Práctica 9-
Ecuaciones en Derivadas Parciales

Asignatura: *Computación Avanzada*

Profesor: *David Martín y Marero*

Luis Sánchez Izquierdo | jueves 10 de mayo de 2018

Contents

Introducción teórica

En esta práctica vamos a estudiar la resolución de ecuaciones en derivadas parciales de tipo parabólico. Son aquellas que tienen la forma

$$A\frac{\partial^2 u}{\partial x^2} + 2B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + F = 0$$

$$\begin{vmatrix} A & B \\ B & C \end{vmatrix} > 0$$

Numéricamente estos problemas se resuelven de forma iterativa. En este caso hemos estudiado el método de Gauss – Seidel con un factor sobre-amortiguado.

Objetivos

La práctica consiste en:

- Proponer un problema dado de electrostática. Yo he elegido el problema de una simetría especial, en el que resolveré la ecuación de Laplace.
- Describir la solución del problema y los algoritmos computacionales
- Presentar las soluciones obtenidas

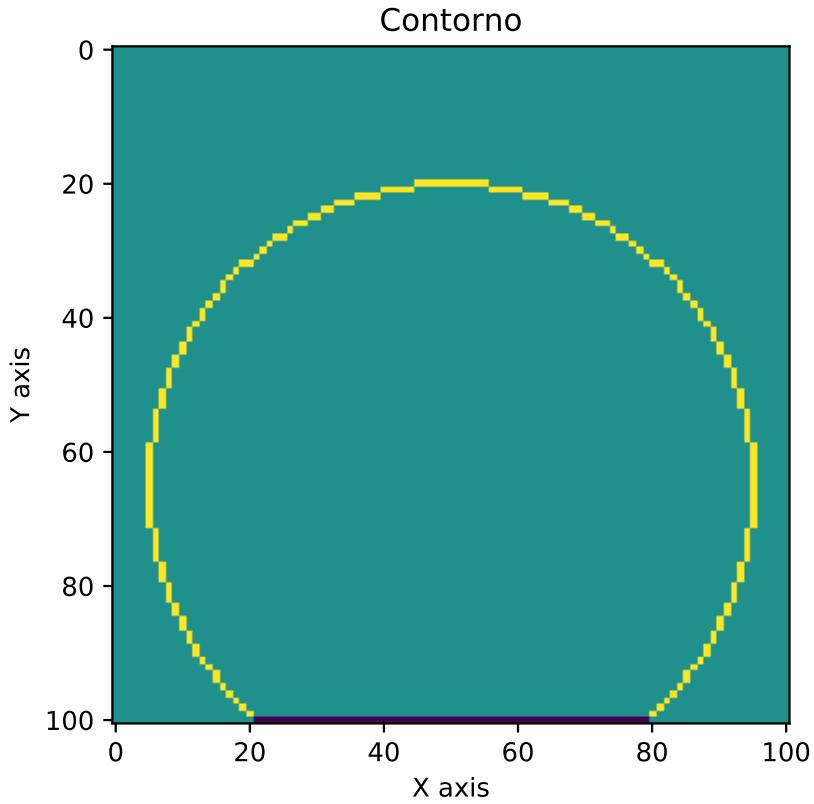
Además, realizaré un caso extra en el que obtengo la solución a la ecuación de Poisson sobre la misma geometría

Enunciado del problema

En este caso lo que voy a resolver es una geometría especial: una gota bidimensional de un material dieléctrico.

La motivación es para ver que el método de resolución de ecuaciones diferenciales se ajusta a cualquier geometría, ya que lo que hace es dividir el problema total en muchas partes pequeñas con un mallado específico.

El mallado consiste en una gota:



La gota tiene un borde que delimita con el aire (borde amarillo) y otro borde que delimita con el suelo (borde azul oscuro).

La ecuación que rige el comportamiento es la ecuación de Poisson:

$$\nabla^2 E = \frac{\partial^2 E(x,y)}{\partial x^2} + \frac{\partial^2 E(x,y)}{\partial y^2} = \frac{\rho(x,y)}{\varepsilon_0}$$

Sin embargo, en la primera parte no tendremos cargas, de modo que restringimos el problema a la ecuación de Laplace:

$$\nabla^2 E = \frac{\partial^2 E(x,y)}{\partial x^2} + \frac{\partial^2 E(x,y)}{\partial y^2} = 0$$

Algoritmo de resolución

Se crean los valores iniciales del potencial y de la densidad de carga. Como se trata de la ecuación de Laplace el valor de la densidad es $\rho(i,j) = 0$

```
# Constants
M = 100         # Grid squares on a side
V = 1.0         # Voltage at top wall
target = 1e-3   # Target accuracy

# Condiciones de la esfera
R = M/2.2;
H = R/3;

# Create arrays to hold potential values
phi = zeros([M+1,M+1],float)

# Sin cargas
rho = zeros([M+1,M+1],float)
#rho[20:40, 60:80] = -1
#rho[60:80, 20:40] = 1
```

- ## Geometría del sistema

Y se crea la geometría del sistema. La variable pos guarda los índices que son la gota. Vienen impuestos por la condición:

$$\left(r_x - \frac{M}{2}\right)^2 + \left(r_y - \frac{M}{2}\right)^2 \leq R^2$$

```
pos = full((M+1,M+1), False)

for i in range(M+1):
    for j in range(M+1):
        if (i-M/2-H)**2 + (j-M/2)**2 <= R**2:
            pos[i,j]=True
```

Y el borde será a partir del exterior y el interior:

```
geom = zeros((M+1,M+1))             # para mostrar la geometría
borde = full((M+1,M+1), False)      # borde con el aire
borde2 = full((M+1,M+1), False)     # borde con el suelo
interior = full((M+1,M+1), False)   # zona interior

for i in range(1,M):
    for j in range(1,M):
        if pos[i,j] == True:
            if pos[i-1,j] == False or pos[i+1,j] == False \
                or pos[i,j-1] == False or pos[i,j+1] == False:
                    borde[i,j] = True
                    geom[i,j] = 1


for j in range(1,M):
    if pos[M,j]==True:
        borde2[M,j] = True
        borde[M,j] = True
        geom[M,j] = -1

for i in range(M+1):
    for j in range(M+1):
        if pos[i,j] == True and borde[i,j]==False:
            interior[i,j]=True
```

- Algoritmo convergente

Para resolver la ecuación se sigue el algoritmo de Gauss-Seidel con método sobre-amortiguado, que básicamente nos dice que el nuevo valor dependerá del anterior como:

$$\phi'(x,y) = \frac{1+\omega}{4}[\,\phi(x+a,y) + \phi(x-a,y) + \phi(x,y-a) + \phi(x,y-a)] - \omega\phi(x,y)$$

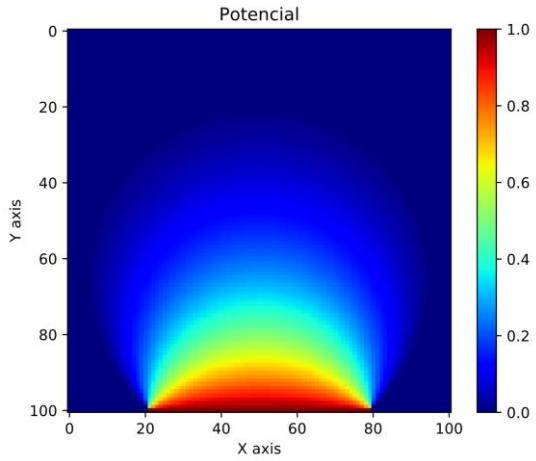
Las líneas de código son:

```
# CONDICIONES DE CONTORNO
for i in range(M):
    if borde[M,i] == True:
        phi[M,i] = V

a = 1
c= a**2 / (4 * 1)

omega = 0.9

# Main loop
delta = 1.0
while delta>target:

    # Calculate new values of the potential
    delta = 0
    for i in range(M,1,-1):
        for j in range(M,1,-1):
            if interior[i,j] == True:
                phi_aux = phi[i,j]
                phi[i,j] = (1+omega)*(phi[i+1,j] + phi[i-1,j] + phi[i,j+1] + phi[i,j-1])/4 \
                           - omega * phi[i,j] + c *rho[i,j]
                if abs(phi[i,j]-phi_aux)>delta:
                    delta = abs(phi[i,j]-phi_aux)

    # Calculate maximum difference from old values
    print("delta: ", delta)

elapsed = time() - start
print("Elapsed time: ", elapsed, "seconds")
```

Resultados obtenidos

Primeramente, resuelvo el problema de Laplace con condición de contorno borde inferior a potencial V y el resto de los bordes a potencial nulo

La solución se puede ver a continuación. Lo que se representa es el avance del potencial con forma curvada. El exterior de la gota está formado de un material metálico que hace que el potencial allí sea constante, de ahí que los resultados sean como los mostrados en la figura.

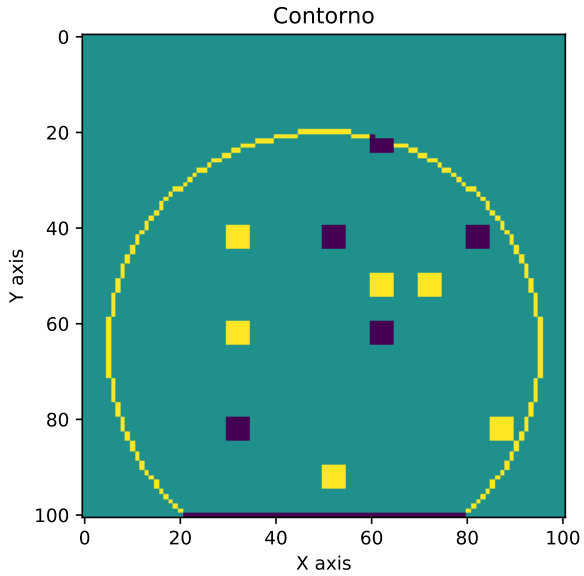


Apartado Extra:

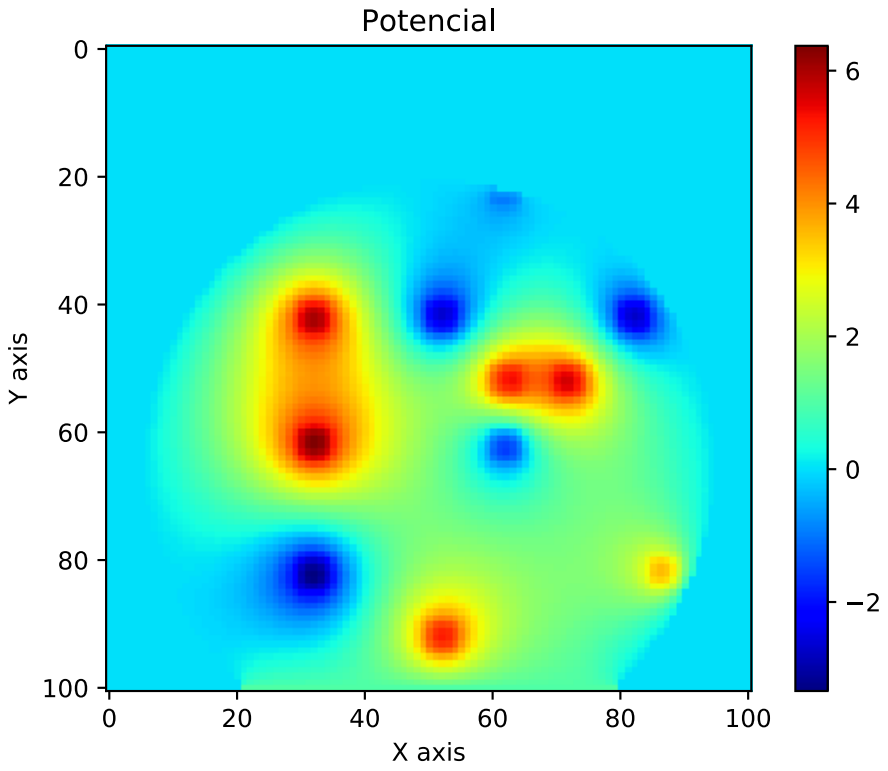
Ahora propongo una gota con partículas en el interior. Las coloco de forma aleatoria:

```
rho = zeros([M+1,M+1],float)
rho[20:25, 60:65] = -1
rho[40:45, 50:55] = -1
rho[50:55, 70:75] = 1
rho[80:85, 30:35] = -1
rho[50:55, 60:65] = 1
rho[80:85, 85:90] = 1
rho[60:65, 30:35] = 1
rho[90:95, 50:55] = 1
rho[60:65, 60:65] = -1
rho[40:45, 30:35] = 1
rho[30:35, 85:90] = -1
rho[40:45, 80:85] = -1
```

La geometría es pues:



Esta vez se trata del problema de Poisson, ya que $\rho(x,y)$ es distinto de cero. Al realizar la simulación se obtiene:



Conclusiones

Como se puede ver, hemos realizado una simulación de un sistema eléctrico y para ello hemos tenido que resolver una ecuación diferencial. Para ello hemos usado el método de Gauss-Seidel que converge más rápidamente usando la sobre-amortiguación.

Primeramente, hemos resuelto la ecuación de Laplace y en un apartado extra hemos visto la ecuación de Poisson. En ambos casos la geometría era especial: una gota apoyada sobre una placa a voltaje V.

Gracias a la simulación se ha comprobado de el algoritmo de resolución funciona y resulta en la solución del problema: es un algoritmo que converge.