Metody Numeryczne – projekt 2

Paweł Gościak 188778

1. Cel projektu

Celem projektu było zaimplementowanie trzech metod rozwiązywania układy równań – dwóch metod iteracyjnych: Jacobiego i Gaussa-Seidla oraz jednej metody bezpośredniej (faktoryzacja LU). Projekt został napisany w języku Python przy użyciu biblioteki *matplotlib* do rysowania wykresu. Układ równań liniowych wykorzystywany w obliczeniach jest postaci:

$$Ax = b$$

gdzie A jest daną macierzą pasmową, b danym wektorem wyrazów wolnych, a x jest wektorem rozwiązań.

2. Analiza zadania

Zadanie A:

Dla numeru indeksu 188778 otrzymujemy wartości: a1 = 12, a2 = a3 = -1, N=987, b (wektor o długości N), którego n−ty element ma wartość sin(n · 9)

$$\begin{bmatrix} 12 & -1 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 12 & -1 & -1 & 0 & 0 & 0 & \cdots & 0 \\ -1 & -1 & 12 & -1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & -1 & 12 & -1 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & -1 & -1 & 12 \end{bmatrix}$$

Macierz pasmowa A o rozmiarze 987x987

Zadanie B:

W zadaniu należało zaimplementować dwie metody iteracyjne: Jacobiego oraz Gaussa-Seidla, a następnie porównać ich czas wykonania oraz ilość iteracji do osiągnięcia oczekiwanego rezultatu. Algorytmy wykonują się dopóki norma z wektora residuum była większa niż 10^{-9}.

```
Jacobi method:
Time: 2.884342908859253 Number of iterations: 15

Gauss-Seidel method:
Time: 1.870460033416748 Number of iterations: 11
```

Wyniki w pythonie

Zadanie C:

Celem zadania było rozwiązać metodami iteracyjnymi równanie o takiej samej postaci jak w zadaniu A z macierzą pasmową A_2 wyglądającą następująco.

$$\begin{bmatrix} 3 & -1 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & \cdots & 0 \\ -1 & -1 & 3 & -1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & -1 & 3 & -1 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & -1 & -1 & 3 \end{bmatrix}$$

Macierz pasmowa A_2 o rozmiarze 987x987

W czasie działania algorytmu norma z residuum zamiast maleć zaczyna rosnąć do nieskończoności. Już przy ustawieniu maksymalnej ilości iteracji do 1000 występuje następujący błąd.

```
Jacobi method:
Time: 159.9047999382019 Number of iterations: 1000

Traceback (most recent call last):
  File "C:\Users\Paweł\PycharmProjects\pythonProject8\main.py", line 144, in <module>
    x, k = gauss_seidel(A, b, res)
  File "C:\Users\Paweł\PycharmProjects\pythonProject8\main.py", line 39, in gauss_seidel
    norm = math.sqrt(sum(r[i] ** 2 for i in range(n)))
  File "C:\Users\Paweł\PycharmProjects\pythonProject8\main.py", line 39, in <genexpr>
    norm = math.sqrt(sum(r[i] ** 2 for i in range(n)))
OverflowError: (34, 'Result too large')

Process finished with exit code 1
```

Overflow error

Powyższy błąd jest błędem przepełnienia, który powstał podczas obliczania normy wektora residuum, gdyż była ona zbyt duża. Możemy po tym wywnioskować, że metody zarówno Jacobiego, jak i Gaussa-Seidla nigdy się nie zbiegną.

Zadanie D:

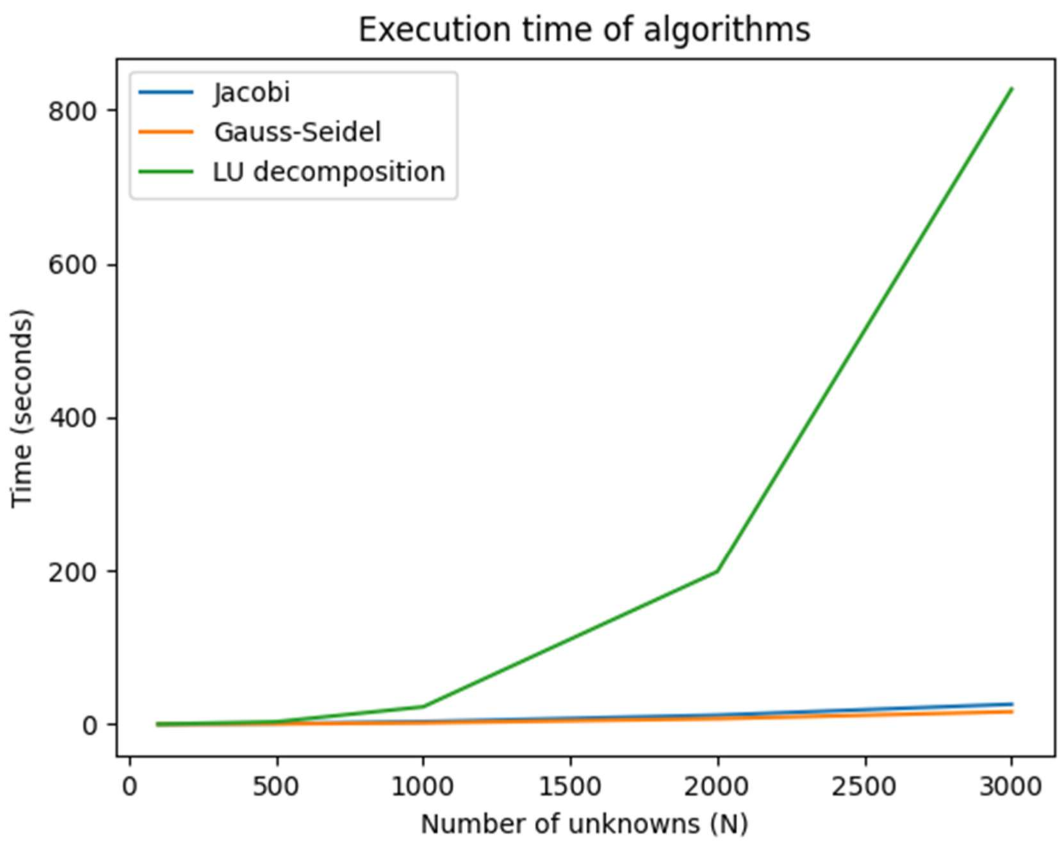
Celem zadania było zaimplementować metodę faktoryzacji LU do rozwiązania układu równań liniowych o postaci $A_2 x = b$, a następnie obliczyć normę residuum.

```
LU method:
Residual norm: 1.1234213631004928e-13
```

Norma wynosi $1.1234213631004928 \cdot 10^{-13}$. Jest to wartość bliska zeru, co oznacza że rozwiązanie jest niezwykle dokładne.

Zadanie E:

Poniżej przedstawiony jest wykres zależności czasu trwania poszczególnych algorytmów od liczby niewiadomych N = {100, 500, 1000, 2000, 3000} dla przypadku z zadania A.



Zadanie F:

W metodzie Jacobiego, obliczenia dla każdej iteracji są przeprowadzane niezależnie od siebie. To oznacza, że używamy tej samej macierzy x do obliczenia wszystkich elementów x_new. W związku z tym, nie zależy nam na kolejności obliczeń.

W przypadku metody Gaussa-Seidla, natychmiast aktualizowana jest wartość x_new i te aktualizacje są natychmiast uwzględniane w obliczeniach dla następnych elementów. To jest możliwe dzięki podziale macierzy A na L i U. Dla elementów L (poniżej przekątnej), używamy już zaktualizowanych wartości x_new, natomiast dla elementów U (powyżej przekątnej) używamy starych wartości x.

Ta różnica sprawia, że metoda Gaussa-Seidla zwykle zbiega szybciej niż metoda Jacobiego.

Metody iteracyjne są nieporównywalnie szybsze od bezpośredniej (faktoryzacji LU), jednak nie zawsze się zbiegają. Metoda bezpośrednia zwraca znacznie dokładniejsze rozwiązanie. W niektórych przypadkach metoda ta okaże się lepsza od iteracyjnych.