Electrostática Computacional
Potencial eléctrico bidimensional como matriz

1. Motivación

En electrostática computacional es relativamente sencillo extender un problema unidimensional a dos dimensiones. Sin embargo, esta extensión introduce una sutileza importante: **la diferencia entre coordenadas físicas y coordenadas de matriz**.

En este ejemplo se muestra cómo calcular un potencial eléctrico bidimensional y visualizarlo como una matriz, enfatizando que los ejes de la gráfica *no representan directamente longitudes físicas*, sino índices discretos de la malla numérica.

2. Modelo físico

Se considera un potencial bidimensional de la forma

$$\phi(x, y) = \frac{1}{x^2 + y^2}, \tag{1}$$

el cual presenta una singularidad en el origen $(0, 0)$, análoga al caso unidimensional.

3. Discretización del dominio

Se define una malla cuadrada con el mismo número de puntos en ambas direcciones:

$$N = 11.$$

Los dominios físicos se definen como

$$x \in [x_{\text{mín}}, x_{\text{máx}}], \qquad y \in [y_{\text{mín}}, y_{\text{máx}}],$$

con espaciamientos

$$\Delta x = \frac{x_{\text{máx}} - x_{\text{mín}}}{N - 1}, \qquad \Delta y = \frac{y_{\text{máx}} - y_{\text{mín}}}{N - 1}. \tag{2}$$

A pesar de esta definición física, la visualización final no utilizará directamente estas coordenadas.

4. Construcción de la matriz de potencial

El potencial se almacena en una matriz bidimensional

$$\phi_{i,j} = \phi(x_i, y_j),$$

donde los índices i y j representan posiciones dentro de la matriz, no coordenadas espaciales continuas.

5. Código completo en Python

El siguiente script construye la malla, evalúa el potencial y muestra la matriz utilizando `imshow`.

5.1. Script

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

# N mero de puntos por direcci n
N = 11

# Dominio en x
xmin = -1.0
xmax =  1.0
x = np.zeros(N)
dx = (xmax - xmin) / (N - 1)

# Dominio en y
ymin = -1.0
ymax =  1.0
y = np.zeros(N)
dy = (ymax - ymin) / (N - 1)

# Matriz del potencial
phi = np.zeros((N, N))

# Construcci n de la malla y c lculo del potencial
for ix in range(N):
    x[ix] = xmin + ix * dx
    for iy in range(N):
        y[iy] = ymin + iy * dy
        phi[ix, iy] = 1.0 / (x[ix]**2 + y[iy]**2)

# Visualizaci n como matriz
plt.figure()
plt.imshow(phi, cmap='viridis')
plt.colorbar(label='potencial')
plt.title('Matriz del potencial $\phi_{i,j}$')
plt.show()
```

6. Interpretación de la gráfica

La Figura 1 muestra el potencial representado como una imagen bidimensional.

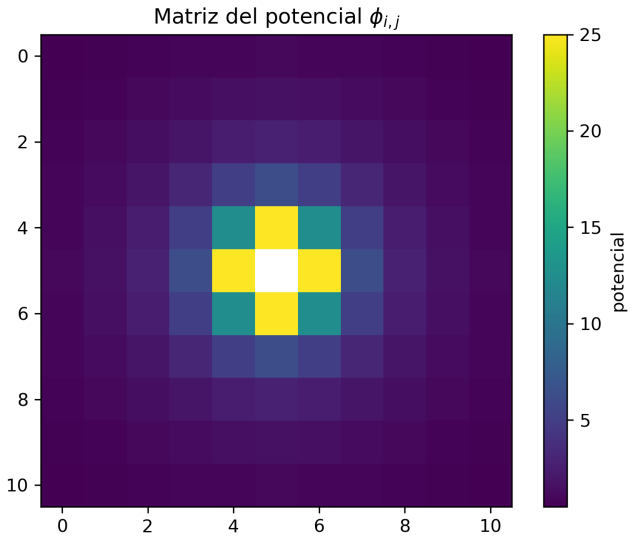

Figura 1: Visualización del potencial bidimensional como una matriz. Los ejes corresponden a índices de la matriz, no a coordenadas físicas.

Es fundamental notar que:

- Los ejes horizontal y vertical representan los índices i y j de la matriz.
- No existe correspondencia directa entre la escala de los ejes y las coordenadas físicas x y y.
- La función `imshow` asume una malla uniforme de índices por defecto.

7. Comentario computacional importante

Aunque el potencial fue calculado usando coordenadas físicas, la visualización:

- no conserva unidades de longitud,
- no muestra explícitamente los valores de x y y,
- interpreta la matriz como una imagen discreta.

Este enfoque es útil para:

- inspección rápida de resultados,
- depuración de códigos numéricos,
- visualización preliminar de soluciones.

Sin embargo, **no es adecuado** cuando se requiere una representación física correcta del dominio espacial.

8. Conclusión

Este ejemplo demuestra que es posible calcular fácilmente un potencial bidimensional, pero también pone en evidencia una limitación importante: la visualización directa con `imshow` representa índices de matriz, no coordenadas espaciales reales.

Este hecho motiva la introducción de herramientas como:

- `extent` en `imshow`,
- `meshgrid`,
- o métodos explícitos de graficación en coordenadas físicas.