Álgebra Lineal Computacional

Primer Cuatrimestre 2022

Trabajo Práctico N° 2.

En este trabajo vamos a analizar la convergencia del método SOR para resolver sistemas lineales, para distintos valores del parámetro ω.

1. Escribir un programa que implemente el método de Jacobi para resolver un sistema $Ax = b$.

   El programa debe recibir la matriz A, el vector b, un vector inicial $x^{(0)}$ y la cantidad máxima de iteraciones s, y debe finalizar cuando el error relativo sea pequeño:

   $$\frac{\|Ax^{(k)} - b\|}{\|b\|} < 10^{-8}$$

   o se alcance la cantidad máxima de iteraciones. Debe devolver una lista de dos elementos, el primer elemento es la solución obtenida y el segundo elemento la cantidad de iteraciones realizadas. Si no se alcanza convergencia, debe devolver el último vector obtenido $x^{(s)}$ y el valor s.

2. Utilizar el programa implementado para resolver el sistema

   $$\begin{pmatrix} 4 & 2 & 1 \\ 1 & 4 & 1 \\ 2 & 1 & 5 \end{pmatrix} x = \begin{pmatrix} 9 \\ 5 \\ 0 \end{pmatrix}$$

   con vector inicial $x^{(0)} = (1, 1, 1)$ y un máximo de 50 iteraciones.

   ¿Se alcanza convergencia? ¿En cuántos pasos?

3. **Método SOR** Modificar el programa anterior para implementar el método SOR para resolver un sistema $Ax = b$. El programa debe recibir como parámetro adicional el valor ω.

   Recordar que la iteración del método SOR viene dada por la fórmula

   $$\mathbf{x}^{(k+1)} = (D + \omega L)^{-1}\big(\omega \mathbf{b} - [\omega U + (\omega - 1)D]\mathbf{x}^{(k)}\big).$$

4. Utilizar el programa implementado para resolver el sistema del ítem 2 utilizando valores $\omega = 1$, $\omega = 1.5$ $\omega = 2.5$, con vector inicial $x^{(0)} = (1, 1, 1)$ y un máximo de 50 iteraciones. ¿En qué casos se alcanza convergencia? ¿En cuántos pasos?

5. Construir una matriz $A \in \mathbb{R}^{5\times 5}$ con entradas aleatorias en el intervalo $[0, 1]$ en las casillas fuera de la diagonal y $a_{ii} = 4$ para $1 \leq i \leq 5$. Tomar como b y $x^{(0)}$ vectores aleatorios en $\mathbb{R}^5$. Verificar si el método de Jacobi implementado converge para esta matriz con una cantidad máxima de 100 iteraciones.

6. Para la matriz construida en el ítem anterior:

   (a) Correr el método SOR para 1001 valores de ω equiespaciados en el intervalo $[0, 2]$ y una cantidad máxima de 100 iteraciones. Guardar en una lista o un vector la cantidad de pasos utilizadas para cada valor de ω.

(b) Graficar la cantidad de pasos en función de ω.

(c) ¿En qué intervalo de valores de ω se obtuvo convergencia del método sin superar la cantidad máxima de iteraciones?

(d) ¿Para qué valor de ω se obtuvo la convergencia más rápida?

Importante: tanto en este ejercicio como en los ejercicios 8 y 9, utilizar siempre la misma matriz A y los mismos vectores b y $x^{(0)}$, no generar cada vez una nueva matriz aleatoria o vectores aleatorios.

Sugerencias:

- Para generar puntos equiespaciados utilizar el comando `np.linspace`.
- Para graficar los valores obtenidos, utilizar `plot` del paquete `matplotlib.pyplot`.

Por ejemplo, para graficar la función seno entre 0 y 5, utilizamos el siguiente código.

```
import matplotlib.pyplot as plt
x = np.linspace(0,5,1001)  # Tomamos 1001 puntos equiespaciados
y = np.sin(x)              # Calculamos seno(x) para los 1001 valores
plt.plot(x, y)             # Graficamos y en funcion de x
```

7. Implementar un programa que reciba una matriz A y un valor ω y devuelva el radio espectral de la matriz del método SOR:

$$T = (D + \omega L)^{-1}(-[\omega U + (\omega - 1)D])$$

Sugerencia: el comando `np.linalg.eigvals` devuelve un vector con los autovalores de la matriz.

8. Para la matriz construida en el ítem 5,

(a) Calcular el radio espectral de la matriz del método SOR para 1001 valores de ω equiespaciados en el intervalo $[0, 2]$. Guardar en una lista o un vector los valores obtenidos para cada valor de ω.

(b) Graficar el radio espectral en función de ω.

(c) ¿Para qué valores de ω el radio espectral es menor que 1?

(d) ¿Para qué valor de omega se obtuvo el radio espectral más chico?

(e) ¿Coiniciden los resultados obtenidos con los obtenidos en el ítem anterior? ¿Cómo explican las diferencias?

9. Para la matriz construida en el ítem 5,

(a) Calcular el determinante de la matriz T del método SOR para 1001 valores de ω equiespaciados en el intervalo $[0, 2]$ (construyendo la matriz y calculando su determinante o utilizando la fórmula vista en clase). Guardar en una lista o un vector los valores obtenidos para cada valor de ω.

(b) Graficar el determinante de T en función de ω.

(c) ¿Para qué valores de ω el determinante es menor que 1?

(d) ¿Para qué valor de ω se obtuvo el determinante más chico?

10. (opcional) Superponer en un mismo gráfico el gráfico de la cantidad de pasos en función de ω y el gráfico del radio espectral en función de ω. Para esto, puede modificar el siguiente ejemplo que grafica dos funciones en el mismo gráfico utilizando dos escalas distintas en el eje Y.

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

# Graficamos la funcion seno y la funcion exponencial
t = np.linspace(0,10,1001)
data1 = np.exp(t)
data2 = np.sin(2 * np.pi * t)

fig, ax1 = plt.subplots()

color = 'tab:red'   # Color del primer conjunto de datos
ax1.set_xlabel('time (s)')  # Etiqueta del eje x
ax1.set_ylabel('exp', color=color)  # Etiqueta del eje y
ax1.plot(t, data1, color=color) # Se grafica el primer conjunto de datos
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # Usamos un segundo eje y que comparte el mismo eje x

color = 'tab:blue'
ax2.set_ylabel('sin', color=color)
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()
plt.show()
```