Metody numeryczne

Mateusz Kwolek

1. **Rozwiąż układ równań**

$$\begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix} \tag{1}$$

Uzasadnij wybór algorytmu. Uwaga! (1) to macierz rzadka, więc użycie algorytmu dla macierzy pełnej jest nieprawidłowe (zadanie nie będzie zaliczone).

1) Wybór algorytmu

Algorytmem, którym posłużyłem się rozwiązując podane zadanie jest metoda eliminacji Gaussa-Crouta. Algorytm ten jest swoistym ulepszeniem samej eliminacji Gaussa, w której mogłoby się zdarzyć, że przy dzieleniu przez elementy przekątnej głównej macierzy współczynników, któryś z nich jest równy zero lub zostanie wyzerowany w wyniku obliczeń. Algorytm Gaussa-Crouta jest stosowany w przypadku macierzy rzadkich gdyż jest on dla nich szybkobieżny i eliminuje błąd numeryczny.
Metoda Gaussa-Crouta polega na tym, iż na początku eliminacji wyszukujemy w wierszu macierzy element o największym module, po czym zamieniamy miejscami kolumnę ze znalezionym elementem z kolumną zawierającą element głównej przekątnej. W ten sposób dzielnik będzie posiadał największą na moduł wartość i pozbędziemy się sytuacji, gdy może on posiadać wartość 0.

2) Kod źródłowy

Do stworzenia programu użyłem języka Python ze względu na jego prostotę, bogactwo bibliotek oraz częste zastosowanie do wykonywania obliczeń matematycznych.

```python
import numpy as np
from scipy.sparse import csc_matrix
from scipy.sparse.linalg import spsolve

A = csc_matrix([
    [4, 1, 0, 0, 0, 0, 1],
    [1, 4, 1, 0, 0, 0, 0],
    [0, 1, 4, 1, 0, 0, 0],
    [0, 0, 1, 4, 1, 0, 0],
    [0, 0, 0, 1, 4, 1, 0],
    [0, 0, 0, 0, 1, 4, 1],
    [1, 0, 0, 0, 0, 1, 4]
])

b = np.array([1, 2, 3, 4, 5, 6, 7])

# Do rozwiązania układu z zadaną macierzą rzadką wykorzystałem
# metodę eliminacji Gaussa-Crouta.

# Opcja #1 - bogactwo bibliotek Pythona
def solve_matrix_scipy(A,b):
    # Skorzystanie z biblioteki SciPy,
    # oraz funkcji spsolve przeznaczonej do rozwiązywania układów z macierzami rzadkimi.
    x = spsolve(A, b)
    return x

# Opcja #2 - własna implementacja algorytmu
def gauss_crout(A, b):
    n = A.shape[0]
    L = np.zeros((n, n))
    U = np.zeros((n, n))

    # Rozkład LU metodą Gaussa-Crouta
    for i in range(n):
        for j in range(i, n):
            L[j, i] = A[j, i] - sum(L[j, k] * U[k, i] for k in range(i))
        for j in range(i, n):
            if i == j:
                U[i, j] = 1
            else:
                U[i, j] = (A[i, j] - sum(L[i, k] * U[k, j] for k in range(i))) / L[i, i]

    # Rozwiązanie układu Ly = b
    y = np.zeros(n)
    for i in range(n):
        y[i] = (b[i] - sum(L[i, j] * y[j] for j in range(i))) / L[i, i]

    # Rozwiązanie układu Ux = y
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = y[i] - sum(U[i, j] * x[j] for j in range(i+1, n))

    return x

# Wywołanie funkcji
C = gauss_crout(A, b)
C1 = solve_matrix_scipy(A, b)
print("Rozwiązanie układu równań:")
print("1) Rozwiązanie układu równań funkcją z biblioteki SciPy:\n", C1)
print("2) Rozkład LU metodą Gaussa-Crouta:\n", C)

```

3) Wynik

```
Rozwiązanie układu równań:
1) Rozwiązanie układu równań funkcją z biblioteki SciPy:
 [-0.2601626   0.44715447  0.47154472  0.66666667  0.86178862  0.88617886  1.59349593]
2) Rozkład LU metodą Gaussa-Crouta:
 [-0.2601626   0.44715447  0.47154472  0.66666667  0.86178862  0.88617886  1.59349593]
```