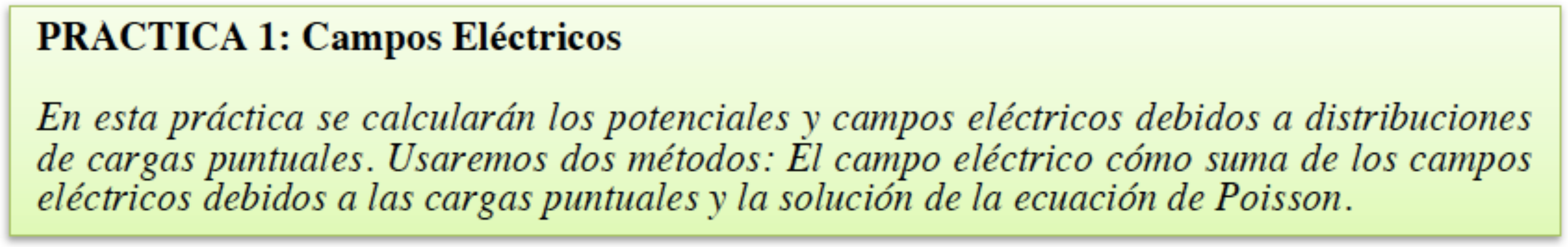
Practicas Electromagnetismo I

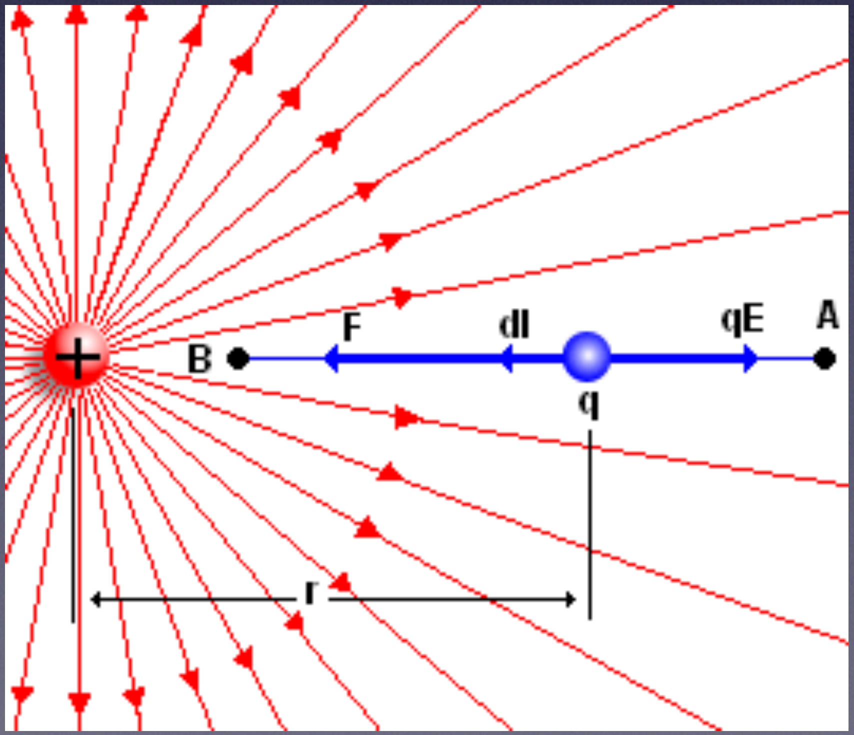
Campos eléctricos



PRACTICA 1: Campos Eléctricos

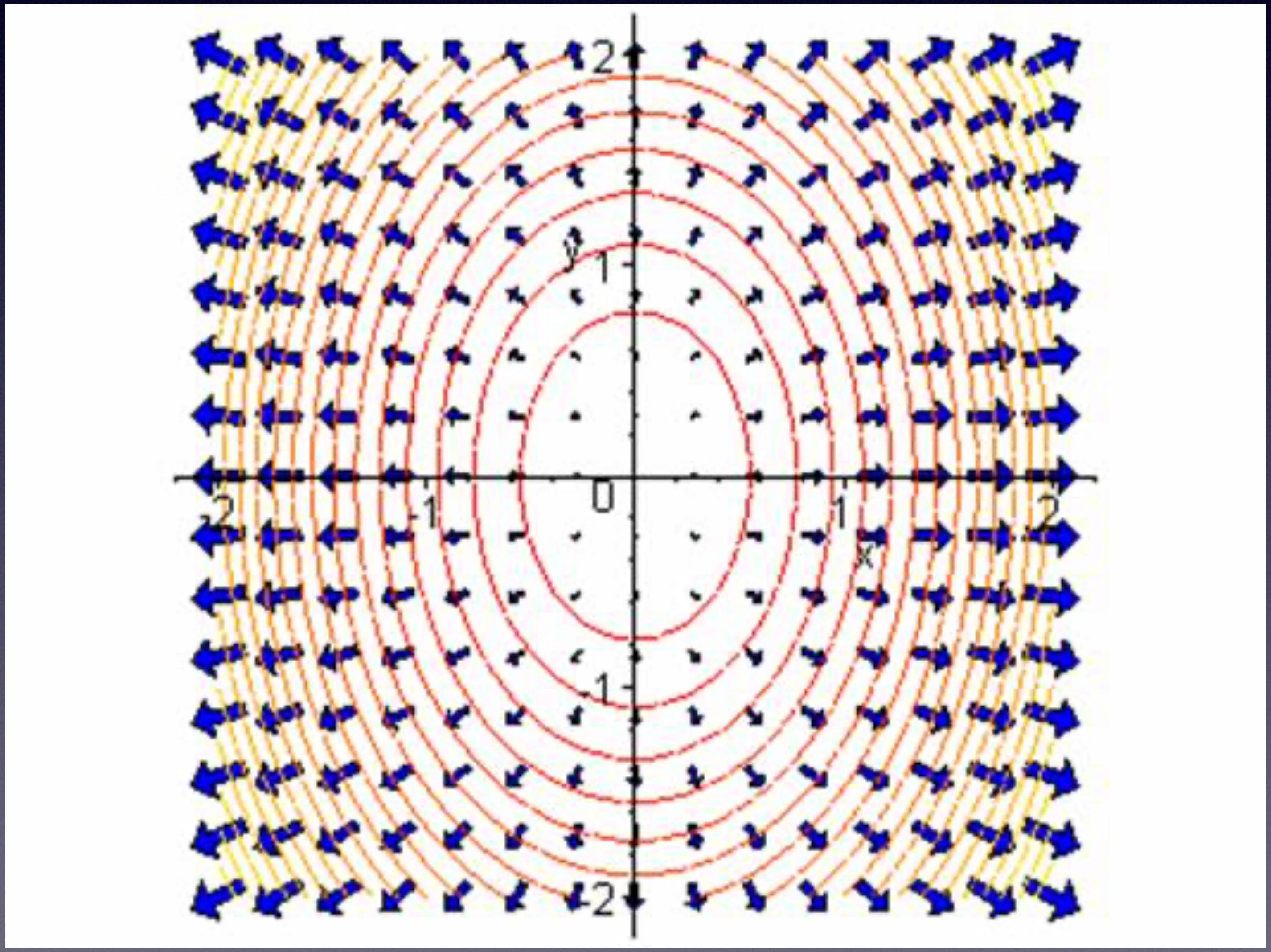
En esta práctica se calcularán los potenciales y campos eléctricos debidos a distribuciones de cargas puntuales. Usaremos dos métodos: El campo eléctrico cómo suma de los campos eléctricos debidos a las cargas puntuales y la solución de la ecuación de Poisson.

El **potencial eléctrico** en un punto, es el trabajo a realizar por unidad de carga para mover dicha carga dentro de un campo electrostático desde el punto de referencia hasta el punto considerado

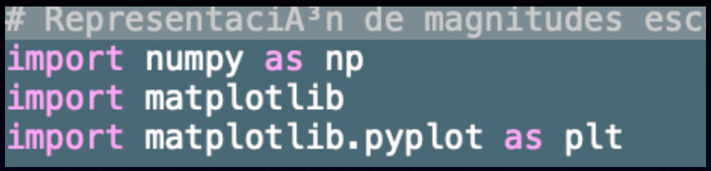


$$\boxed{V = \frac{1}{4\pi\epsilon}\frac{q_0}{r}}$$

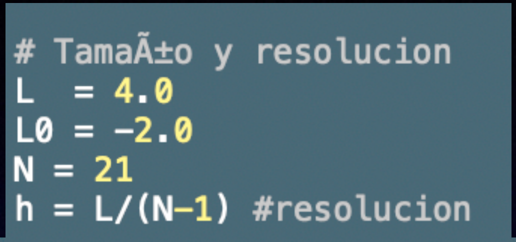
El **campo eléctrico** (región del espacio en la que interactúa la fuerza eléctrica) es un campo físico que se representa por medio de un modelo que describe la interacción entre cuerpos y sistemas con propiedades de naturaleza eléctrica.[1] Se puede describir como un campo vectorial en el cual una carga eléctrica puntual de valor q sufre los efectos de una fuerza eléctrica F



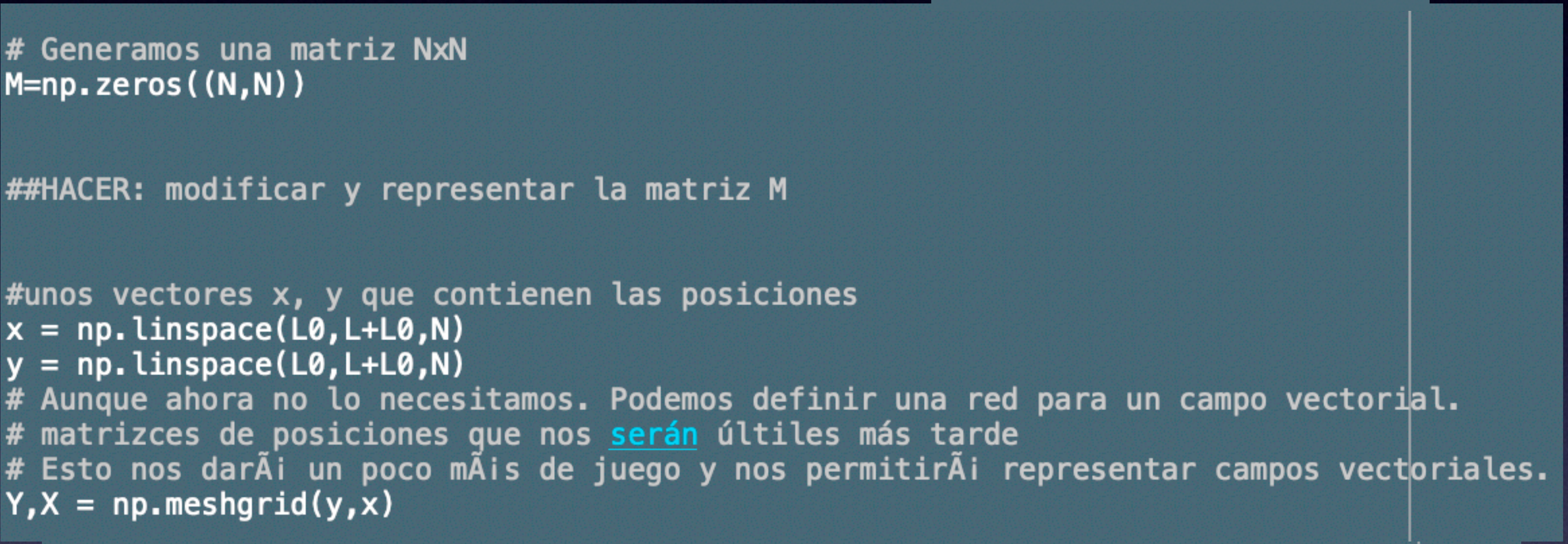
Intro: Representación de magnitudes escalares y vectoriales



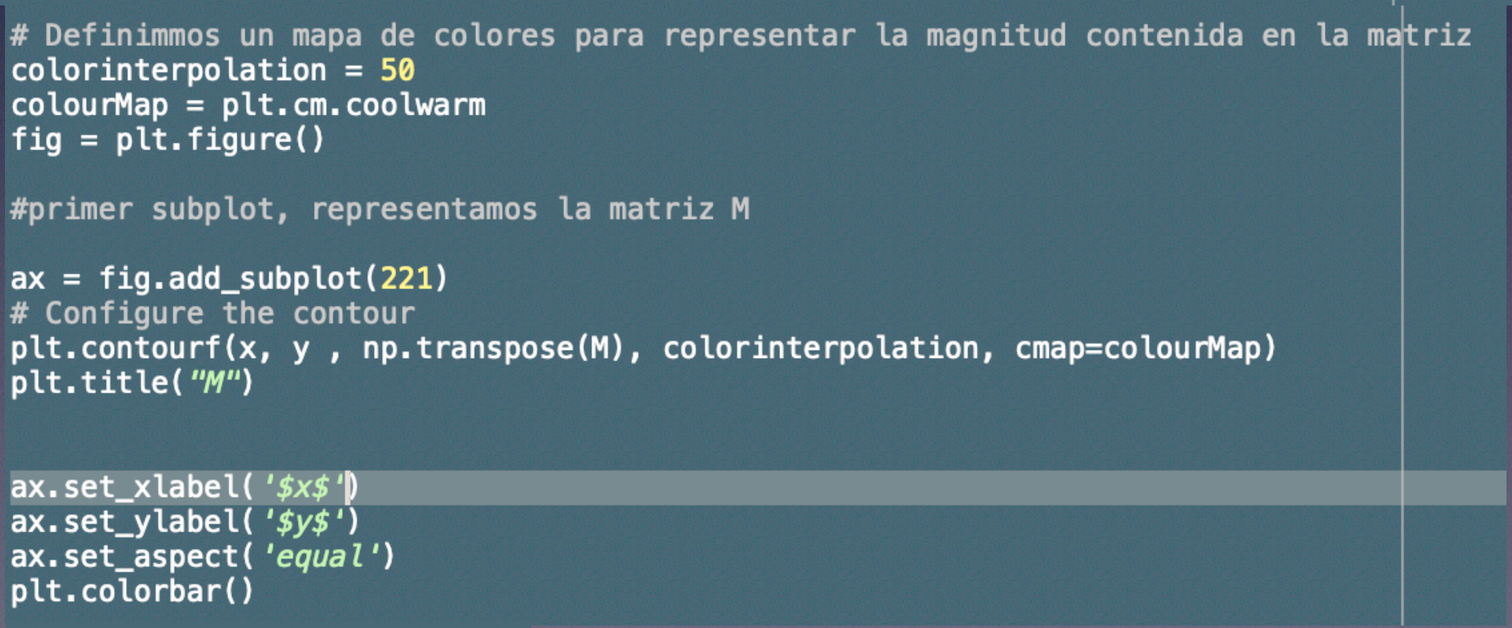
```
# RepresentaciA³n de magnitudes esc
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
```



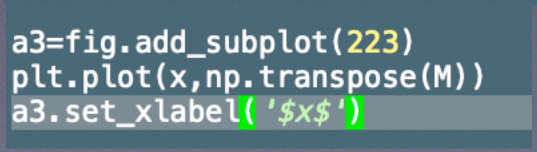
```
# TamaÃ±o y resolucion
L  = 4.0
L0 = -2.0
N = 21
h = L/(N-1) #resolucion
```



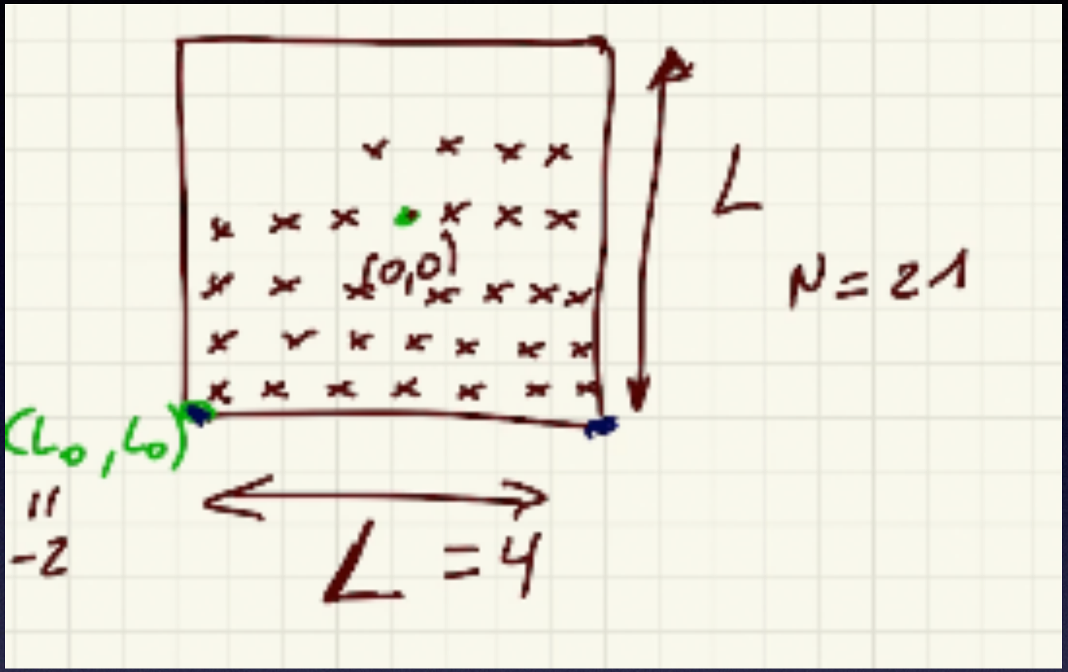
```
# Generamos una matriz NxN
M=np.zeros((N,N))


##HACER: modificar y representar la matriz M


#unos vectores x, y que contienen las posiciones
x = np.linspace(L0,L+L0,N)
y = np.linspace(L0,L+L0,N)
# Aunque ahora no lo necesitamos. Podemos definir una red para un campo vectorial.
# matrizces de posiciones que nos serán últiles más tarde
# Esto nos darÃi un poco mÃis de juego y nos permitirÃi representar campos vectoriales.
Y,X = np.meshgrid(y,x)
```

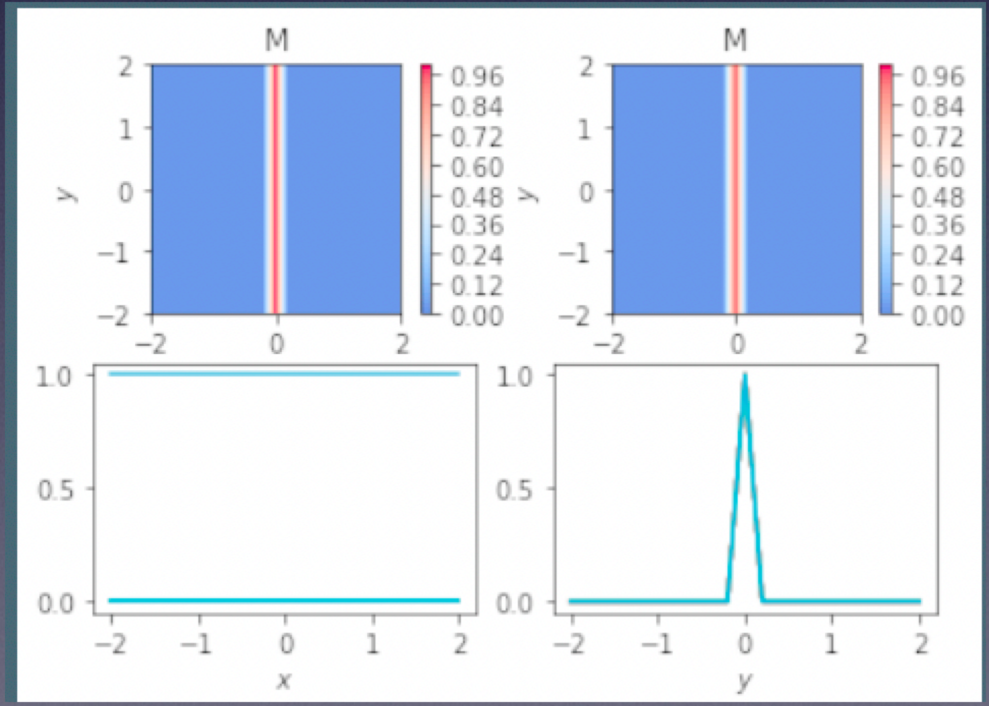


```
# Definimmos un mapa de colores para representar la magnitud contenida en la matriz
colorinterpolation = 50
colourMap = plt.cm.coolwarm
fig = plt.figure()

#primer subplot, representamos la matriz M

ax = fig.add_subplot(221)
# Configure the contour
plt.contourf(x, y , np.transpose(M), colorinterpolation, cmap=colourMap)
plt.title("M")


ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
ax.set_aspect('equal')
plt.colorbar()
```

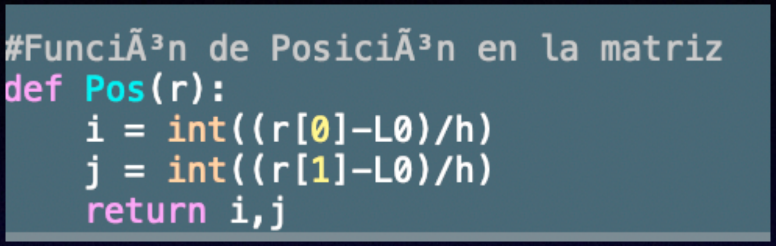


```
a3=fig.add_subplot(223)
plt.plot(x,np.transpose(M))
a3.set_xlabel('$x$')
```

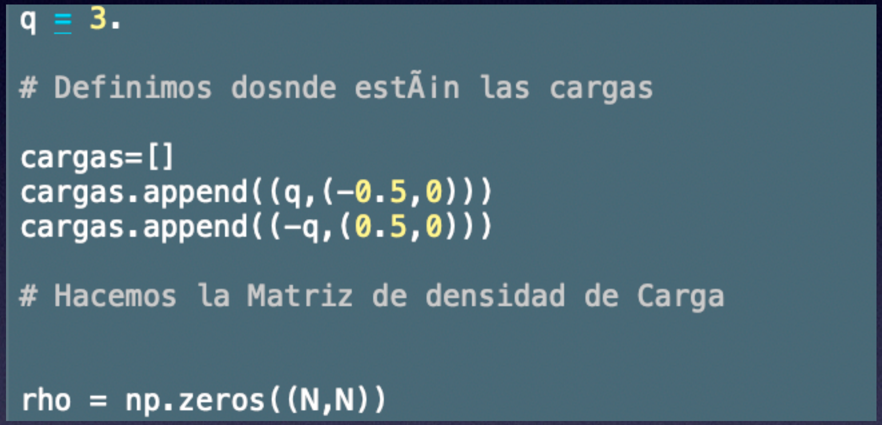




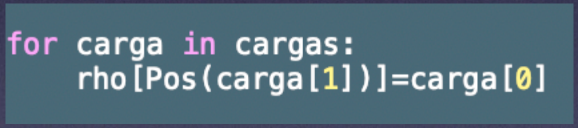
1. Definimos y representamos la matriz de densidad de carga



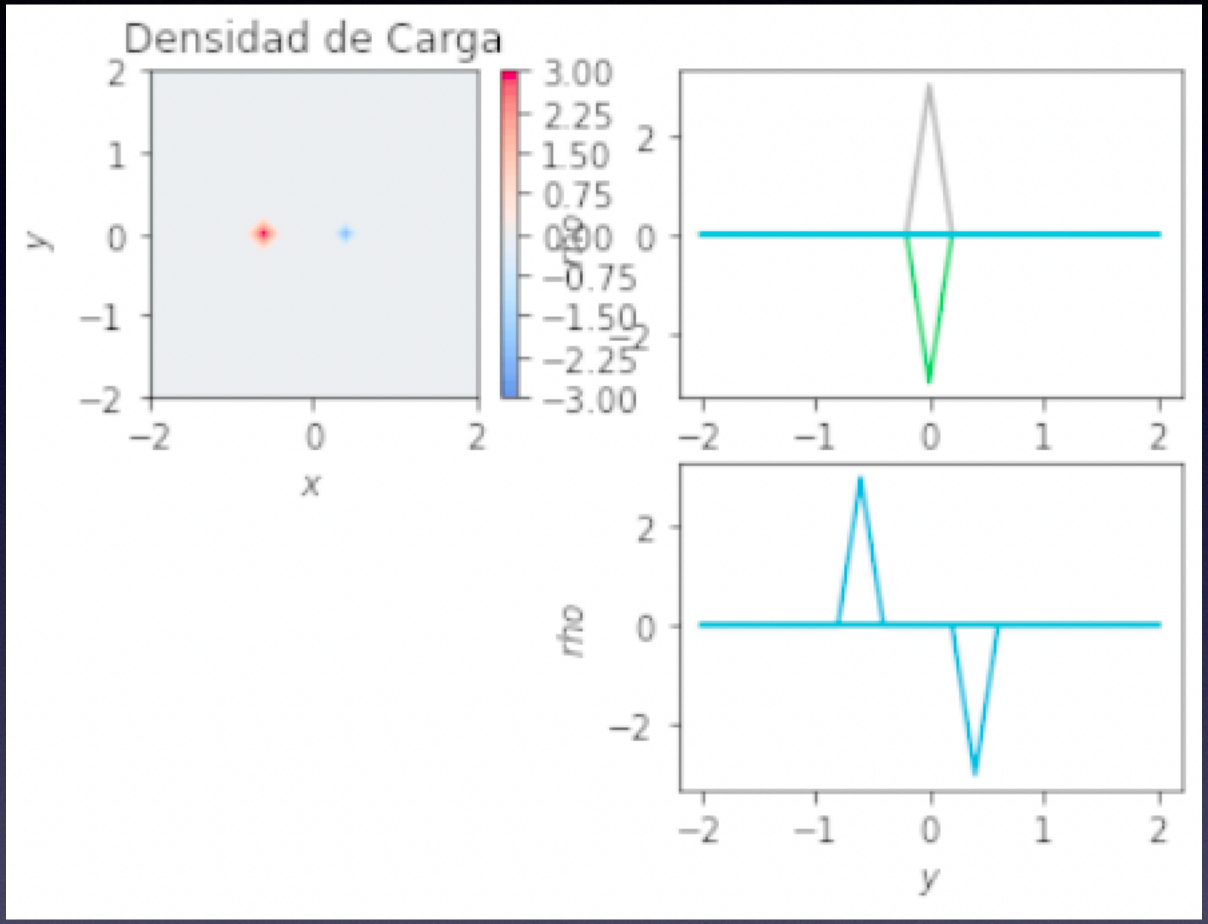
```
#FunciÃ³n de PosiciÃ³n en la matriz
def Pos(r):
    i = int((r[0]-L0)/h)
    j = int((r[1]-L0)/h)
    return i,j
```

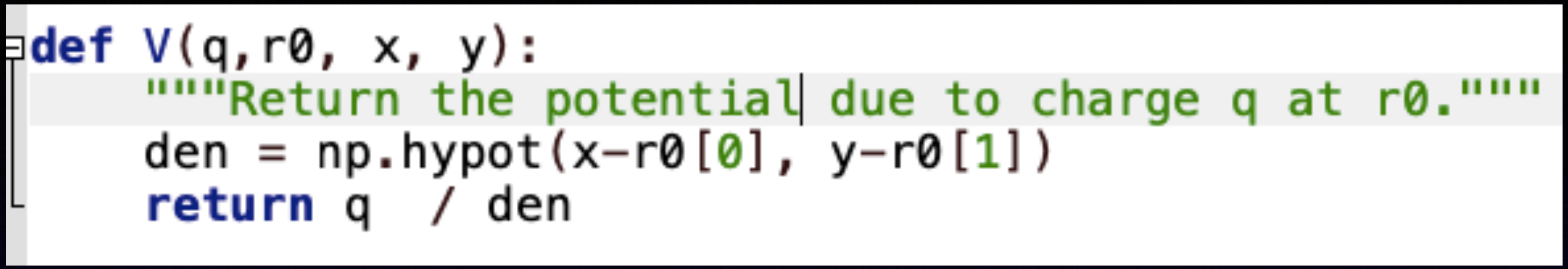


```
q = 3.

# Definimos dosnde estÃ¡n las cargas

cargas=[]
cargas.append((q,(-0.5,0)))
cargas.append((-q,(0.5,0)))

# Hacemos la Matriz de densidad de Carga


rho = np.zeros((N,N))
```

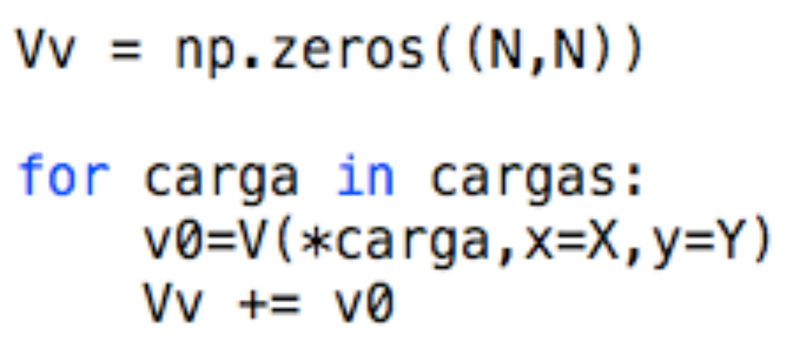


```
for carga in cargas:
    rho[Pos(carga[1])]=carga[0]
```

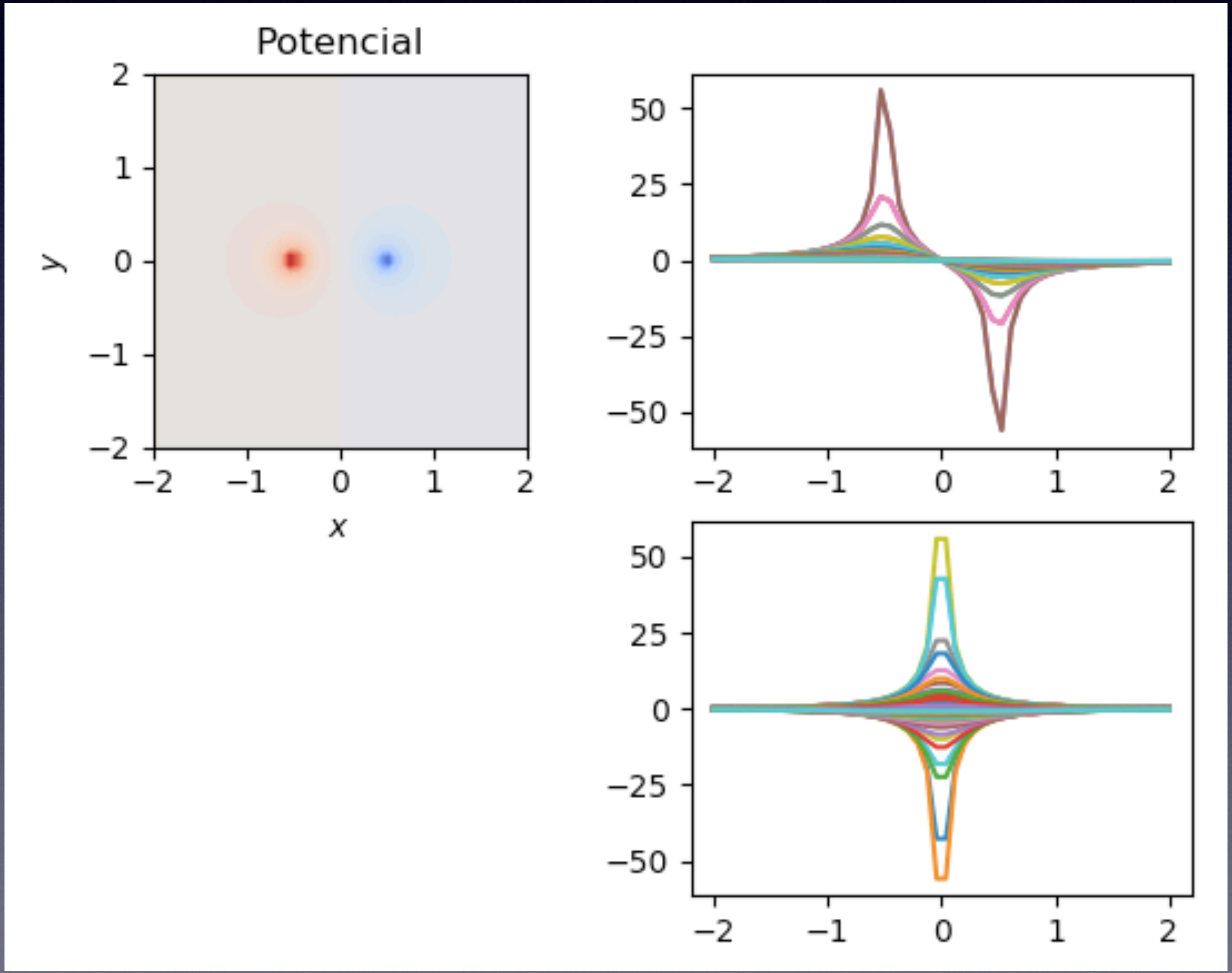


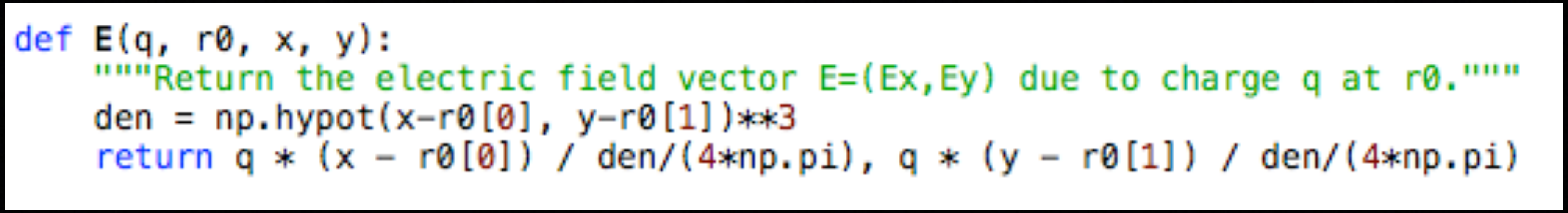


```python
def V(q,r0, x, y):
    """Return the potential due to charge q at r0."""
    den = np.hypot(x-r0[0], y-r0[1])
    return q  / den
```

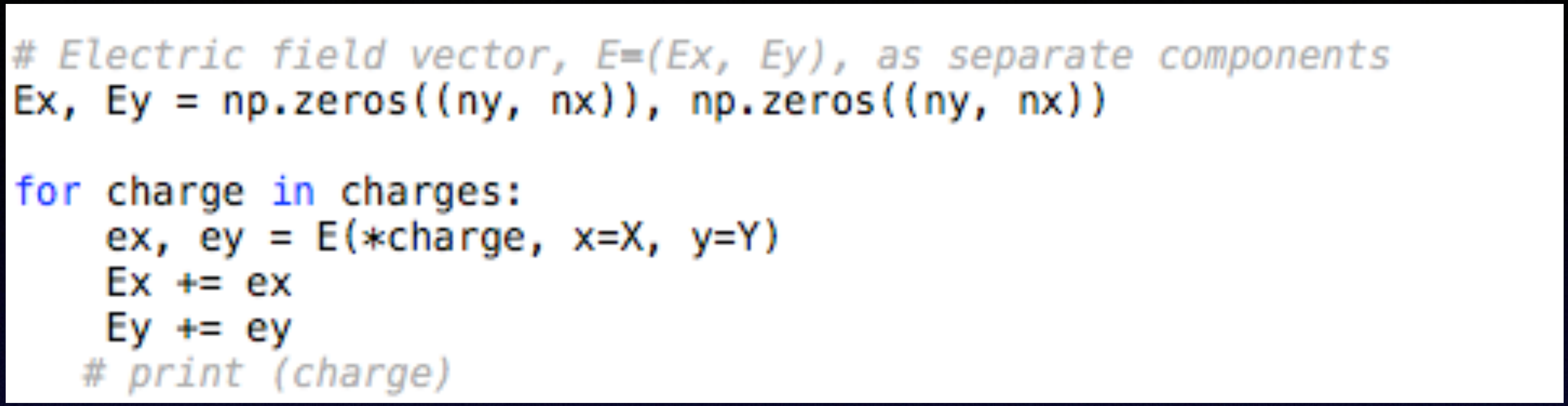


```python
Vv = np.zeros((N,N))

for carga in cargas:
    v0=V(*carga,x=X,y=Y)
    Vv += v0
```

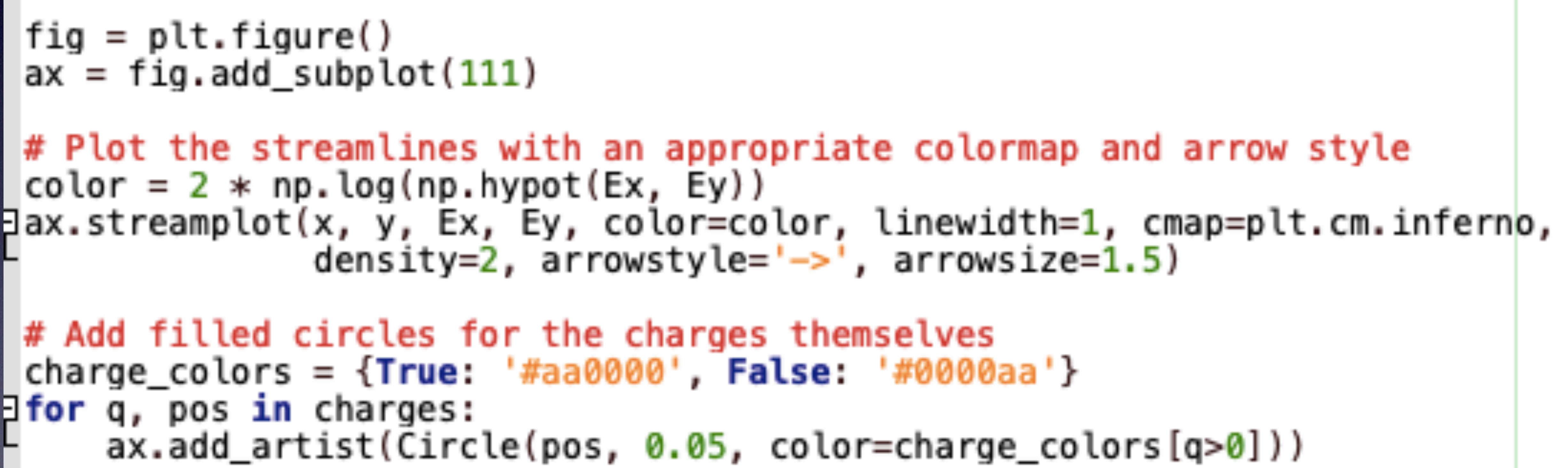




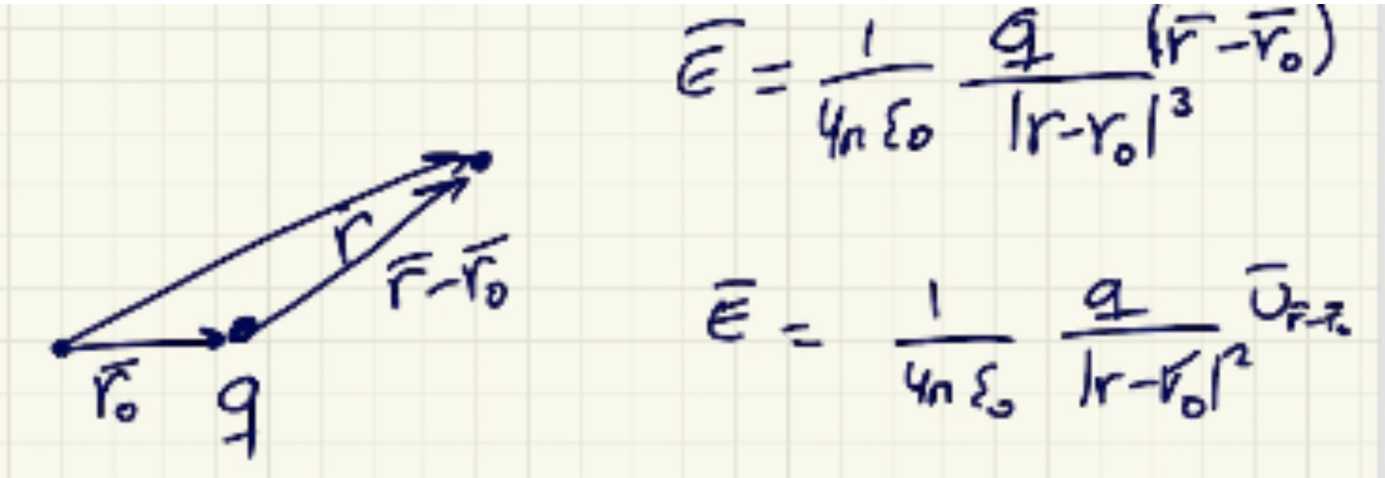
```
def E(q, r0, x, y):
    """Return the electric field vector E=(Ex,Ey) due to charge q at r0."""
    den = np.hypot(x-r0[0], y-r0[1])**3
    return q * (x - r0[0]) / den/(4*np.pi), q * (y - r0[1]) / den/(4*np.pi)
```



```
# Electric field vector, E=(Ex, Ey), as separate components
Ex, Ey = np.zeros((ny, nx)), np.zeros((ny, nx))

for charge in charges:
    ex, ey = E(*charge, x=X, y=Y)
    Ex += ex
    Ey += ey
  # print (charge)
```

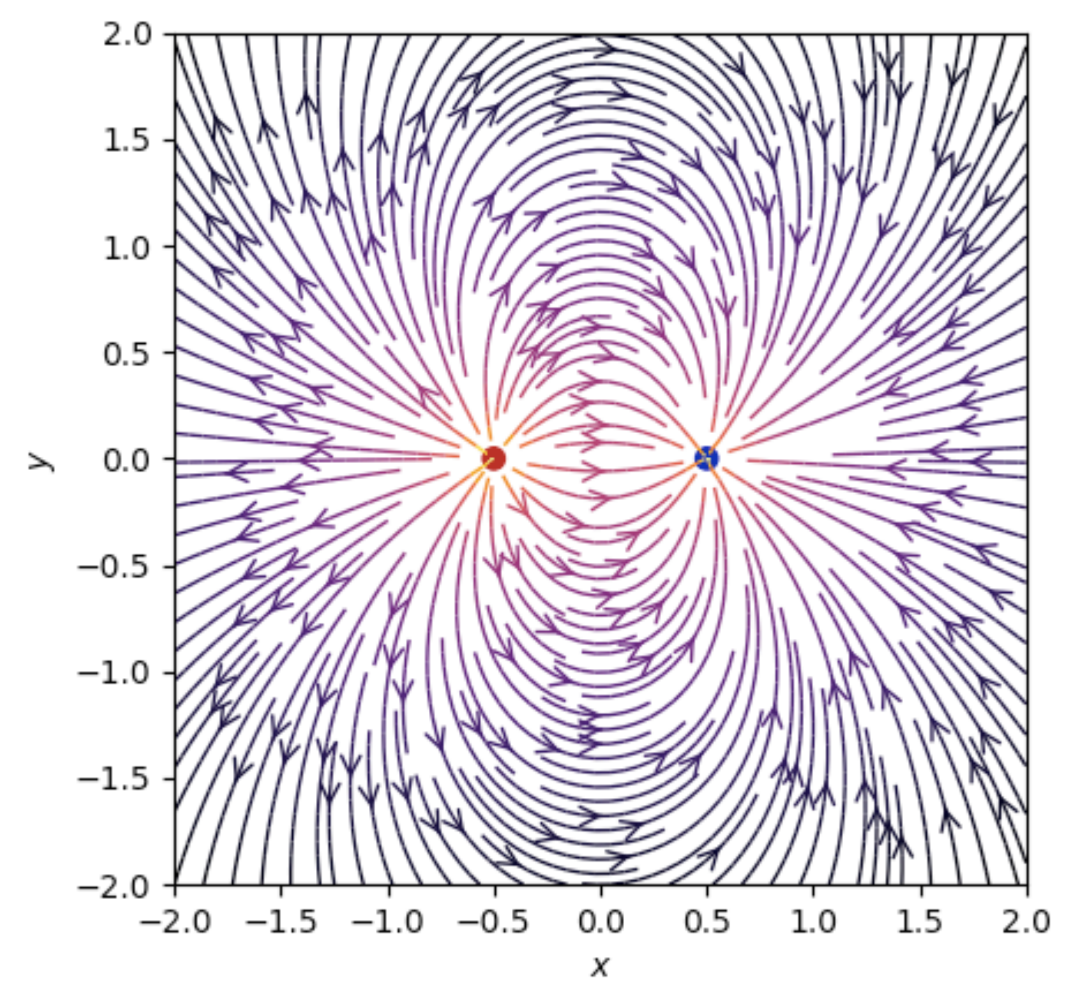


```
fig = plt.figure()
ax = fig.add_subplot(111)

# Plot the streamlines with an appropriate colormap and arrow style
color = 2 * np.log(np.hypot(Ex, Ey))
ax.streamplot(x, y, Ex, Ey, color=color, linewidth=1, cmap=plt.cm.inferno,
              density=2, arrowstyle='->', arrowsize=1.5)

# Add filled circles for the charges themselves
charge_colors = {True: '#aa0000', False: '#0000aa'}
for q, pos in charges:
    ax.add_artist(Circle(pos, 0.05, color=charge_colors[q>0]))

ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
ax.set_xlim(-2,2)
ax.set_ylim(-2,2)
ax.set_aspect('equal')
plt.show()
```

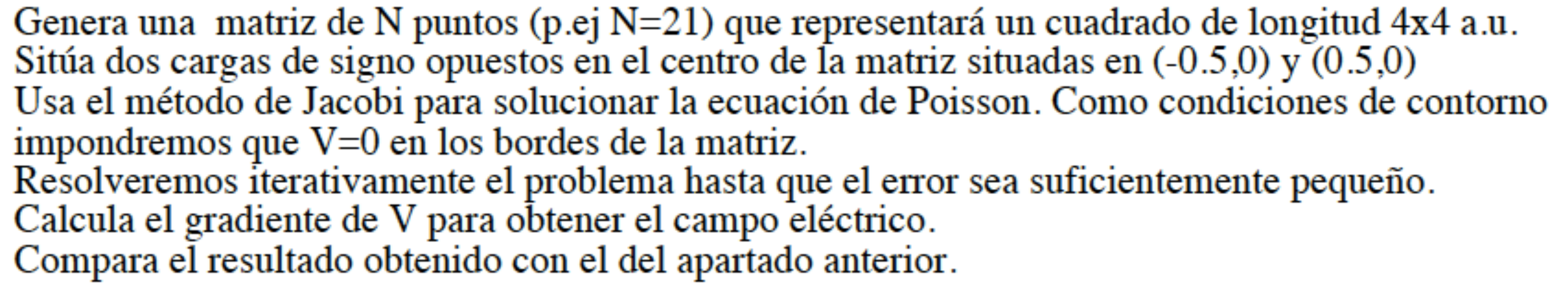


$$\bar{E} = \frac{1}{4\pi\varepsilon_0}\frac{q}{|r-r_0|^3}(\bar{r}-\bar{r}_0)$$

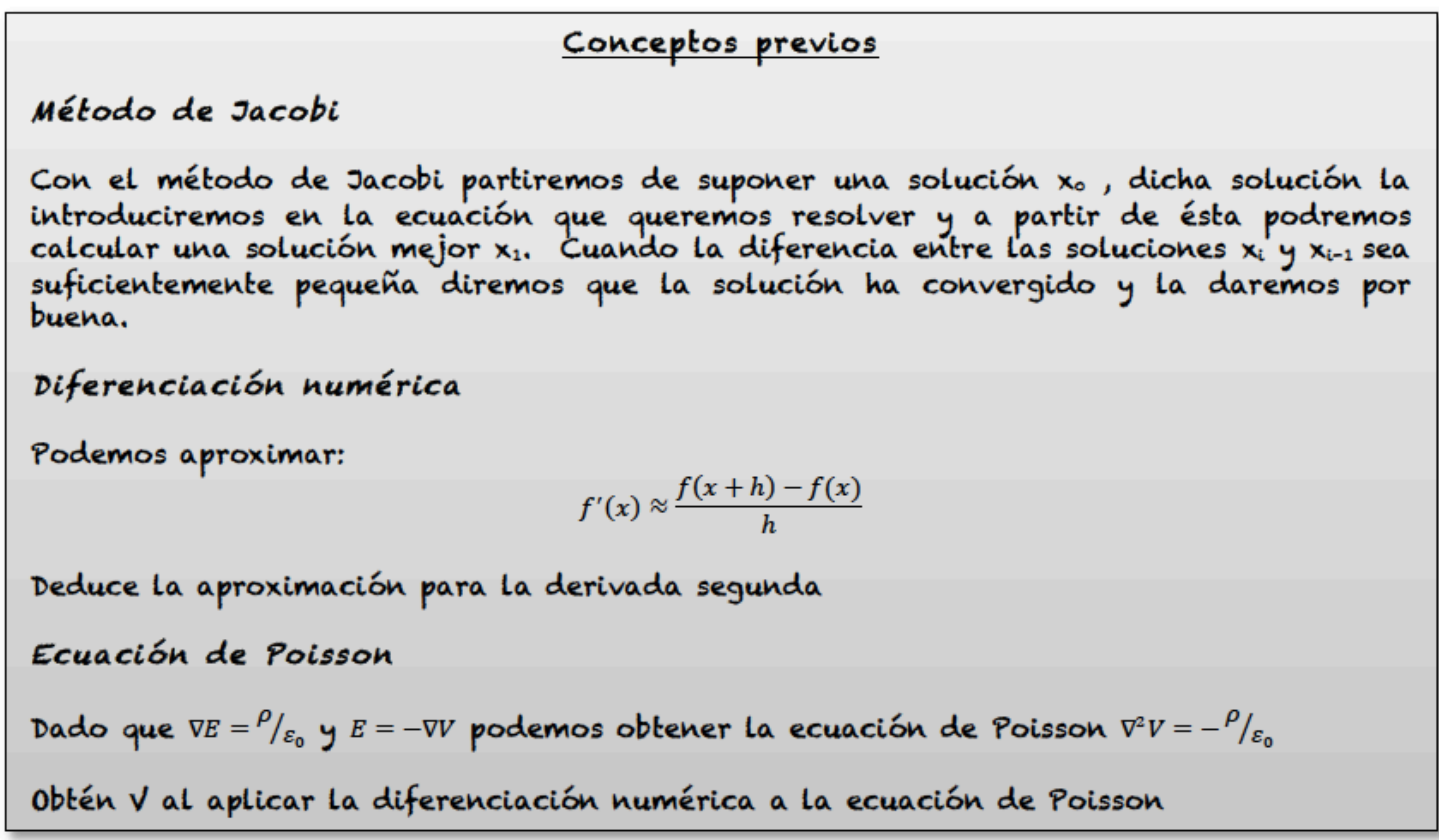
$$\bar{E} = \frac{1}{4\pi\varepsilon_0}\frac{q}{|r-\bar{r}_0|^2}\bar{u}_{\bar{r}-\bar{r}_0}$$


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

Campo eléctrico a partir de la solución de la ecuación de Poisson



Genera una matriz de N puntos (p.ej N=21) que representará un cuadrado de longitud 4x4 a.u.
Sitúa dos cargas de signo opuestos en el centro de la matriz situadas en (-0.5,0) y (0.5,0)
Usa el método de Jacobi para solucionar la ecuación de Poisson. Como condiciones de contorno impondremos que V=0 en los bordes de la matriz.
Resolveremos iterativamente el problema hasta que el error sea suficientemente pequeño.
Calcula el gradiente de V para obtener el campo eléctrico.
Compara el resultado obtenido con el del apartado anterior.



Conceptos previos

Método de Jacobi

Con el método de Jacobi partiremos de suponer una solución x_0 , dicha solución la introduciremos en la ecuación que queremos resolver y a partir de ésta podremos calcular una solución mejor x_1. Cuando la diferencia entre las soluciones x_i y x_{i-1} sea suficientemente pequeña diremos que la solución ha convergido y la daremos por buena.

Diferenciación numérica

Podemos aproximar:

$$f'(x) \approx \frac{f(x+h)-f(x)}{h}$$

Deduce la aproximación para la derivada segunda

Ecuación de Poisson

Dado que $\nabla E = {}^{\rho}/_{\varepsilon_0}$ y $E = -\nabla V$ podemos obtener la ecuación de Poisson $\nabla^2 V = -{}^{\rho}/_{\varepsilon_0}$

Obtén V al aplicar la diferenciación numérica a la ecuación de Poisson

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

$$f''(x) = \lim_{h\to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Poisson 2D

$$\nabla^2 V = -\rho/\varepsilon_0 \qquad \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = -\rho/\varepsilon_0$$

$\varepsilon_0 = 1$

h : resolució espacial

$$\frac{V(x+h,y) - 2V(x,y) + V(x-h,y)}{h^2} + \frac{V(x,y+h) - 2V(x,y) + V(x,y-h)}{h^2} = -\rho/\varepsilon_0$$

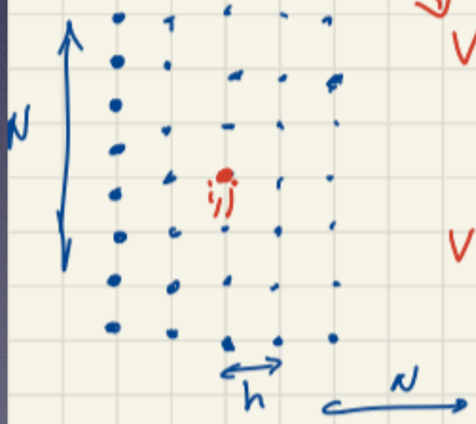
$$(1) \quad V(x,y) = \frac{1}{4}\left[V(x+h,y) + V(x-h,y) + V(x,y+h) + V(x,y-h) + \rho h^2\right]$$

JACOBI

$$V_0 = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&-&-&-&- \end{bmatrix} \xrightarrow[(1)]{} V_1 \xrightarrow{(1)} V_2 \rightarrow \cdots \xrightarrow{n} V_{n+1}$$

$V_1 - V_0 > eps$ $\quad$ $V_2 - V_1 > eps$ $\quad$ $V_{n+1} - V_n < eps$

$$(1) \quad V(x,y) = \frac{1}{4}\left[V(x+h,y) + V(x-h,y) + V(x,y+h) + V(x,y-h) + \rho(x,y) h^2\right]$$

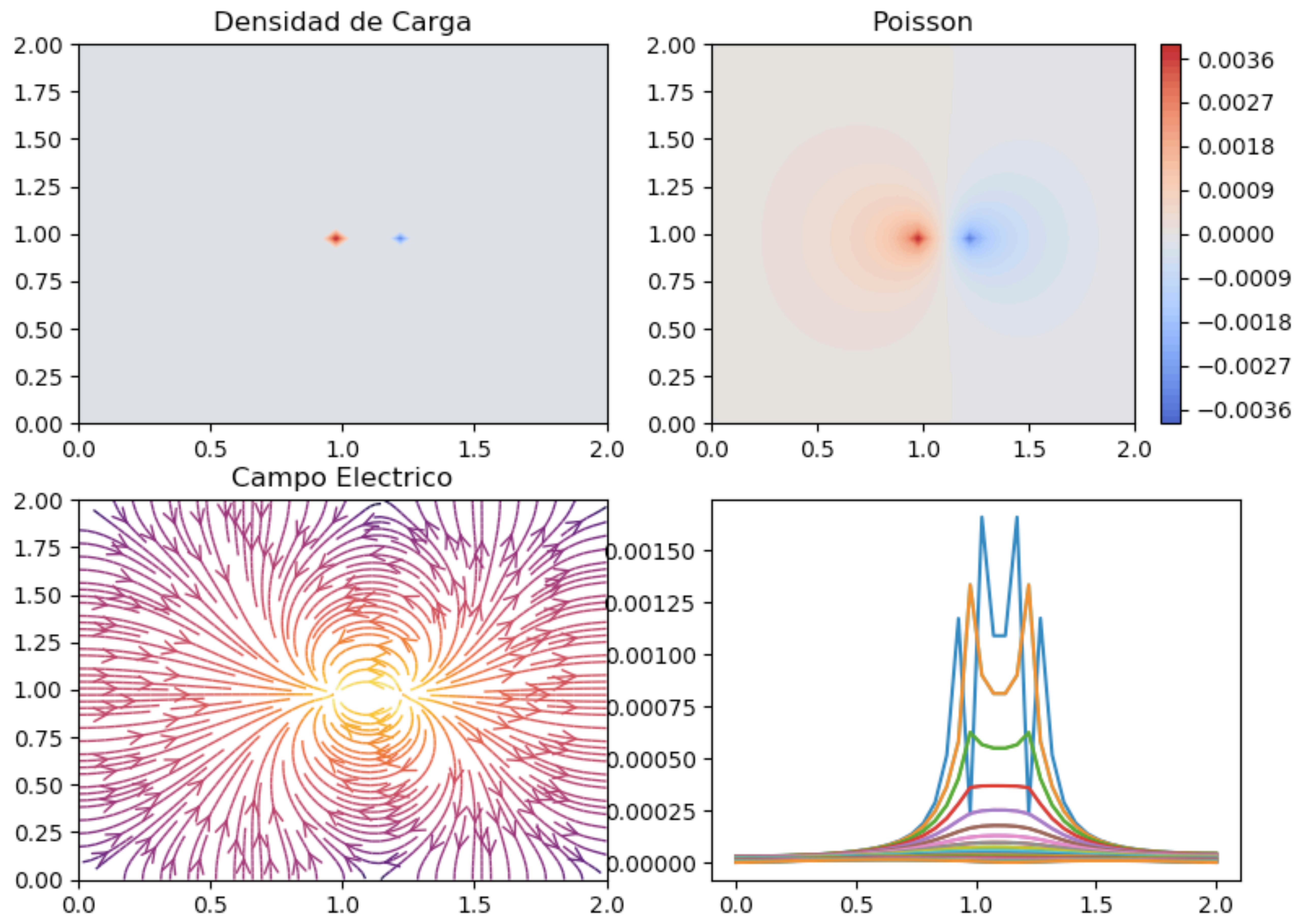


$$V(2,2) = \frac{1}{4}\left[\left(V(3,2) + V(1,2) + V(2,3) + V(2,1) + \rho(2,2)/h^2\right.\right.$$

$\vdots$

$$V(i,j) = \frac{1}{4}\left[\left(V(i+1,j) + \cdots\right.\right.$$

$$V_0 = \begin{bmatrix} 0&0&0&\cdots \\ 0&&& \\ \vdots&&& \\ 0&&& \end{bmatrix} \xrightarrow{(1)} V_1 = \begin{bmatrix} 0&0&0&0&0&&0 \\ 0&&&&&&0 \\ 0&&(2,2)&&&&0 \\ 0&&&&&&0 \\ 0&0&0&0&0&&0 \end{bmatrix} \rightarrow \cdots \xrightarrow{K} \cdots$$



Densidad de Carga

Poisson

Campo Electrico

Los resultados anteriores son cualitativos, para que las soluciones obtenidas por los dos métodos sean cuantitativamente similares y considerando epsilon_0 =1:

- V=q/(4*pi*r) , análogamente para E

- rho es la densidad de carga, como estamos resolviendo en dos dimensiones rho=q/h^2

- Podeis tambien resolverlo en tres dimensiones, donde rho=q/h^3 y habría que escribir el laplacians en tres dimensiones y representar los resultados utilizando por ejemplo:
- https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

Resultados con las correcciones anteriores