and are the unit vectors in the , , and directions, respectively. It acts on a scalar function as follows:

When taking the dot product (the divergence) of a vector function , it acts as follows:

And finally, when taking the cross product (the curl) of a vector function , it acts as follows:

We rarely make use of the full form of these differential equations, instead solving them in restricted and highly symmetric cases (e.g., using Gauss’s Law), computing directly the field contributed by infinitesimal charges (e.g., using Coulomb’s Law), or using numerical methods (e.g., using summation or other techniques).

Often, the most productive approach tends to be using electric potential (a scalar function), which we will discuss later.

## run to import libraries  
import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.colors import Normalize  
import matplotlib.cm as cm

## Coulomb’s Law and the Electric Field

As you have encountered in the past, a generic expression for the electric field observed at location due to a point charge at location is given by:

Notice that the electric field is a vector function of position and the differences that appear are vector ones:

We can show that the general solution for the electric field due to a set of point charges is simply the [superposition](https://en.wikipedia.org/wiki/Superposition_principle) of the individual fields due to each charge:

where is the electric field due to the ith charge.

### Visualization of the Electric Field

Below we have written a a function that computes the electric field for a point charge at a known location. We use a few different libraries and known functions that we have used in the past for dynamical systems. But we apply them to the electric field.

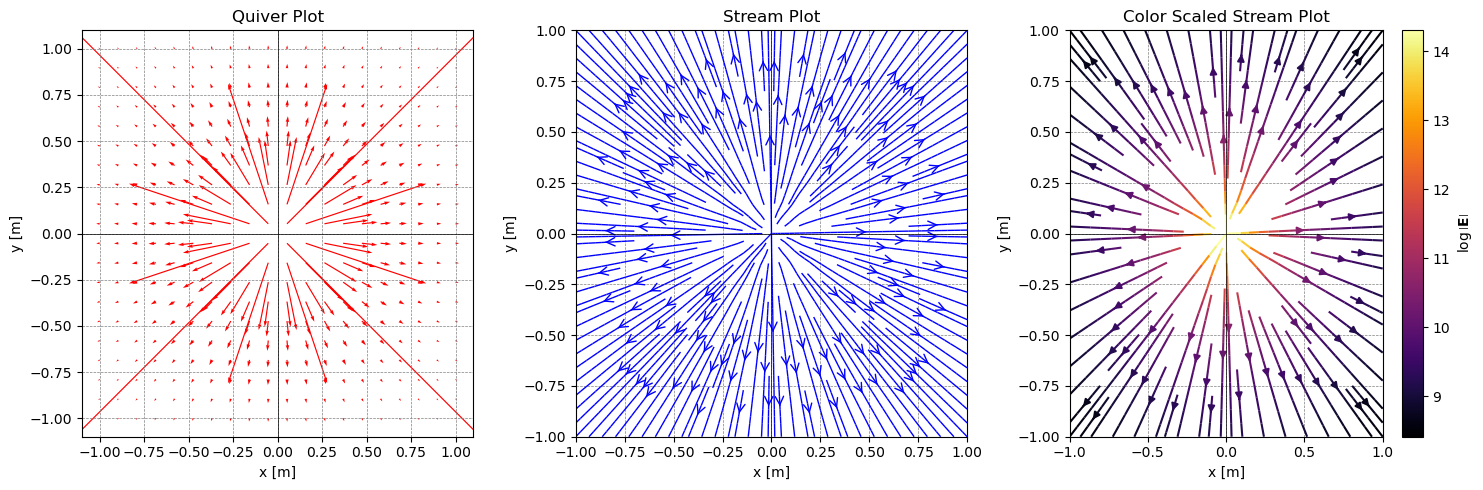
def electric\_field(charge, x\_points, y\_points, x\_charge=0, y\_charge=0):  
 k = 8.99e9 # Nm^2/C^2, Coulomb's constant  
   
 # Initialize electric field components to zero  
 E\_x = np.zeros\_like(x\_points)  
 E\_y = np.zeros\_like(y\_points)  
   
 # Calculate electric field components due to the point charge at each point on the grid  
 for i in range(x\_points.shape[0]):  
 for j in range(y\_points.shape[1]):  
 r\_x = x\_points[i, j] - x\_charge  
 r\_y = y\_points[i, j] - y\_charge  
 r\_magnitude = np.sqrt(r\_x\*\*2 + r\_y\*\*2)  
 if r\_magnitude != 0: # Avoid division by zero  
 r\_unit\_x = r\_x / r\_magnitude  
 r\_unit\_y = r\_y / r\_magnitude  
 E\_x[i, j] = k \* charge \* r\_unit\_x / r\_magnitude\*\*2  
 E\_y[i, j] = k \* charge \* r\_unit\_y / r\_magnitude\*\*2  
   
 return E\_x, E\_y  
  
def plot\_electric\_field(X, Y, E\_x, E\_y, stream\_color='b'):  
 fig, axs = plt.subplots(1, 3, figsize=(15, 5))  
   
 # First Subplot - Quiver Plot  
 axs[0].quiver(X, Y, E\_x, E\_y, scale=1e6, color='r')  
 axs[0].set\_title('Quiver Plot')  
 axs[0].set\_xlabel('x [m]')  
 axs[0].set\_ylabel('y [m]')  
 axs[0].axhline(0, color='black', linewidth=0.5)  
 axs[0].axvline(0, color='black', linewidth=0.5)  
 axs[0].grid(color='gray', linestyle='--', linewidth=0.5)  
   
 # Second Subplot - Stream Plot  
 axs[1].streamplot(X, Y, E\_x, E\_y, color=stream\_color, linewidth=1, density=2, arrowstyle='->', arrowsize=1.5)  
 axs[1].set\_title('Stream Plot')  
 axs[1].set\_xlabel('x [m]')  
 axs[1].set\_ylabel('y [m]')  
 axs[1].axhline(0, color='black', linewidth=0.5)  
 axs[1].axvline(0, color='black', linewidth=0.5)  
 axs[1].grid(color='gray', linestyle='--', linewidth=0.5)  
   
 # Third Subplot - Color Scaled Quiver Plot  
 magnitude = np.sqrt(E\_x\*\*2 + E\_y\*\*2)  
 colors = np.log(magnitude)  
 strm = axs[2].streamplot(X, Y, E\_x, E\_y, color=colors, cmap = cm.inferno)  
 axs[2].set\_title('Color Scaled Stream Plot')  
 axs[2].set\_xlabel('x [m]')  
 axs[2].set\_ylabel('y [m]')  
 axs[2].axhline(0, color='black', linewidth=0.5)  
 axs[2].axvline(0, color='black', linewidth=0.5)  
 axs[2].grid(color='gray', linestyle='--', linewidth=0.5)  
 # Add a colorbar to the plot  
 cbar = plt.colorbar(strm.lines)  
 cbar.set\_label(r'$\log{\left|\mathbf{E}\right|}$')  
  
   
 plt.tight\_layout() # Adjust spacing between subplots  
 plt.show()

### A point charge

Below, we plot the electric field in 2D of a single point charge. We are using several different representations of the electric field. Sometimes, some work better than others. The code we wrote, should be able to handle multiple point charges. And you will need to create a list of charges and their locations to investigate the field of multiple charges.

#### Positive Point Charge at the Origin

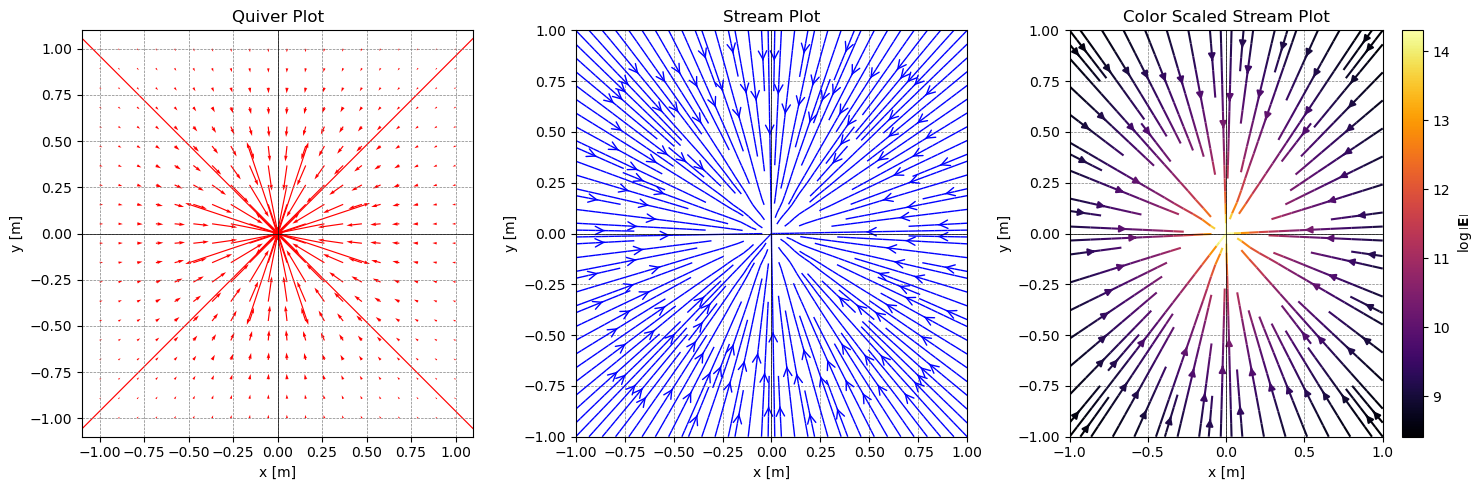
## Set Grid of Points  
x = np.linspace(-1, 1, 20)  
y = np.linspace(-1, 1, 20)  
X, Y = np.meshgrid(x, y)  
  
## A point charge  
charge = 1e-6 # C  
  
E\_x, E\_y = electric\_field(charge, X, Y)  
plot\_electric\_field(X, Y, E\_x, E\_y)



png

#### Negative Point Charge at the Origin

x = np.linspace(-1, 1, 20)  
y = np.linspace(-1, 1, 20)  
X, Y = np.meshgrid(x, y)  
  
## A negative point charge  
charge = -1e-6 # C  
  
E\_x, E\_y = electric\_field(charge, X, Y)  
plot\_electric\_field(X, Y, E\_x, E\_y)

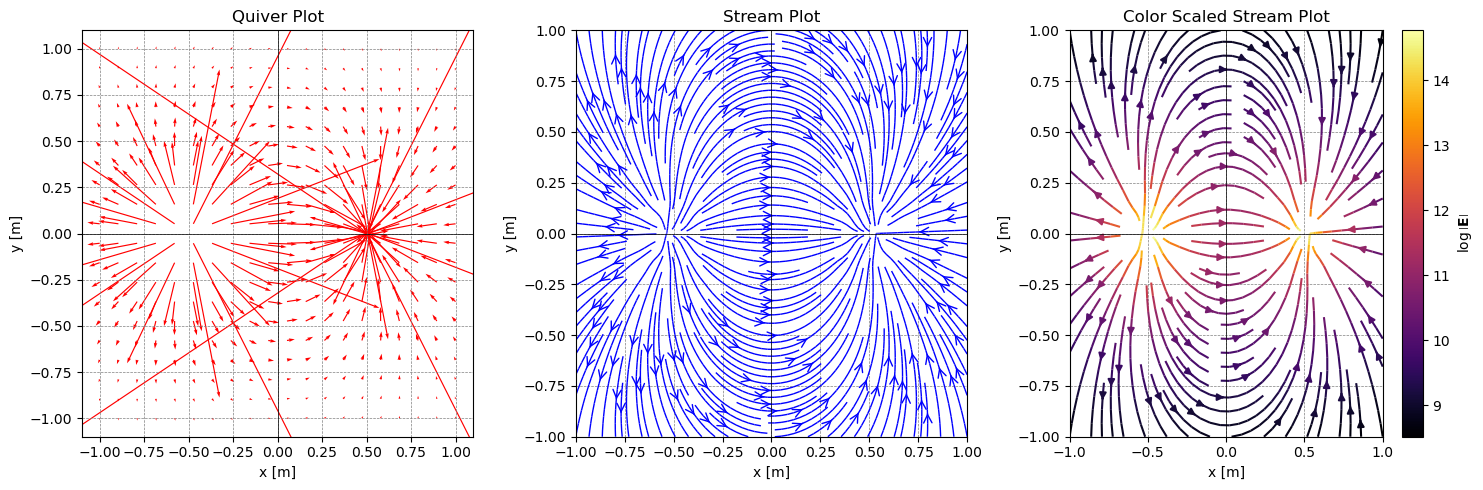


png

### The electric dipole

Now we can use these functions to place charges where we like and compute the total field. Below, we placed two oppositely signed charges to illustrate the field of an [electric dipole](https://en.wikipedia.org/wiki/Electric_dipole_moment). While not every distribution has an electric dipole moment, many do; especially those in water (as it’s a polar molecule).

x = np.linspace(-1, 1, 20)  
y = np.linspace(-1, 1, 20)  
X, Y = np.meshgrid(x, y)  
  
## A positive point charge  
charge1 = 1e-6 # C  
x1 = -0.5 # m  
y1 = 0 # m  
  
## A negative point charge  
charge2 = -1e-6 # C  
x2 = 0.5 # m  
y2 = 0 # m  
  
E\_1\_x, E\_1\_y = electric\_field(charge1, X, Y, x1, y1)  
E\_2\_x, E\_2\_y = electric\_field(charge2, X, Y, x2, y2)  
  
  
## Find the net electric field in each direction  
E\_T\_x = E\_1\_x + E\_2\_x  
E\_T\_y = E\_1\_y + E\_2\_y  
  
plot\_electric\_field(X, Y, E\_T\_x, E\_T\_y)



png

### Quadrupole

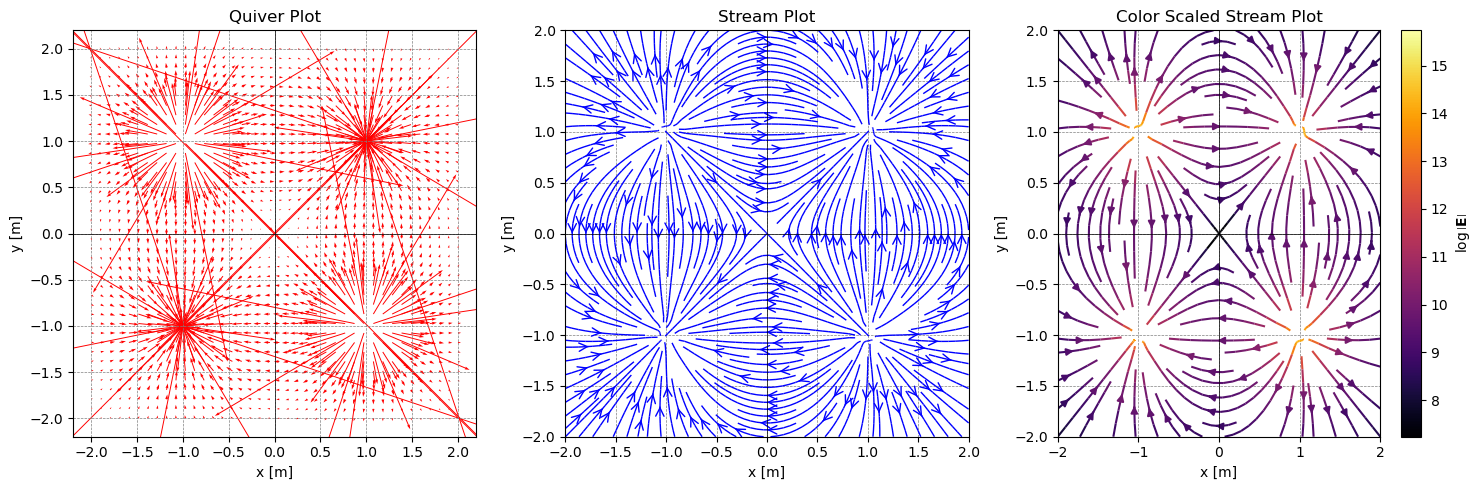
We will end this game with a list of charges that we can iterate through. This will help you develop more complicated distributions of charges.

Below, we have a [quadrupole](https://en.wikipedia.org/wiki/Quadrupole). You can change the charges and their locations to see how the field changes. Your activity below is to develop several situations and see how the field changes and consider the symmetry of the field. For this we will make a [Python class](https://docs.python.org/3/tutorial/classes.html) for the charges. This will allow us to create a list of charges and iterate through them. This is showing an advantage of using aspects of [object oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming).

class Charge:  
 def \_\_init\_\_(self, q, x, y):  
 self.q = q # charge value  
 self.x = x # x-coordinate  
 self.y = y # y-coordinate  
  
# Creating instances of Charge class  
charge1 = Charge(1e-6, -1, 1) # e.g. Charge of 1 μC at (0, 0)  
charge2 = Charge(-1e-6, 1, 1) # e.g. Charge of -1 μC at (1, 0)  
charge3 = Charge(1e-6, 1, -1) # e.g. Charge of 1 μC at (0, 1)  
charge4 = Charge(-1e-6, -1, -1) # e.g. Charge of -1 μC at (1, 1)  
  
# Storing instances in a list  
charges = [charge1, charge2, charge3, charge4]  
  
# Iterating over the list of charges  
for charge in charges:  
 print(f"Charge: {charge.q} C, Location: ({charge.x}, {charge.y})")

Charge: 1e-06 C, Location: (-1, 1)  
Charge: -1e-06 C, Location: (1, 1)  
Charge: 1e-06 C, Location: (1, -1)  
Charge: -1e-06 C, Location: (-1, -1)

## Set up the space again  
x = np.linspace(-2, 2, 40)  
y = np.linspace(-2, 2, 40)  
X, Y = np.meshgrid(x, y)  
  
## Set the total electric field to zero  
E\_T\_x = np.zeros\_like(X)  
E\_T\_y = np.zeros\_like(Y)  
  
## Calculate the electric field due to each charge  
for charge in charges:  
 E\_x, E\_y = electric\_field(charge.q, X, Y, charge.x, charge.y)  
 E\_T\_x += E\_x  
 E\_T\_y += E\_y  
  
plot\_electric\_field(X, Y, E\_T\_x, E\_T\_y)



png

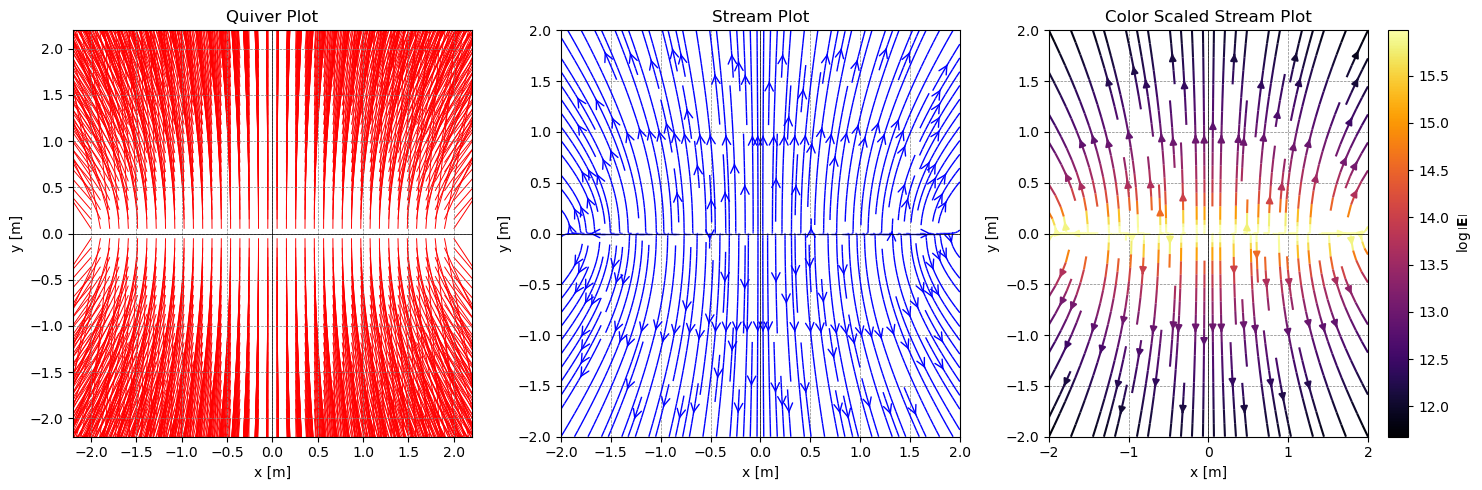
**✅ Do this**

Now that we have built up some visualization codes. Let’s make use of them. Create the following graphics and answer the following questions.

* Create a uniform line of charges. Make sure to include enough charges so that the field looks smooth. What is the symmetry of this situation. Where is the electric field constant (if at all)? Where is the electric field zero (if at all)?
* What if the line of charges varying their strength like a sinusoidal function?
* Create a circle of charges (like a ring). Is there any symmetry to this situation? Where is the electric field zero? How close does your calculation get to zero? What makes the calculation better or worse?
* Create a small locus of charges of different sizes and signs, but make sure the net charge is non-zero. How close do the charges need to be (or how far away do you need to be) for the field to look like a point charge field? What is the symmetry of this situation (or approximately so)? Where is the electric field zero (if at all)?

#### Uniform line of charges

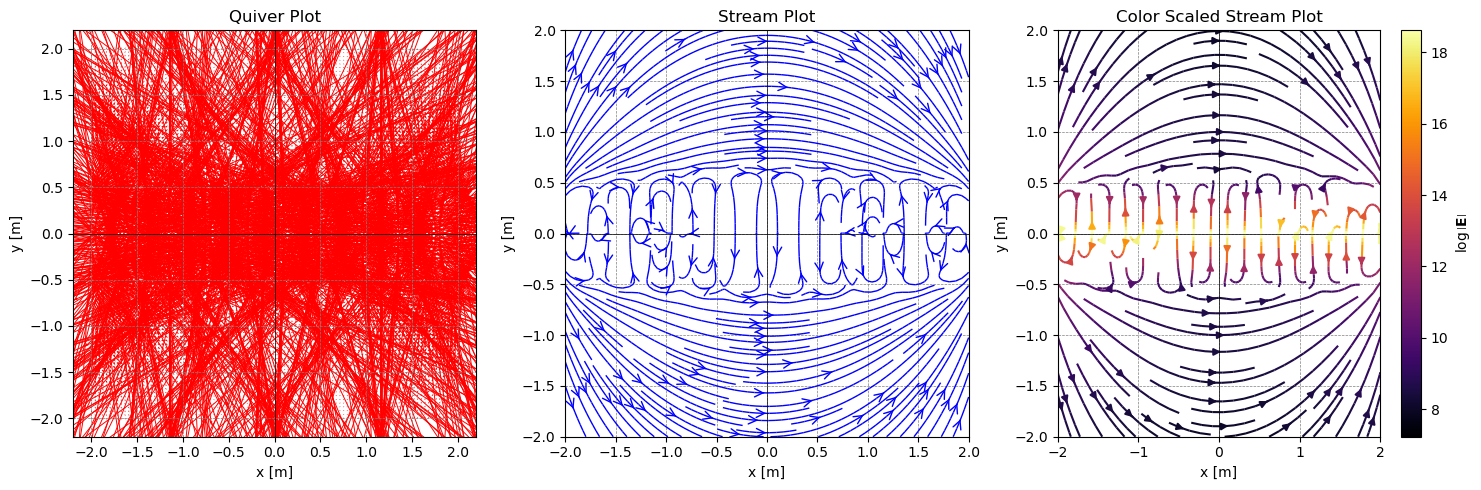
## Set up the space again  
size = 2  
step = 40  
  
x = np.linspace(-size, size, step)  
y = np.linspace(-size, size, step)  
X, Y = np.meshgrid(x, y)  
  
N = 100 # Number of charges  
charges = []  
for i in range(N):  
 q = 1e-6 # Charge Value  
 x = i\*(2\*size)/(N-1) - size # Spread the charges out  
 y = 0 # everything on the x-axis  
 charges.append(Charge(q, x, y)) # Add the charge to the list of charges  
  
## Set the total electric field to zero  
E\_T\_x = np.zeros\_like(X)  
E\_T\_y = np.zeros\_like(Y)  
  
## Calculate the electric field due to each charge  
for charge in charges:  
 E\_x, E\_y = electric\_field(charge.q, X, Y, charge.x, charge.y)  
 E\_T\_x += E\_x  
 E\_T\_y += E\_y  
  
plot\_electric\_field(X, Y, E\_T\_x, E\_T\_y)



png

#### Sinusoidal line of charges

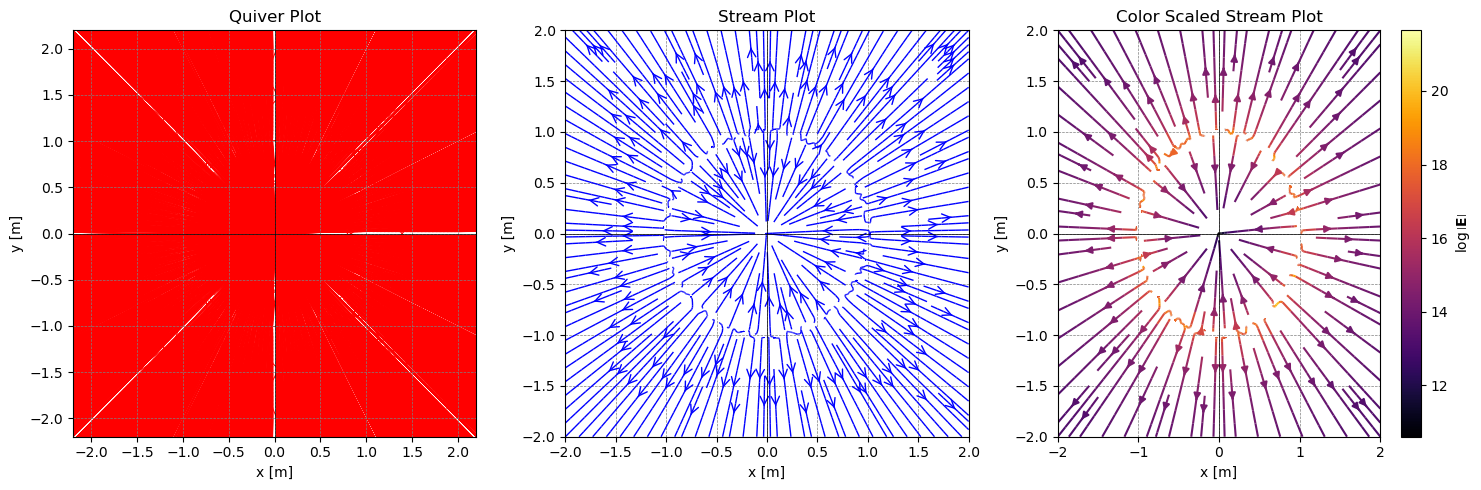
## Set up the space again  
size = 2  
step = 100  
  
x = np.linspace(-size, size, step)  
y = np.linspace(-size, size, step)  
X, Y = np.meshgrid(x, y)  
  
N = 600 # Number of charges  
charges = []  
for i in range(N):  
 x = i\*(2\*size)/(N-1) - size # Spread the charges out  
 y = 0 # everything on the x-axis  
   
 q = 1e-6\*np.sin(30\*x/(size)) # Charge Value  
   
 charges.append(Charge(q, x, y)) # Add the charge to the list of charges  
  
## Set the total electric field to zero  
E\_T\_x = np.zeros\_like(X)  
E\_T\_y = np.zeros\_like(Y)  
  
## Calculate the electric field due to each charge  
for charge in charges:  
 E\_x, E\_y = electric\_field(charge.q, X, Y, charge.x, charge.y)  
 E\_T\_x += E\_x  
 E\_T\_y += E\_y  
  
plot\_electric\_field(X, Y, E\_T\_x, E\_T\_y)



png

#### Circle of charges

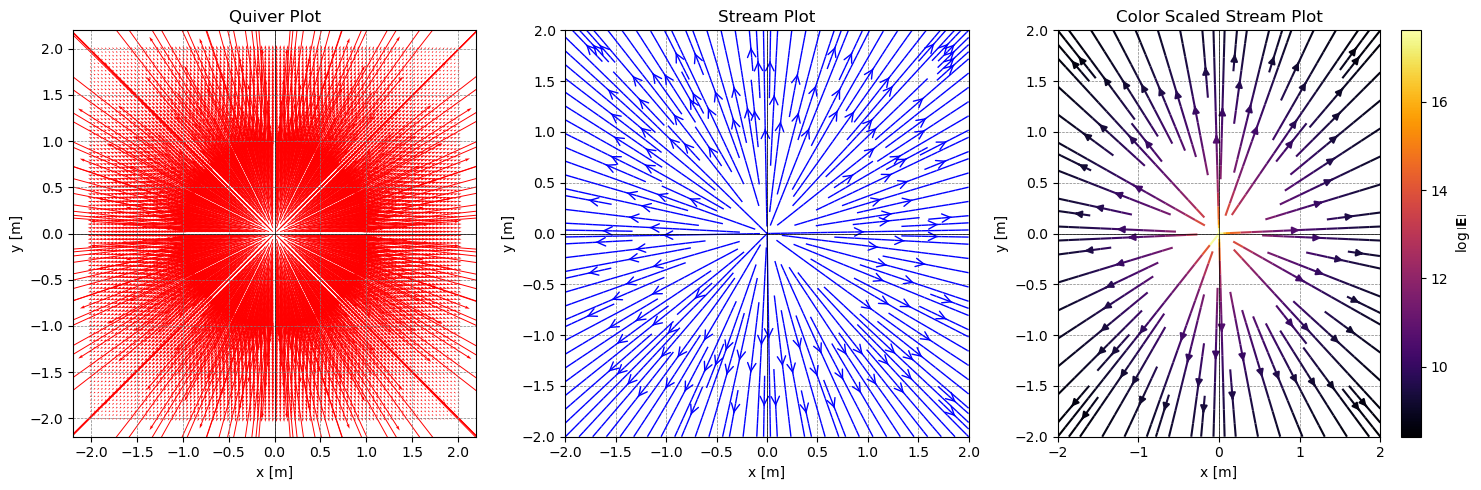
## Set up the space again  
size = 2  
step = 100  
R = size/2  
  
x = np.linspace(-size, size, step)  
y = np.linspace(-size, size, step)  
X, Y = np.meshgrid(x, y)  
  
N = 360 # Number of charges  
charges = []  
for i in range(N):  
   
 theta = i\*(2\*np.pi)/(N-1) # Spread the charges out  
 x = R\*np.cos(theta)  
 y = R\*np.sin(theta)  
   
 q = 1e-6 # Charge Value  
   
 charges.append(Charge(q, x, y)) # Add the charge to the list of charges  
  
## Set the total electric field to zero  
E\_T\_x = np.zeros\_like(X)  
E\_T\_y = np.zeros\_like(Y)  
  
## Calculate the electric field due to each charge  
for charge in charges:  
 E\_x, E\_y = electric\_field(charge.q, X, Y, charge.x, charge.y)  
 E\_T\_x += E\_x  
 E\_T\_y += E\_y  
  
plot\_electric\_field(X, Y, E\_T\_x, E\_T\_y)



png

#### Small Locus of Random Charges

## Set up the space again  
size = 2  
step = 100  
R = size/2  
scale = 1e-3 #smaller scale means charges are closer to the origin  
  
x = np.linspace(-size, size, step)  
y = np.linspace(-size, size, step)  
X, Y = np.meshgrid(x, y)  
  
N = 6 # If even there's a chance of a net charge of zero  
charges = []  
for i in range(N):  
   
 ## Random charge value  
 q = np.random.choice([-1e-6, 1e-6]) # Either a positive or negative charge  
 ## Random x,y locations that are located close to the origin  
 x = np.random.uniform(-scale\*size, scale\*size)  
 y = np.random.uniform(-scale\*size, scale\*size)  
   
 charges.append(Charge(q, x, y)) # Add the charge to the list of charges  
  
## Set the total electric field to zero  
E\_T\_x = np.zeros\_like(X)  
E\_T\_y = np.zeros\_like(Y)  
  
## Calculate the electric field due to each charge  
for charge in charges:  
 E\_x, E\_y = electric\_field(charge.q, X, Y, charge.x, charge.y)  
 E\_T\_x += E\_x  
 E\_T\_y += E\_y  
  
plot\_electric\_field(X, Y, E\_T\_x, E\_T\_y)  
  
total\_charge = 0  
  
for charge in charges:  
 print(f"Charge: {charge.q} C, Location: ({charge.x}, {charge.y})")  
 total\_charge += charge.q  
   
print(f"Total Charge: {total\_charge} C")



png

Charge: 1e-06 C, Location: (-0.0012184532929861245, -0.0004979964164912648)  
Charge: 1e-06 C, Location: (0.00048390202162626635, 0.0014723630957774528)  
Charge: 1e-06 C, Location: (-0.0006583745072223969, -0.00044763882490302283)  
Charge: 1e-06 C, Location: (0.0011431645429140196, 0.001931463822718301)  
Charge: 1e-06 C, Location: (-5.60897369906103e-05, -0.0003944506192641625)  
Charge: -1e-06 C, Location: (-0.0011296991527068485, 0.0017233177048307227)  
Total Charge: 4e-06 C

## Gauss’s Law

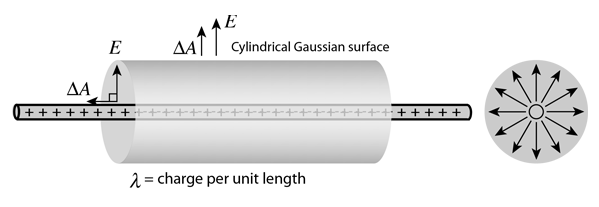
Clearly Coulomb’s Law gives us a prescription to solve most problems and serves as the basis for generic approaches to solving these problems. However, we can use the [divergence theorem](https://en.wikipedia.org/wiki/Divergence_theorem) with the first of the electrostatic equations above. This will convert this equation into a tool for solving highly symmetric problems.

The left hand side can be rewritten as:

where is an infinitesimal area element that has a vector direction. It points outward from the surface (normal to the local surface). We use this to write [Gauss’s Law](https://en.wikipedia.org/wiki/Gauss%27s_law) for the electric field:

Notice that this expression is always true, but it almost never useful for solving problems in general. The electric flux is often easily calculated considering the total enclosed charge:

However, using that measure of the total electric flux to find the field is only possible in highly symmetric situations. For example, if we have a spherical charge distribution , we can use Gauss’s Law to find the electric field at any point outside the sphere. Below, is a figure of a cylindrical charge distribution. We can use Gauss’s Law to find the electric field at any point outside the cylinder because the charge per unit length is constant.

**[Image not embedded: remote images are not included in PDF export. Check the original file for the image.]** 

### A charged cylinder

**✅ Do this**

Consider a long and thin plastic cylinder with a radius of and length , such that . The cylinder has a charge distribution that varies with radius , where is a positive constant. Find the electric field inside and outside the cylinder using Gauss’s Law.

* Sketch the situation. What are the units of ?
* Prove to yourself and your group that you can use Gauss’s Law to find the electric field inside and outside the cylinder (why? what assumptions are you making?)
* Compute the electric field and graph it’s magnitude as a function of for .

### Invent a planar problem

**✅ Do this**

It is very common to show that the electric field on a planar surface is given by:

where is the charge density on the surface (typically, ). We can use Gauss’s Law to find the electric field due to a charged plane. But, we want you to invent a problem for a thick plane of charge that has a varying charge density that depends on the distance from the center of the plane.

1. Invent this problem and sketch it. Note here: will still allow you to use Gauss’s Law.
2. Solve for the electric field using Gauss’s Law inside and outside the thick plane.

## Numerical Superposition - Coulomb’s Law

We gave you code and a class structure that lets you develop solutions to 2D electrostatic problems. In fact, we built a simple numerical integrator that uses superposition in 2D to solve for the electric field. This works because the electric field obeys superposition. So we can add up the electric field due to each charge.

In the continuous limit (), this becomes an integral that we can solve (sometimes). It depends on our ability to express the results in integrable functions:

However, the scheme to numerically integrate this expression to find the field follows this prescription. And as long as we avoid singularities and the sources are compact (locally distributed), we simply add the contributions numerically. We can use smaller and smaller chunks of the continuous distribution to get better approximations.

where is electric field of the ith point charge with charge . That is effectively what our 2D electric field visualization code does. It is a numerical integrator that uses superposition to find the electric field.

**✅ Do this**

* Abstract our code to 3D. This will be a usueful tool for you to use in the future.
* Create known 3D distributions of charges and demonstrate you can view the field in 3D.
  + Note the visualizations will have to change more than the code to create the charges and compute their fields.

## Your code here