Zadanie numeryczne 2

Kamila Dmowska

15 listopada 2021

1 Wstęp

W tym zadaniu, należy rozwiązać podany układ równań.

$$\begin{bmatrix} b_0 & c_0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_1 & b_1 & c_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & a_3 & b_3 & c_3 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & a_N & b_N \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-1} \\ y_N \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-1} \\ f_N \end{bmatrix}$$

Rysunek 1: Układ równań do rozwiązania

przy pomocy wybranego algorytmu do rozwiązywania układu z macierzą trójdiagonalną, przy podanych warunkach.

$$b_0 = b_N = f_N = 1, \qquad c_0 = a_N = 0, \qquad f_{0:N-1} = 0,$$
$$b_n = -\frac{2}{h^2}, \qquad a_n = c_n = \frac{1}{h^2}, \qquad h = \frac{1}{N}$$

Rysunek 2: Założenia do układu równań

2 Implementacja

Do rozwiązania podanego układu zastosowałam algorytmu Thomasa.
Do zaimplementowania algorytmu użyłam języka **Python** wraz z biblioteką do obliczeń numerycznych **numpy** oraz biblioteką do tworzenia wykresów **matplotlib**.

```
N = 20
h = 1.0/N
a = np.zeros(N-1)
b = np.zeros(N)
c = np.zeros(N-1)
f = np.zeros(N)
```

Inincjalizuję tablicę za pomocą np.zeros, czyli n-elentowa tablicę w ktorej wszystkie elementy sa zerami, ponieważ zaalokowana wcześniej pamięć na tą tablicę umożliwi szybsze działanie kodu.

```
b[0] = 1
b[N-1] = 1
f[N-1] = 1
c[0] = 0
a[N-2] = 0

for i in range(1, N-1):
    b[i] = -2 / (h*h)

for i in range(0, N-2):
    a[i] = 1 / (h**2)
for i in range(1, N-1):
    c[i] = 1 / (h**2)
```

Dodaję tutaj założenia które były podane w zadaniu, a następnie tworzę 3 pętle, kolejno korzystając z założeń z zadania, przypisuje kolejne wartości tablic b, a, c.

```
def t_algorithm(a, b, c, f):
    n = len(f)
    beta = np.zeros(n - 1, float)
    gamma = np.zeros(n, float)
    x = np.zeros(n, float)

    beta[0] = (c[0] / b[0])
    gamma[0] = f[0] / b[0]

    for i in range(1, n - 1):
        beta[i] = (c[i] / (b[i] - a[i - 1] * beta[i - 1]))

    for i in range(1, n):
        gamma[i] = (f[i] - (a[i - 1] * gamma[i - 1])) / (b[i] - (a[i - 1] * beta

    x[n - 1] = gamma[n - 1]

    for i in range(n - 1, 0, -1):
```

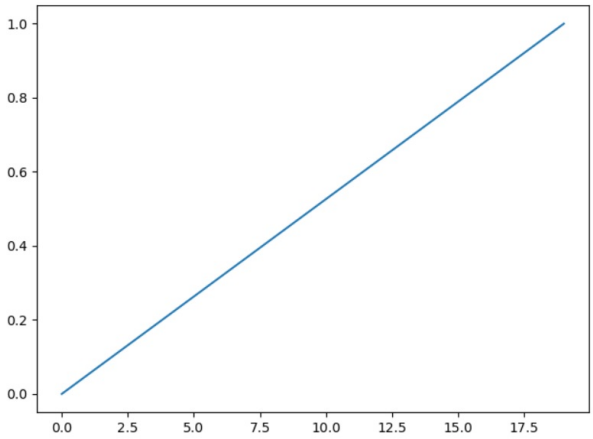
```
        x[i - 1] = gamma[i - 1] - (beta[i - 1] * x[i])
    return x
```

Funkcja $talgorithm(a, b, c, f)$ implementuje algorytm Thomasa. Konieczne jest zaincjowanie n równej długości tablicy f wyrazow wolnych.

3 Wyniki

Dla danego $N = 20$ oraz założenień podanych w treści zadania, otrzymaliśmy podane wyniki.

[0. 0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
0.31578947 0.36842105 0.42105263 0.47368421 0.52631579 0.57894737
0.63157895 0.68421053 0.73684211 0.78947368 0.84210526 0.89473684
0.94736842 1.]



Rysunek 3: Wyniki w postaci graficznej

4 Wnioski

Zaimplementowany algorytm poprawnie rozwiązał układ równań. Jego złożoność wynosi $O(n)$.