Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Modelación computacional de sistemas eléctricos

Felipe Gerardo Ramón Fox

17 de Abr, 2024

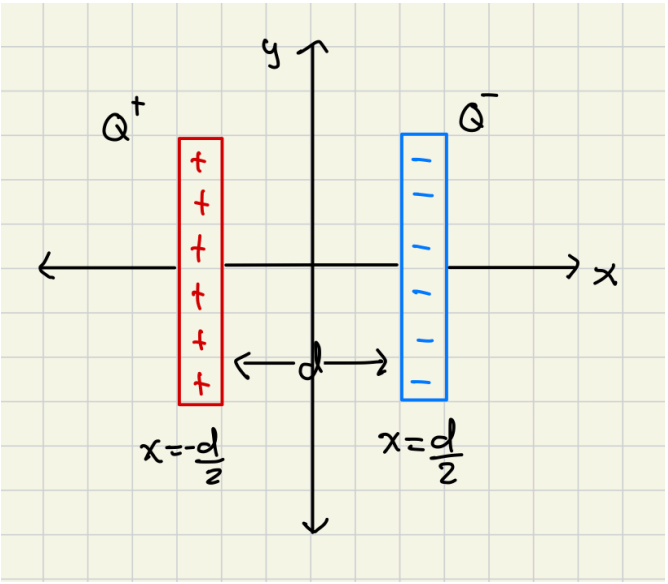
“Reto: Etapa 3”

Estudiantes:

Angel Orlando Anguiano Peña A00838418
Juan Luis Alvarez Cisneros A01643154
Silvanna Farías A01178494

Planteamiento

Se tienen dos placas de carga +Q y -Q. Vamos a ubicar la carga positiva del lado izquierdo y la carga negativa del lado derecho de modo que haya una separación “d” entre ellas. El eje y, y el orígen del sistema de coordenadas están exactamente a la mitad de la distancia entre las placas de modo que las coordenadas x de las placas son las siguientes: x = -d/2 para la carga positiva y x = d/2 para la carga negativa.



Modelo del dipolo

Para determinar el campo eléctrico de un dipolo en un punto (x,y), podemos calcular de manera individual el campo eléctrico de cada una de las placas para al final hacer la suma de ambos. Pues el campo total de un sistema de cargas es representado por la suma de los campos eléctricos correspondientes a cada una de las cargas en el sistema.

Por ejemplo:

$$\vec{E}_{Total} = \vec{E}_{+} + \vec{E}_{-}$$

$$\vec{E}_{Total} = \frac{K q^{+}}{r^2}\hat{r} + \frac{K q^{-}}{r^2}\hat{r}$$

Gráfica del campo vectorial en función de (x, y) y asumiendo una carga Q = 1 C y una separación d = 1 m en una región de 2.0 x 2.0 metros.

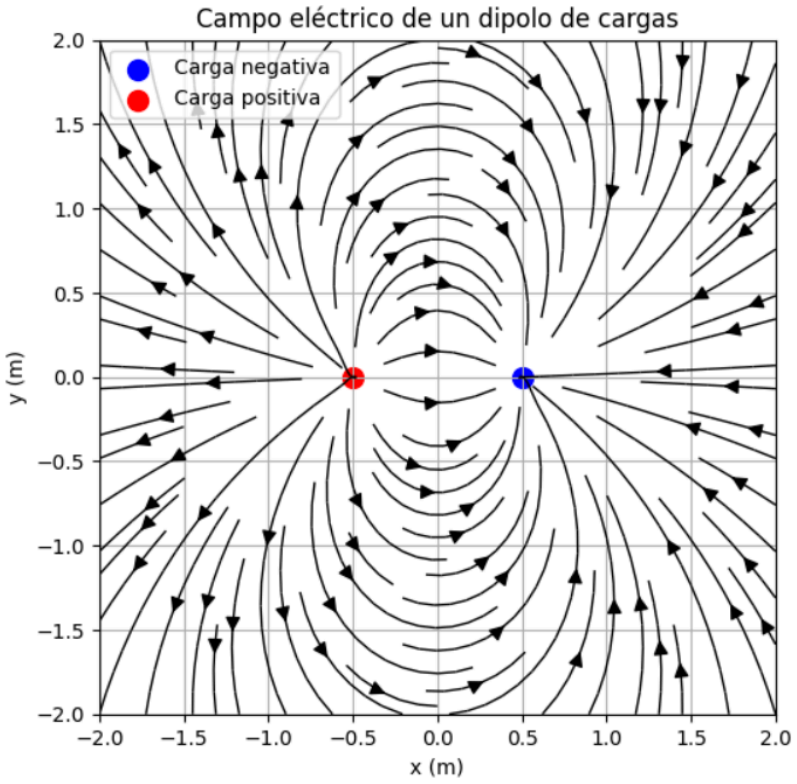
Autores: Angel Orlando Anguiano Peña A00838418, Juan Luis Alvarez Cisneros A01643154 y Silvanna Farías A01178494

Propósito: Generar una gráfica del campo vectorial en función de (x, y) y asumiendo una carga Q = 1 C y una separación d = 1 m en una región de 2.0 x 2.0 metros.

Ejecución: En lenguaje python con el uso de funciones y librerias.

Paqueterías requeridas: numpy, matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

# Parametros de entrada
k = 8.9875e9  # Constante de Coulomb en (N*m^2) / C^2
q1 = -1  # Carga 1 en Coulombs
q2 = 1  # Carga 2 en Coulombs
x1, y1 = 0.5, 0  # Posición de la carga 1
x2, y2 = -0.5, 0  # Posición de la carga 2

# Función para calcular el campo eléctrico en un punto (x, y)
"""
Parámetros :
- x (float): Posición en el eje x en (m)
- y (float): Posición en el eje y en (m)

Varibles de salida:
- Ex (float): Componente x del campo eléctrico en (N/C)
- Ey (float): Componente y del campo eléctrico en (N/C)
"""
def campo_electrico(x, y):
    r1 = np.sqrt((x - x1)**2 + (y - y1)**2)  # Distancia de la carga 1 al punto
    r2 = np.sqrt((x - x2)**2 + (y - y2)**2)  # Distancia de la carga 2 al punto

    # Componentes del campo eléctrico debido a cada carga
    Ex1 = k * q1 * (x - x1) / r1**3
    Ey1 = k * q1 * (y - y1) / r1**3
    Ex2 = k * q2 * (x - x2) / r2**3
    Ey2 = k * q2 * (y - y2) / r2**3

    # Sumamos las contribuciones de ambas cargas
    Ex = Ex1 + Ex2
    Ey = Ey1 + Ey2

    return Ex, Ey

# Creamos una malla de puntos para evaluar el campo eléctrico
x = np.linspace(-2, 2, 100)
y = np.linspace(-2, 2, 100)
X, Y = np.meshgrid(x, y)

# Cálculo del campo eléctrico en cada punto de la malla
Ex, Ey = campo_electrico(X, Y)

# Graficar el campo eléctrico
plt.figure(figsize=(8, 6))
plt.streamplot(X, Y, Ex, Ey, color='black', linewidth=1, arrowsize=1.5)
plt.scatter(x1, y1, color='blue', s=100, label='Carga negativa')
plt.scatter(x2, y2, color='red', s=100, label='Carga positiva')
plt.xlabel('x (m)')
plt.ylabel('y (m)')
plt.title('Campo eléctrico de un dipolo de cargas')
plt.xlim(-2, 2)
plt.ylim(-2, 2)
plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)
plt.legend()
plt.show()
```



Interpretación de la gráfica:

Gracias a la actividad “Tutorial: El dipolo”, realizada anteriormente, nos fue más fácil llevar a cabo el desarrollo del código que representa esta gráfica, pues solo constó de afinar detalles como la nueva posición de las cargas y su magnitud. Hasta este momento, las cargas del dipolo siguen siendo representadas en forma de círculos.

Modelo de Línea de Cargas

Usamos el campo eléctrico en forma: $E = \frac{ke(r-r')}{|r-r'|^3}$ para plantear el campo eléctrico que generan todas las cargas de la siguiente manera:

1. Definiendo las posiciones de las cargas y sus valores
2. Calcular el campo eléctrico generado por cada carga en el punto dado
3. Sumar los campos eléctricos generados por todas las cargas para obtener el campo eléctrico total en el punto

Gráfica del campo vectorial en función de (x, y) asumiendo que Q = 1.0 C, L = 1.0 m y d = 1.0 m y dividimos las líneas de carga en N = 5 en una región de 2.0 x 2.0 metros.

Autores: Angel Orlando Anguiano Peña A00838418, Juan Luis Alvarez Cisneros A01643154 y Silvanna Farías A01178494

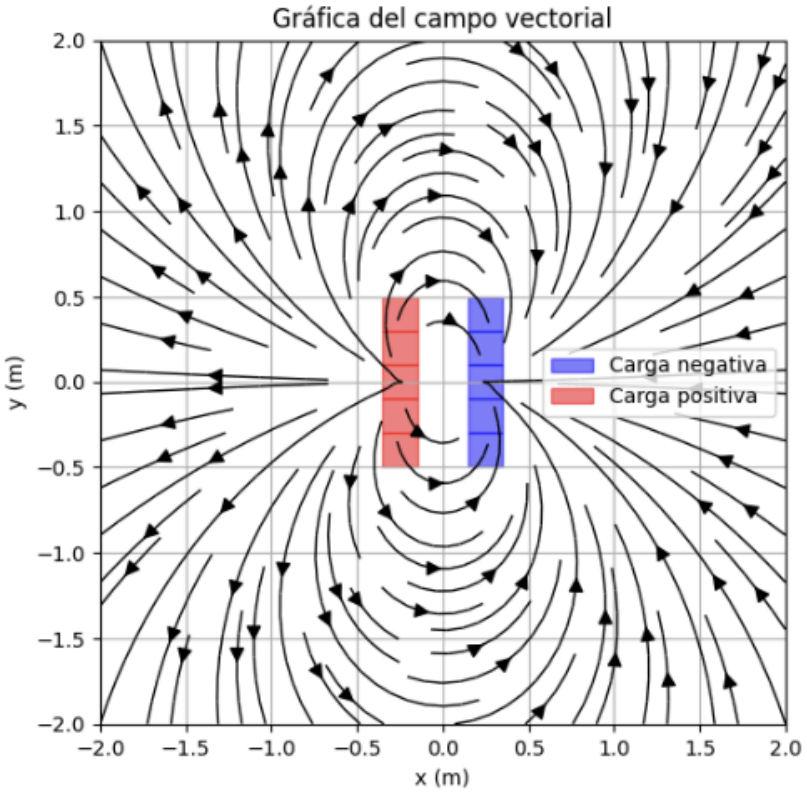
Propósito: Generar una gráfica del campo vectorial en función de (x, y) asumiendo que Q = 1.0 C, L = 1.0 m y d = 1.0 m y dividimos las líneas de carga en N = 5 en una región de 2.0 x 2.0 metros.

Ejecución: En lenguaje python con el uso de funciones y librerias.

Paqueterías requeridas: numpy, matplotlib

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle

# Parametros de entrada
k = 8.9875e9  # Constante de Coulomb en (N*m^2) / C^2
q1 = -1  # Carga 1 en Coulombs
q2 = 1  # Carga 2 en Coulombs
x1, y1 = 0.25, 0  # Posición de la carga 1
x2, y2 = -0.25, 0  # Posición de la carga 2

# Función para calcular el campo eléctrico en un punto (x, y)
"""
Parámetros :
- x (float): Posición en el eje x en (m)
- y (float): Posición en el eje y en (m)

Varibles de salida:
- Ex (float): Componente x del campo eléctrico en (N/C)
- Ey (float): Componente y del campo eléctrico en (N/C)
"""
def campo_electrico(x, y):
    r1 = np.sqrt((x - x1)**2 + (y - y1)**2)  # Distancia de la carga 1 al punto
    r2 = np.sqrt((x - x2)**2 + (y - y2)**2)  # Distancia de la carga 2 al punto

    # Componentes del campo eléctrico debido a cada carga
    Ex1 = k * q1 * (x - x1) / r1**3
    Ey1 = k * q1 * (y - y1) / r1**3
    Ex2 = k * q2 * (x - x2) / r2**3
    Ey2 = k * q2 * (y - y2) / r2**3

    # Sumamos las contribuciones de ambas cargas
    Ex = Ex1 + Ex2
    Ey = Ey1 + Ey2

    return Ex, Ey
# Creamos una malla de puntos para evaluar el campo eléctrico
x = np.linspace(-2, 2, 100)
y = np.linspace(-2, 2, 100)
X, Y = np.meshgrid(x, y)

# Cálculo del campo eléctrico en cada punto de la malla
Ex, Ey = campo_electrico(X, Y)

# Graficar el campo eléctrico
plt.figure(figsize=(8, 6))
plt.streamplot(X, Y, Ex, Ey, color='black', linewidth=1, arrowsize=1.5)
plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - -0.3), 0.2, 0.2, color='blue', alpha=0.5, label='Carga negativa'))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - -0.3), 0.2, 0.2, color='red', alpha=0.5, label='Carga positiva'))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - -0.1), 0.2, 0.2, color='blue', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - -0.1), 0.2, 0.2, color='red', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 0.1), 0.2, 0.2, color='blue', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 0.1), 0.2, 0.2, color='red', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 0.3), 0.2, 0.2, color='blue', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 0.3), 0.2, 0.2, color='red', alpha=0.5))
```

```
plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 0.5), 0.2, 0.2, color='blue', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 0.5), 0.2, 0.2, color='red', alpha=0.5))

plt.xlabel('x (m)')
plt.ylabel('y (m)')
plt.title('Gráfica del campo vectorial')
plt.xlim(-2, 2)
plt.ylim(-2, 2)
plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)
plt.legend()
plt.show()
```



Interpretación de la gráfica:

Podemos observar que ahora nuestras cargas que formaban al dipolo se han convertido en las placas que, si bien, han sido divididas en cinco partes según las indicaciones del entregable, cada una con sus respectivas características como coordenadas en el eje Y, pues para una placa, todas sus subdivisiones tendrán la misma coordenada en el eje X, campo eléctrico, etc.

Modelo de Placas de Cargas

¿Cómo se calcula la posición de cada elemento de carga y cuál sería una estructura de datos adecuada para almacenarlas?
Podemos iterar por todas las placas del dipolo, ya que están divididas en diez partes por iguales, creamos el ciclo for que es el que se encarga de ir por cada una de las placas. Podemos hacerlo dividiendo la longitud de las placas (L/2).

¿Cómo se utilizará el principio de superposición y la ecuación de campo eléctrico para evaluar el campo eléctrico en una posición dada?

Utilizando el principio de superposición y la ecuación de campo eléctrico evaluamos el campo eléctrico en una posición dada de la siguiente manera:

1. Calcular la magnitud y el vector de posición, para cada carga
2. Calcular la distancia desde la carga hasta el punto donde se evalúa el campo
3. Calcular el vector unitario
4. Con la ecuación del campo eléctrico para calcular el campo eléctrico de cada carga
5. Calcular el campo eléctrico total sumando todos los campos eléctricos

Pseudocódigo

Importar librerías
- numpy como np
- matplotlib.pyplot como plt
- Rectangle de matplotlib.patches

Definir parámetros de entrada
- k como 8.9875e9 (constante de Coulomb)
- q1 como -1 (carga negativa en Coulombs)
- q2 como 1 (carga positiva en Coulombs)
- L como 1 (longitud de cada placa en metros)
- d como 1 (distancia entre las placas en metros)
- N como 10 (número de subdivisiones por lado de cada placa)

Definir función campo eléctrico con parámetros (x, y, z, cargas)
- Inicializar Ex, Ey, Ez a 0
- Para cada carga en cargas hacer:
  - Calcular diferencia en x, y, z (dx, dy, dz)
  - Calcular distancia r como raíz cuadrada de (dx^2 + dy^2 + dz^2)
  - Actualizar Ex, Ey, Ez usando la fórmula de campo eléctrico
- Devolver Ex, Ey, Ez

Generar posiciones de las cargas en las placas
- Inicializar lista cargas vacía
- Calcular incremento delta como L / N
- Para i de 0 a N hacer:
  - Para j de 0 a N hacer:
    - Calcular posiciones x1, y1 para carga positiva
    - Agregar (x1, y1, 0, q2) a cargas
    - Calcular posiciones x2, y2 para carga negativa
    - Agregar (x2, y2, 0, q1) a cargas

Crear una malla de puntos para evaluar el campo eléctrico
- Definir x, y usando linspace con rangos -2 a 2 y 100 puntos

- Fijar z a 0
- Crear malla X, Y usando meshgrid con x, y

Calcular el campo eléctrico en cada punto de la malla
- Inicializar matrices Ex, Ey, Ez con ceros del mismo tamaño que X
- Para cada punto (i, j) en la malla hacer:
  - Calcular Ex[i, j], Ey[i, j], Ez[i, j] usando función campo eléctrico

Graficar el campo eléctrico
- Configurar figura y gráfico de líneas de campo usando stream plot
- Agregar rectángulos para representar las placas con diferentes colores para cargas positivas y negativas
- Configurar etiquetas, títulos, límites, y aspectos del gráfico
- Mostrar gráfico

Fin del pseudocódigo

Gráfica del campo vectorial en función de (x, y) asumiendo que Q = 1.0 C, L = 1.0 m y d = 1.0 m y dividimos las placas en N = 10 elementos por lado. La gráfica es para una región de 2.0 x 2.0 m

Autores: Angel Orlando Anguiano Peña A00838418, Juan Luis Alvarez Cisneros A01643154 y Silvanna Farías A01178494

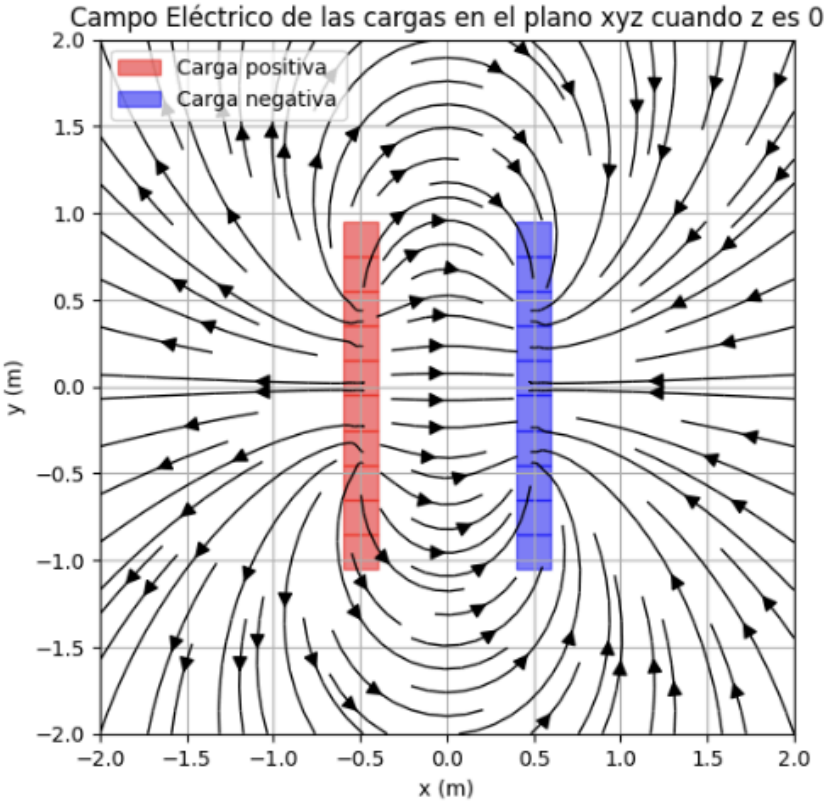
Propósito: Generar una gráfica del campo vectorial en función de (x, y) asumiendo que Q = 1.0 C, L = 1.0 m y d = 1.0 m y dividimos las placas en N = 10 elementos por lado. La gráfica es para una región de 2.0 x 2.0 m

Ejecución: En lenguaje python con el uso de funciones y librerias.

Paqueterías requeridas: numpy, matplotlib, matplotlib.patches

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle

# Parámetros de entrada
k = 8.9875e9  # Constante de Coulomb en (N*m^2) / C^2
q1 = -1  # Carga 1 en Coulombs (Carga negativa)
q2 = 1   # Carga 2 en Coulombs (Carga positiva)
L = 1   # Longitud de cada placa en metros
d = 1  # Distancia entre las placas en metros
N = 10   # Número de subdivisiones por lado de cada placa

# Función para calcular el campo eléctrico en un punto (x, y, z)
"""
Parámetros :
- x (float): Posición en el eje x en (m)
- y (float): Posición en el eje y en (m)
- z (float): Posición en el eje z en (m)

Varibles de salida:
- Ex (float): Componente x del campo eléctrico en (N/C)
- Ey (float): Componente y del campo eléctrico en (N/C)
- Ez (float): Componente z del campo eléctrico en (N/C)
"""
def campo_electrico(x, y, z, cargas):
    Ex, Ey, Ez = 0, 0, 0
    for carga in cargas:
        dx = x - carga[0]
        dy = y - carga[1]
        dz = z - carga[2]
        r = np.sqrt(dx**2 + dy**2 + dz**2)
        Ex += k * carga[3] * dx / r**3
        Ey += k * carga[3] * dy / r**3
        Ez += k * carga[3] * dz / r**3
    return Ex, Ey, Ez

# Generar posiciones de las cargas en las placas
cargas = []
delta = L / N
for i in range(N):
    for j in range(N):
        # Carga positiva en la placa a la izquierda
        x1 = -0.5
        y1 = -L / 2 + delta * (j + 0.5)
        cargas.append((x1, y1, 0, q2))
        # Carga negativa en la placa a la derecha
        x2 = 0.5
        y2 = -L / 2 + delta * (i + 0.5)
        cargas.append((x2, y2, 0, q1))
```

```
  # Crear una malla de puntos para evaluar el campo eléctrico
  x = np.linspace(-2, 2, 100)
  y = np.linspace(-2, 2, 100)
  z = 0  # Se fija z en 0 para el plano xy
  X, Y = np.meshgrid(x, y)

  # Calcular el campo eléctrico en cada punto de la malla
  Ex, Ey, Ez = np.zeros_like(X), np.zeros_like(Y), np.zeros_like(X)
  for i in range(X.shape[0]):
      for j in range(Y.shape[1]):
          Ex[i, j], Ey[i, j], Ez[i, j] = campo_electrico(X[i, j], Y[i, j], z, cargas)

  # Graficar el campo eléctrico
  plt.figure(figsize=(8, 6))
  plt.streamplot(X, Y, Ex, Ey, color='black', linewidth=1, arrowsize=1.5)



# Agregar las placas al gráfico
plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - -0.3), 0.2, 0.2, color='red', alpha=0.5, label='Carga positiva'))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - -0.3), 0.2, 0.2, color='blue', alpha=0.5, label='Carga negativa'))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - -0.1), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - -0.1), 0.2, 0.2, color='blue', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 0.1), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 0.1), 0.2, 0.2, color='blue', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 0.3), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 0.3), 0.2, 0.2, color='blue', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 0.5), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 0.5), 0.2, 0.2, color='blue', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 0.7), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 0.7), 0.2, 0.2, color='blue', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 0.9), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 0.9), 0.2, 0.2, color='blue', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 1.1), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 1.1), 0.2, 0.2, color='blue', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 1.3), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 1.3), 0.2, 0.2, color='blue', alpha=0.5))

plt.gca().add_patch(Rectangle((x1 - 0.1, y1 - 1.5), 0.2, 0.2, color='red', alpha=0.5))
plt.gca().add_patch(Rectangle((x2 - 0.1, y2 - 1.5), 0.2, 0.2, color='blue', alpha=0.5))


plt.xlabel('x (m)')
plt.ylabel('y (m)')
plt.title('Campo Eléctrico de las cargas en el plano xyz cuando z es 0')
plt.xlim(-2, 2)
plt.ylim(-2, 2)
plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)
plt.legend()
plt.show()
```



Interpretación de la gráfica:
Tenemos una gráfica tridimensional vista desde arriba, donde los valores de X permanecen constantes, mientras que los de Y y Z van cambiando debido a que la placa tiene una longitud N*N, cada recuadro de las placas representa una coordenada. Podemos observar que las líneas de campo eléctrico fluyen en dirección de la carga positiva a la negativa y ambas tienen una separación de 1 metro. Cabe mencionar que cada pequeño cuadro tiene la misma carga que cualquier otro. Ésta está distribuída uniformemente por las placas, ambas placas tienen la misma magnitud de carga, solo que con signo contrario.