Oliwier Komorowski 197808

# Sprawozdanie

Metody Numeryczne Projekt 2 – Układy równań liniowych

# 1. Wstęp

## 1.1 Cel

Celem projektu jest implementacja i analiza dwóch metod iteracyjnych (Jacobiego i Gaussa-Seidla) oraz jednej metody bezpośredniej (faktoryzacja LU) rozwiązywania układów równań liniowych. Układ równań liniowych ma następującą postać: $Ax = b$, gdzie $A$ jest macierzą systemową, $b$ jest wektorem pobudzenia, natomiast $x$ jest wektorem rozwiązań reprezentującym szukaną wielkość fizyczną. Ważnym elementem algorytmów iteracyjnych (np. Jacobiego, Gaussa-Seidla) jest określenie w której iteracji algorytm powinien się zatrzymać. W tym celu najczęściej korzysta się residuum, czyli wektora, który dla k–tej iteracji przyjmuje postać: $r(k) = Ax(k) - b$. Badając normę euklidesową residuum (*norm(r(k))*), możemy w każdej iteracji algorytmu obliczyć jaki błąd wnosi wektor $x(k)$. Jeżeli algorytm zbiegnie się do dokładnego rozwiązania, to residuum stanowić będzie wektor zerowy. Ze względu na to, że metody iteracyjne praktycznie nigdy nie generują wektora residuum równego zero, podstawowe kryterium zakończenia obliczeń określane jest jako osiągnięcie normy residuum mniejszej niż zadana wartość, np. $10^{-6}$.

## 1.2 Założenia rozwiązywanego równania

Na potrzeby projektu przyjmuje:

- $N$ - rozmiar macierzy, ma wartość 1208 (1200 + 10$c$ + $d$, $c$ jest przedostatnią cyfrą numeru mojego indeksu, natomiast $d$ ostatnią),
- $A$ - macierz systemowa, tzw. macierz pasmowa (1) o rozmiarze $N \times N$ (diagonala główna z elementami a1, dwiema sąsiednimi z elementami a2 i dwoma skrajnymi z elementami a3),
- b - wektor pobudzenia, wektor o długości $N$, którego n−ty element ma wartość sin(n·8) (sin(n·($f$+1)), gdzie $f$ jest trzecią cyfrą mojego indeksu).

(1)
$$\mathbf{A} = \begin{bmatrix} a1 & a2 & a3 & 0 & 0 & 0 & 0 & \dots & 0 \\ a2 & a1 & a2 & a3 & 0 & 0 & 0 & \dots & 0 \\ a3 & a2 & a1 & a2 & a3 & 0 & 0 & \dots & 0 \\ 0 & a3 & a2 & a1 & a2 & a3 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & a3 & a2 & a1 \end{bmatrix}$$

# 2. Implementacja

## 2.1 Stosowane metody:

1) **Metoda Jacobiego**

Metoda Jacobiego definiuje proces iteracyjny w oparciu o równanie:

$x^{(k+1)} = -D^{-1}(L+U)\, x^{(k)} + D^{-1}b$, które można również zapisać jako: $x^{(k+1)} = M_J\, x^{(k)} + w_J$,

przy czym: A = L + U + D.

- *L* - macierz trójkątna dolna, która zawiera wszystkie elementy macierzy *A* poniżej głównej diagonalnej,
- *U* - macierz trójkątna górna, która zawiera wszystkie elementy macierzy *A* powyżej głównej diagonalnej,
- *D* - macierz diagonalna, która zawiera wszystkie elementy macierzy *A* z głównej diagonalnej.
- $x^{(0)}$- wektor, którego każdy element jest równy jeden,
- $10^{-9}$ - zadowalająca norma residuum.

2) **Metoda Gaussa-Seidla**

Metoda Gaussa-Seidla definiuje proces iteracyjny w oparciu o równanie:

$x^{(k+1)} = -(D + L)^{-1}(U\, x^{(k)}) + (D + L)^{-1}b$, które można również zapisać jako: $x^{(k+1)} = M_{GS}\, x^{(k)} + w_{GS}$.

**Przyjmuje takie same założenia i oznaczenia co przy metodzie Jacobiego.*

3) **Metoda LU**

Rozwiązanie równania macierzowego metodą LU składa się z dwóch głównych etapów obliczeń:

- Rozkład (dekompozycja, faktoryzacja) LU

Na tym etapie obliczeń wyznaczane są macierze trójkątne *L* i *U* oraz zwykle również macierz permutacji wierszy macierzy *A* zgodnie z zależnością: $A = P^T LU$

Macierz *L* jest macierzą trójkątną dolną, macierz *U* jest macierzą trójkątną górną, *P* jest macierzą permutacji.

- Rozwiązanie dwóch równań z macierzami trójkątnymi

a) podstawienie w przód (ang. forward substitution) z uwzględnieniem permutacji: $y = L^{-1}(Pb)$

b) podstawienie wstecz (ang. back substitution): $x = U^{-1}y$

Rozwiązanie równania macierzowego metodą LU oznacza wykonanie obu etapów obliczeń: wyznaczenie rozkładu LU oraz rozwiązanie równań z macierzami trójkątnymi.

## 2.2 Implementacja w kodzie

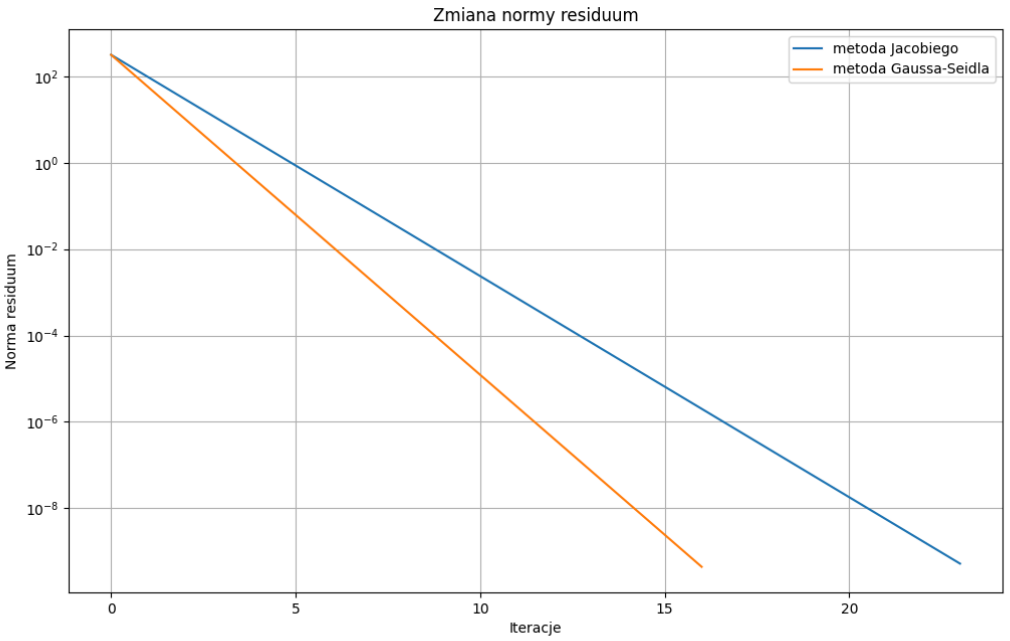
Do przeprowadzenia badania zaimplementowałem powyższe algorytmy w języku Python z użyciem biblioteki numpy.

# 3. Analiza

## 3.1 Układu równań z $A = A_1$

W badanym przypadku macierz $A_1$ przyjmuje wartości:

- a1 = 13 (5+*e*, gdzie *e* jest czwartą cyfrą mojego indeksu)
- a2 = a3 = -1



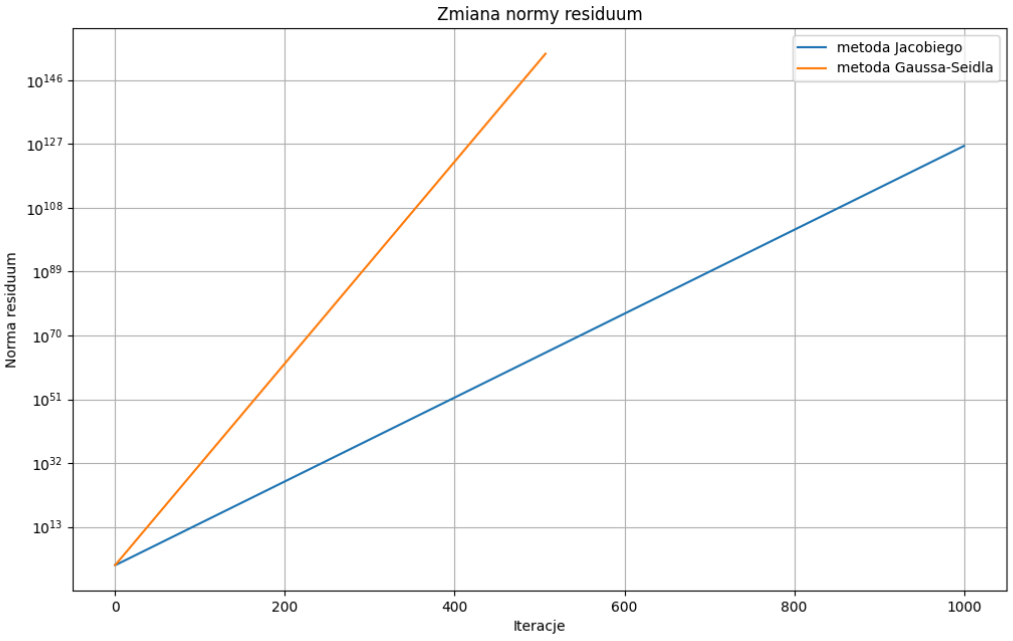

Udało się otrzymać satysfakcjonujący wynik za pomocą obu metod po odpowiednio:

- 23 iteracjach (ok. 0,02875 s) dla metody Jacobiego,
- 16 iteracjach (ok. 0,00923 s) dla metody Gaussa-Seidla.

## 3.2 Układu równań z $A = A_2$

Macierz $A_2$ przyjmuje wartości:

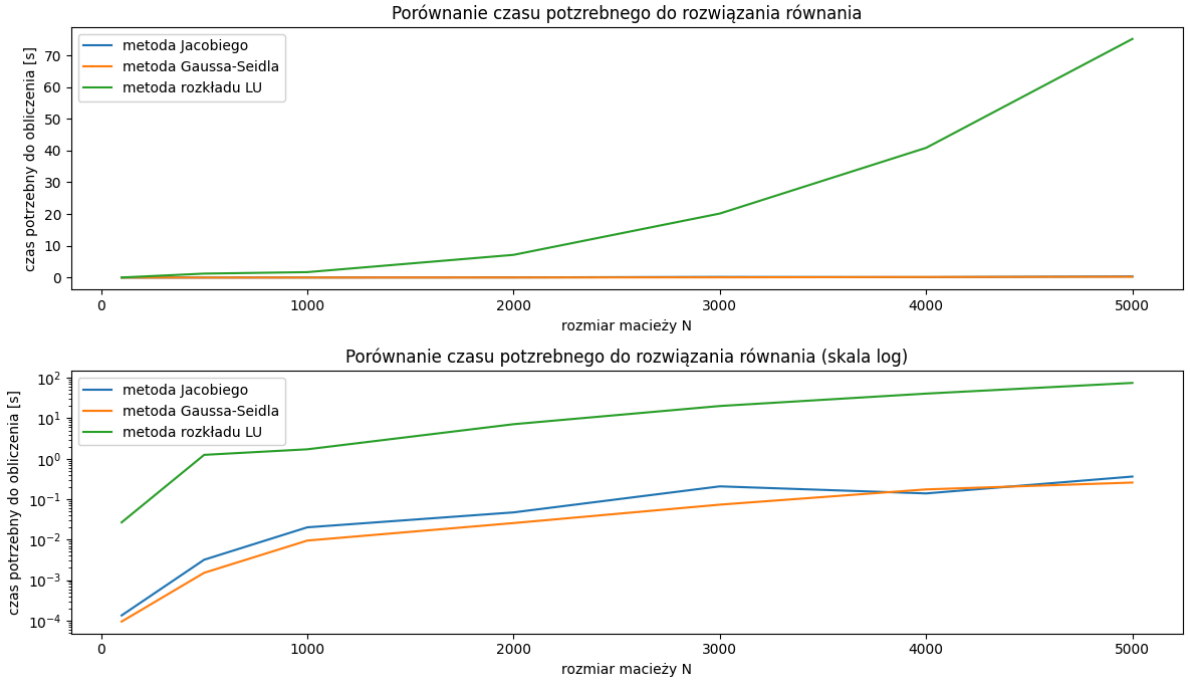
- a1 = 3 (5+*e*, gdzie *e* jest czwartą cyfrą mojego indeksu)
- a2 = a3 = -1



Po wykonaniu 1000 iteracji (granicy zabezpieczającej przed zapętleniem się programu) nie udało się osiągnąć satysfakcjonującego wyniku . Z wykresu widać, że, dla macierzy $A_2$, obie metody z każdą iteracją oddalały się od poprawnego wyniku. Metoda Gaussa-Seidla ok. 2 razy szybciej (po 500 iteracjach norma przekroczyła rozmiar float). Aby rozwiązać ten układ równań trzeba zastosować metodę LU. W tym przypadku algorytm ten zwraca wynik, którego norma residuum wynosi ok. $1{,}543 \cdot 10^{-13}$. Zajmuje to mu ok. 8,59 s.

## 3.3 Porównanie trzech algorytmów dla różnych rozmiarów macierzy.

Jak można zauważyć na poniższym wykresie algorytm LU zajmuje znacznie więcej czasu od dwóch poprzednich metod. Czas dla tej metody rośnie $n^3$.



# 4. Wnioski

1) Skuteczność metod iteracyjnych zależy od właściwości macierzy systemowej – dla dobrze uwarunkowanej macierzy (*A1*) obie metody iteracyjne (Jacobiego i Gaussa-Seidla) zbiegały się szybko do rozwiązania, przy czym metoda Gaussa-Seidla wykazała wyraźnie lepszą efektywność (mniej iteracji i krótszy czas działania).
2) Nie każda macierz nadaje się do rozwiązywania metodami iteracyjnymi – w przypadku macierzy *A2* metody iteracyjne nie zbiegały się do rozwiązania, a kolejne iteracje oddalały się od wyniku. Pokazuje to ograniczenia tych metod i konieczność analizy własności macierzy przed ich zastosowaniem.
3) Metoda LU, mimo większej złożoności obliczeniowej, gwarantuje stabilność i dokładność rozwiązania, nawet w przypadkach, gdzie metody iteracyjne zawodzą. Jest to jednak okupione wyraźnie większym czasem obliczeń, szczególnie dla dużych macierzy.
4) Złożoność czasowa algorytmów znacząco się różni – metody iteracyjne są znacznie szybsze przy dużych rozmiarach macierzy, o ile macierz jest odpowiednio uwarunkowana. Metoda LU natomiast staje się nieopłacalna czasowo przy dużych rozmiarach.
5) Dobór metody powinien być uzależniony od charakterystyki macierzy oraz wymagań co do dokładności i szybkości – w praktyce często warto rozpocząć od metody iteracyjnej i dopiero w razie potrzeby przechodzić do metody bezpośredniej.