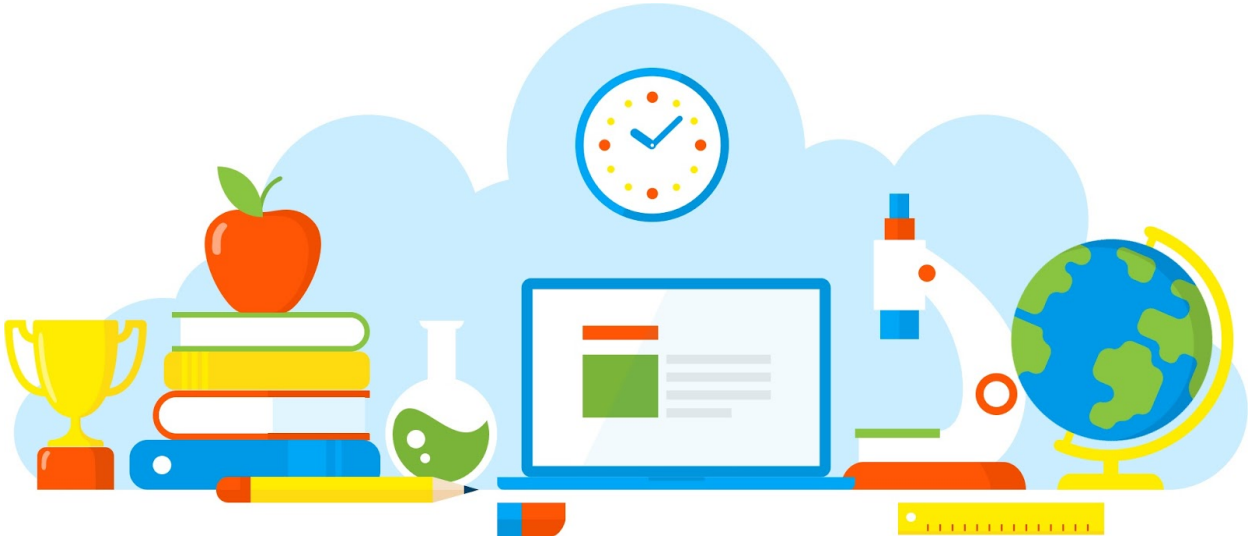

Método de Seidel

Matemática Numérica

G.21

Eva Ercilia Vázquez García.
Jesús Manuel Castellanos Reynaldo.
Ernesto Alejandro Carralero Conde.

Solución

El problema presenta un sistema lineal de 9 ecuaciones con 9 incógnitas. En este caso, el algoritmo de Seidel se utiliza a riesgo y se toma como condición de terminación $\|x^{(n)} - x^{(n-1)}\| \leqslant \varepsilon$ con una tolerancia mucho menor que la que se desea. Como en el problema se piden cuatro cifras decimales exactas, se tomará una tolerancia de ε = 0.000005, diez veces menor que la necesaria.

El algoritmo utilizado está basado en el Ejemplo 10 del libro de texto “Matemática Numérica”. Se ha realizado un script en Python para probar este algoritmo y proporcionar los resultados del problema.

El código siguiente implementa el método iterativo de Seidel para resolver el sistema lineal proporcionado. Las condiciones de frontera se definen para los bordes de la matriz (excluyendo las esquinas) y se inicializan todas las incógnitas en cero. Luego, sobre cada incógnita dentro de la matriz (excluyendo los bordes), se va actualizando la variable provisional *prov* para guardar el nuevo valor sin borrar el de la iteración anterior. De esta manera se puede hallar la diferencia entre ambos valores y así actualizar la variable *error_maximo* si es necesario.

Este procedimiento se realiza hasta que el error máximo en cualquier incógnita cae por debajo del valor de tolerancia especificado. Los resultados se guardan en dos archivos CSV: uno que contiene las matrices completas para cada iteración y otro que contiene una tabla con el valor y el error para cada incógnita en cada iteración. Para terminar se imprime la solución final.

Además, se han incluido dos PDF con los resultados de las iteraciones para la tabla y para la matriz. La matriz que se elabora está en correspondencia con la Figura 3 del principio del capítulo 3 del libro de texto y tiene una estructura así para cada **u[i, j]**:

u[0, 0]	u[0, 1]	u[0, 2]	u[0, 3]	u[0, 4]
u[1, 0]	u[1, 1]	u[1, 2]	u[1, 3]	u[1, 4]
u[2, 0]	u[2, 1]	u[2, 2]	u[2, 3]	u[2, 4]
u[3, 0]	u[3, 1]	u[3, 2]	u[3, 3]	u[3, 4]
u[4, 0]	u[4, 1]	u[4, 2]	u[4, 3]	u[4, 4]

Script en Python:

```
import numpy as np
import csv

# Constantes
TAM_MATRIZ = 5  # Cantidad de filas y columnas de la matriz de incógnitas u.
MAX_ITER = 100  # Número máximo de iteraciones que se realizarán.
TOLERANCIA = 0.000005  # Error máximo permitido entre dos iteraciones consecutivas.
PROV_DIV = 8.0  # Coeficiente de u[i, j] en el sistema lineal

# Inicializar con ceros la matriz de incógnitas u
u = np.zeros((TAM_MATRIZ, TAM_MATRIZ))

# Definir las condiciones de frontera (excluyendo las esquinas) para la matriz u
condiciones_de_frontera = {
    "arriba": {"indice": 0, "k": slice(1, -1), "valor": 25},
    "abajo": {"indice": -1, "k": slice(1, -1), "valor": -3},
    "izquierda": {"indice": slice(1, -1), "k": 0, "valor": 18},
    "derecha": {"indice": slice(1, -1), "k": -1, "valor": 12}
}
for frontera in condiciones_de_frontera.values():
    u[frontera["indice"], frontera["k"]] = frontera["valor"]

# Método iterativo de Seidel
resultados = []

# Abrir un archivo CSV para guardar las matrices de cada iteración.
with open("matrices_iteraciones.csv", "w", newline='') as csvfile:
    escritor_csv = csv.writer(csvfile)
    escritor_csv.writerow(["Iteración: 0"])
    escritor_csv.writerows(u)
    escritor_csv.writerow([])
    # Realizar las iteraciones.
    for iteracion in range(MAX_ITER):
        error_maximo = 0
        # Iterar sobre cada punto interior de la matriz (no en la frontera).
        for i in range(1, TAM_MATRIZ - 1):
            for j in range(1, TAM_MATRIZ - 1):
                # Calcular el nuevo valor de la incógnita.
                prov = (u[i - 1, j] + u[i + 1, j] + u[i, j - 1] + u[i, j + 1]) /
PROV_DIV
                # Calcular el error como la diferencia absoluta entre el nuevo y el
viejo valor de la incógnita.
                error = abs(prov - u[i, j])
                # Actualizar el error máximo si es necesario.
                error_maximo = max(error_maximo, error)
                # Actualizar el valor de la incógnita en la matriz u.
                u[i, j] = prov
                # Guardar los resultados de esta iteración.
                resultados.append([iteracion + 1, i, j, prov, error])
        # Escribir la matriz actualizada en el archivo CSV.
```

```
        escritor_csv.writerow([f"Iteración: {iteracion + 1}"])
        escritor_csv.writerows(u)
        escritor_csv.writerow([])
        # Si el error máximo es menor que la tolerancia, detener las iteraciones.
        if error_maximo < TOLERANCIA:
            break
print("Archivo CSV con todas las matrices guardado como matrices_iteraciones.csv")

# Guardar la tabla de valores de iteración en un archivo CSV
encabezados = ["Iteración", "i", "j", "Valor de u[i, j]", "Error"]
with open("tabla_iteraciones.csv", "w", newline='') as csvfile:
    escritor_csv = csv.writer(csvfile)
    escritor_csv.writerow(encabezados)
    iteracion_previa = 1
    for resultado in resultados:
        # Si la iteración actual es diferente de la anterior,
        # escribir una fila vacía en el archivo CSV para separar las iteraciones.
        if resultado[0] != iteracion_previa:
            escritor_csv.writerow([])
            iteracion_previa = resultado[0]
        # Escribir los resultados de esta iteración en el archivo CSV.
        escritor_csv.writerow(resultado)
print("Tabla con cada iteración guardada en tabla_iteraciones.csv\n")

# Imprimir la solución final aproximando los valores de cada incógnita a enteros
print("La solución es u[i,j] (i = 1, 2, 3; j = 1, 2, 3):")
for i in range(1, TAM_MATRIZ - 1):
    print('\t\t'.join(str(int(u[i, j])) for j in range(1, TAM_MATRIZ - 1)))
```

Respuesta obtenida:

Archivo CSV con todas las matrices guardado como matrices_iteraciones.csv
Tabla con cada iteración guardada en tabla_iteraciones.csv

La solución es u[i,j] (i = 1, 2, 3; j = 1, 2, 3):

6	4	5
3	1	2
2	0	1