Kacper Puda s188625

# Metody Numeryczne Projekt 2
# Układy równań liniowych

## Wstęp

Celem projektu jest implementacja metod iteracyjnych (Jacobiego i Gaussa-Seidla) i bezpośrednich (faktoryzacja LU) rozwiązywania układów równań liniowych. Ich celem jest znalezienie rozwiązania układu równań liniowych $Ax = b$, gdzie A to macierz współczynników, x to wektor niewiadomych, a b to wektor wyrazów wolnych.

Metody Jacobiego i Gaussa-Seidla polegają na iteracyjnym poprawianiu przybliżonego rozwiązania początkowego, aż do osiągnięcia dostatecznie dokładnego wyniku.

Wzór metody Jacobiego:

$$x^{(k+1)} = -D^{-1}(L + U)\,x^{(k)} + D^{-1}\,b$$

Wzór metody Gaussa-Seidla:

$$x^{(k+1)} = -(D + L)^{-1}\,(U\,x^{(k)}) + (D + L)^{-1}\,b$$

gdzie macierz A została rozłożona na macierz trójkątna dolną L, macierz trójkątną górna U oraz macierz diagonalna D.

$$A = L + U + D$$

Metoda bezpośrednia (metoda faktoryzacji LU) nie jest metodą iteracyjna. Zamiast tego polega na rozkładzie macierzy współczynników układu równań na dwie macierze trójkątne: macierz dolnotrójkątną L i macierz górną trójkątną U. Następnie można rozwiązać układ równań poprzez zastosowanie algorytmu rozwiązywania układów równań trójkątnych (forward substitution (do rozwiązywania układów równań z macierzą dolnotrójkątną) i backward substitution (do rozwiązywania układów równań z macierzą górną trójkątną)). Ta metoda jest bardziej dokładna i nie ma ryzyka oscylacji, ale może być bardziej kosztowna obliczeniowo niż metody iteracyjne.

Dużym błędem jest wyznaczenie macierzy odwrotnej, ponieważ operacja ta jest kosztowna numerycznie, może powodować znaczne błędy numeryczne oraz może wywołać znaczny wzrost zapotrzebowania na pamięć RAM. Zamiast odwracania macierzy należy stosować podstawienie w przód (ang. forward substitution)

Ważnym elementem algorytmów iteracyjnych jest warunek zakończenia obliczeń. Zwykle zawiera on sprawdzenie czy norma wektora rezydualnego jest mniejsza od założonego progu dokładności obliczeń, my w zadaniach przyjelismy $10^{-9}$.

Błąd rezydualny res w k-tej iteracji :

$$res^{(k)} = Ax^{(k)} - b$$

Do realizacji projektu wykorzystałem język ***pyhon***, z wykożywstaniem bibliotek:

- ***math*** - do działań matematycznych (np. $sin(x)$)
- ***matplotlib*** - do stworzenia wykresu
- ***time*** - do mierzenia czasu wykonywania algorytmów

Dla mojego indeksu: 188625, c = 2, d = 5, wynika z tego, że obliczenia przeprowadzane w projekcie są wykonywane dla N = 925.

# Realizacja zadań

## Zad. A

Dla mojego indeksu: 188625, e = 6, zatem a1 = 11, f = 8, zatem n-ty wyraz wektora b jest równy $sin(n*9)$.

## Zad. B

Wyniki metod Jacobiego i Gaussa-Seidla dla układu równań z zadania A (warunek zakończenia to norm(res) < $10^{-9}$):

### Metoda Gaussa-Seidla:

- Liczba iteracji: 18
- Czas wykonywania: 17.875 [s]
- Norma błedu rezydualnego: 3.670703480574952e-10

### Metoda Jacobiego:

- Liczba iteracji: 26
- Czas wykonywania: 25.734 [s]
- Norma błedu rezydualnego: 7.972718901661565e-10

### Wnioski do zadania B:

Jak da się zauważyc metoda Gaussa-Seidla wykonała się szybciej o 7,889 sekundy szybciej od metody Jacobiego. Dodatkowo, metoda Gaussa-Seidla potrzebował 8 iteracji mniej, aby osiągnąć satysfakcjonujący wynik. Oznacza to, że metoda Gaussa-Seidla jest bardziej efektywna niż metoda Jacobiego w tym konkretnym przypadku, jeśli chodzi o czas wykonywania i liczbę iteracji potrzebnych do uzyskania rozwiązania o wystarczającej dokładności.

Jednak metoda Jacobiego w swoim wzorze potrzebuje zastosowania forward substitution tylko dla macierzy diagonalnej co jest szczególnym przypadkiem, który (stosując nieco zmieniony algorytm) można przyspieszyć:

### Metoda Jacobiego z forward substitution dla macierzy diagonalnej:

- Liczba iteracji: 26
- Czas wykonywania: 19.594
- Norma błedu rezydualnego: 7.972718901661565e-10

W takim przypadku metoda Jacobiego jest szybsza o 6,14 sekundy, co jest lepszym wynikiem.

Jednak w kolejnych zadaniach dla metody Jacobiego będę stosować standardowy algorytm forward substitution.

## Zad. C

Dla układu równań z zadania C (a1 = 3, a2 = a3 = -1, N i b bez zmian) dla poszczególnych metod otrzymałem następujące wyniki:

### Metoda Gaussa-Seidla:

- Liczba iteracji: 513
- Czas wykonywania: 523.875 [s]
- Norma błedu rezydualnego= inf

### Metoda Jacobiego:

- Liczba iteracji: 1234
- Czas wykonywania: 1280.016 [s]
- Norma błedu rezydualnego = inf

### Wnioski do zadania C:

Norma błedu rezydualnego dla metod Jacobiego i Gaussa-Seidla rosną do nieskończoności, jednak dla metody Gaussa-Seidla rośnie dużo szybciej (w tym przypadku warunkiem skończenia wykonywania iteracji algorytmów był wyjątek kompilatora: OverflowError). Wynika z tego, że normy błędów dla tych metod nie zbiegają się. Metody Jacobiego i Gaussa-Seidla są skuteczne tylko dla określonych typów macierzy, a dla innych mogą nie działać. W związku z tym, nie można uznać ich za uniwersalne metody rozwiązywania układów równań liniowych

## Zad. D

Dla zadania D polegającym na zaimplementowaniu metody bezpośredniego rozwiązania układów równań liniowych: metodę faktoryzacji LU dla przypadku z zadania C, otrzymałem następujący wynik:

### Metoda bezposrednia:

- Czas wykonywania: 1294.047 [s]
- Norma błedu rezydualnego: 2.7136589340820284e-15

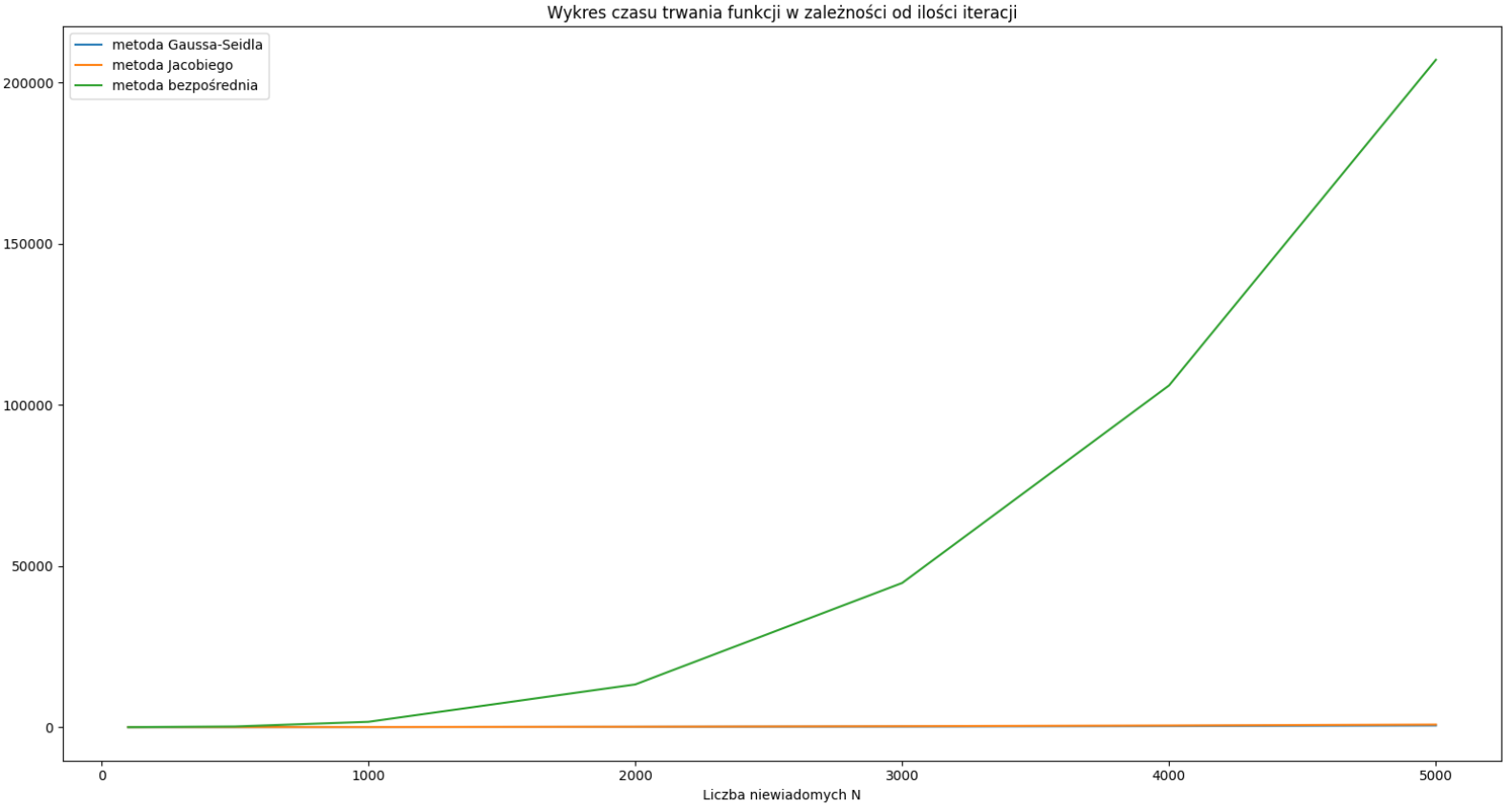
### Wnioski do zadania D:

Norma błędu rezydualnego dla metody bezpośredniej wyniosła 2. 2.7136589340820284e-15, co jest dokładniejszym wynikiem niż założony przez nas warunek stopu dla metod iteracyjnych ($10^{-9}$). Pomimo długiego czasu wykonywania algorytmu wynoszącego 21 minut i 34,047 sekund, otrzymaliśmy poprawny i dokładniejszy wynik. Dowodzi to, że metoda bezpośrednia, mimo że czasochłonna, jest uniwersalna i może być stosowana do dowolnych macierzy, co odróżnia ją od metod iteracyjnych, które wymagają odpowiednio dobranych macierzy, aby działać poprawnie.

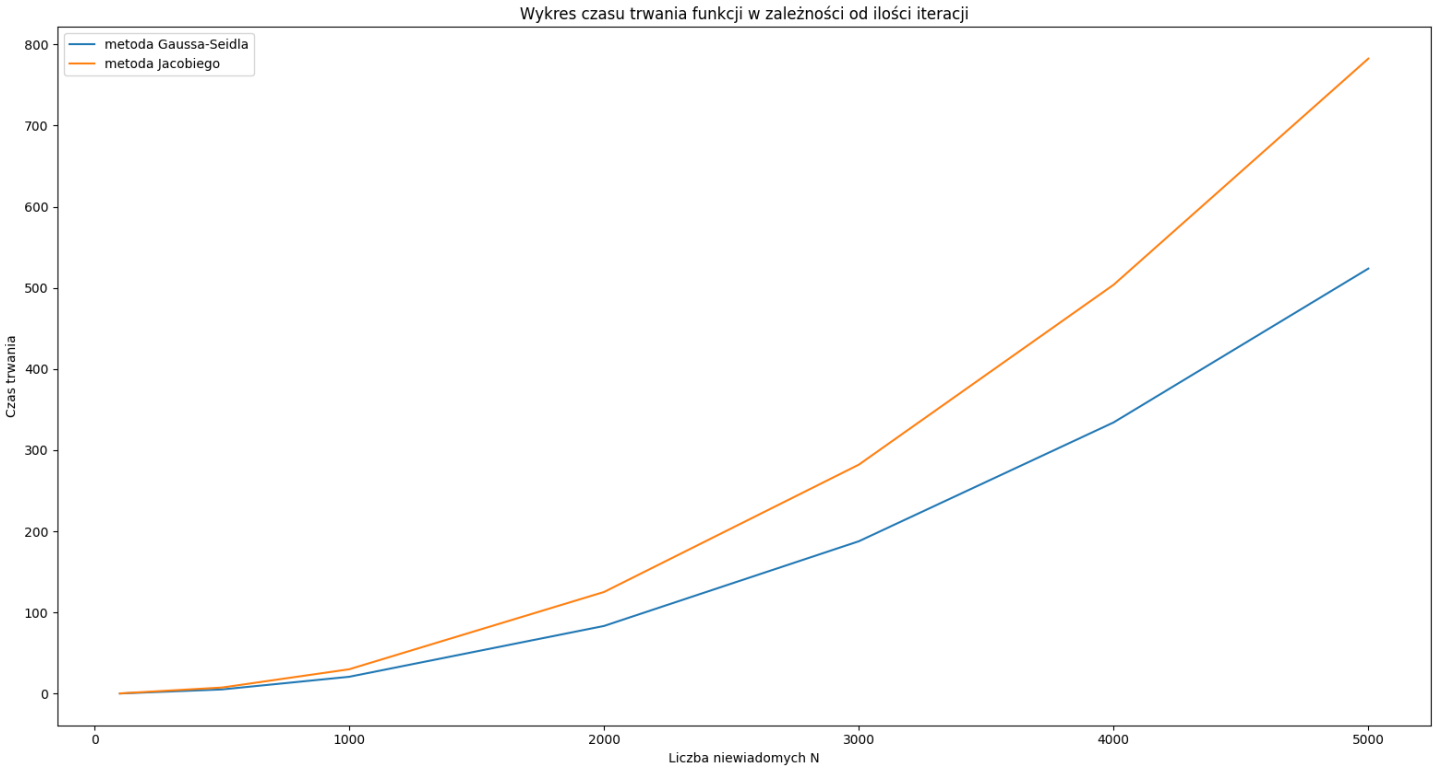
## Zad. E

Dla N = {100, 500, 1000, 2000, 3000, 4000, 5000} dla układu równań z przypadku z punktu A otrzymałem nasępujący wykres:

wykres 1 metody: bezpośrednia, Jacobiego, Gaussa-Seidla



wykres 2 - tylko metody Jacobiego i Gaussa-Seidla




Tworząc wykres 1 pominąłem wykonywanie metody bezpośredniej dla N = {2000,3000,4000,5000}, ponieważ juz dla przypadku N = 1000 widać znaczną róznice w czasie wykonywania, a w zaimplementowany przezemnie sposób obliczenia trwałyby zbyt długo. Zamiast tego dodałem obliczone przeze mnie przybliżone czasy trwania których moglibyśmy sie spodziewać dla danych wielkości N.

### Wnioski do zadania D:

Dla każdego z algorytmów czas trwania rośnie wraz ze zwiększaniem liczby niewiadomych N. Metoda bezpośrednia, pomimo swojej dokładności, jest najwolniejsza ze wszystkich badanych metod. Metoda Gaussa-Seidla okazała się szybsza niż metoda Jacobiego dla każdego z badanych przypadków. Warto jednak zwrócić uwagę, że w przypadku metod iteracyjnych czas ich wykonania jest znacznie krótszy niż dla metody bezpośredniej.

## Zad. F

Wszystkie przedstawione metody wraz ze wzrostem niewiadomych wykonują się dłużej. Widać znaczną przewage algorytmów iteracyjnych nad metodą bezpośrednią, jednak nie są one uniwersalne i potrzebują określonych macierzy aby działać poprawnie, a metoda bezpośrednia jest dokładniejsza i może działać na dowolnej macierzy, jednak jej czas wykonywania jest zdecydowanie gorszy.