Metody numeryczne - Projekt nr 2

Układy równań liniowych

Wiktor R.Wojtyna 192581 Informatyka Semestr 4, Grupa 4, 04.04.2024 r.

Wprowadzenie

Celem projektu jest zaimplementowanie i analiza wydajności metod iteracyjnych i bezpośrednich do rozwiązywania układów równań liniowych. Projekt implementuje i analizuje dwa metody iteracyjne (Jacobi i Gaussa-Seidela) oraz jedną metodę bezpośrednią (dekompozycję LU) do rozwiązywania układów równań liniowych. Do zrealizowania zadania użyłem języka CPP i następujących bibliotek:

1. **iostream**

2.**vector**

3.**cmath**

4.**chrono**

5.***matplotlib-cpp***

> ***matplotlib-cpp*** wykorzystuje, celem osiągnięcia funkcjonalności matplotlib z języka python, następujące biblioteki (z wyjątkiem vector i iostream nie są one wykorzystywane w mojej implementacji zadania:)
>
> -**vector**,map,array,numeric,algorithm, stdexcept, iostream, cstding, functional, string,
>
> **iostream**

Zadanie A

Dla numeru indeksu 192581 otrzymujemy wartości: a1 = 10, oraz N = 981, natomiast n−ty element wektora b ma wartość sin(n ·(3)).

Zadanie B

Wyniki metod Jakobiego i Gaussa-Seidela dla układu z punktu A

Obliczenia przerwano gdy norma z wektora residuum osiągnęła wartość niższą niż 10^{-9}

==========

Jacobi method:

number of iterations: 29

norm of residual vector: 5.36805e-10

duration [ms]: 72.1561

Gauss-Seidel method:

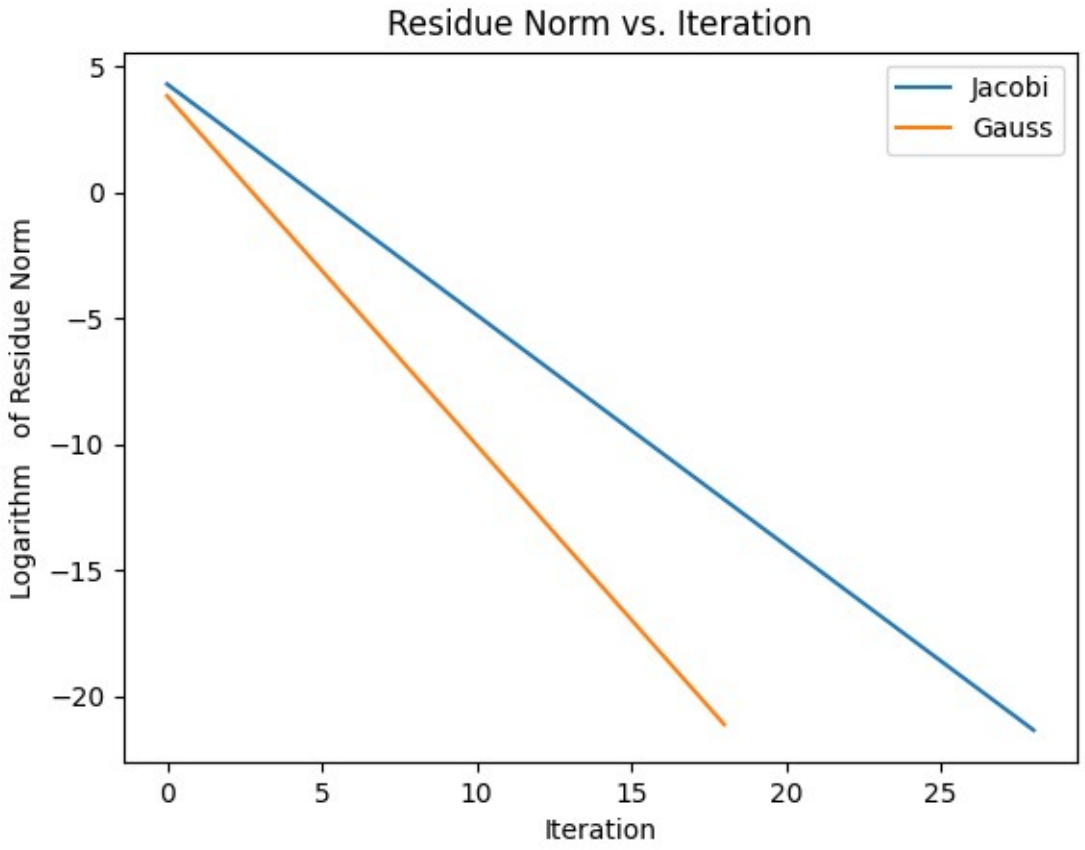
number of iterations: 19

norm of residual vector: 6.75193e-10

duration [ms]: 30.8104

==========

Jak widać metoda Gaussa-Seidela osiąga ten sam rząd dokładności w około 15% krótszym czasie, przy liczbie iteracji mniejszej o 1/3. Pozwala to sądzić, iż metoda Gaussa-Seidela jest szybsza, oraz dla każdej poszczególnej iteracji potrzebuje ona mniej czasu(2.48ms/iterację dla Jacobiego, 1.57ms/iterację dla G-S). Gauss-Seidel osiąga również nieco mniejszą (więc lepszą)liczbę znaczącą – jednak zarówno to, jak i rozbieżność w czasie na iterację mogą być dziełem przypadku.



Zadanie C

Gdy podmienimy wartość a1 na 3 otrzymamy drastycznie inne wyniki:

==========

Jacobi method:

number of iterations: 2464

Warning: Residual norm became NaN.

duration [ms]: 7368.5

Gauss-Seidel method:

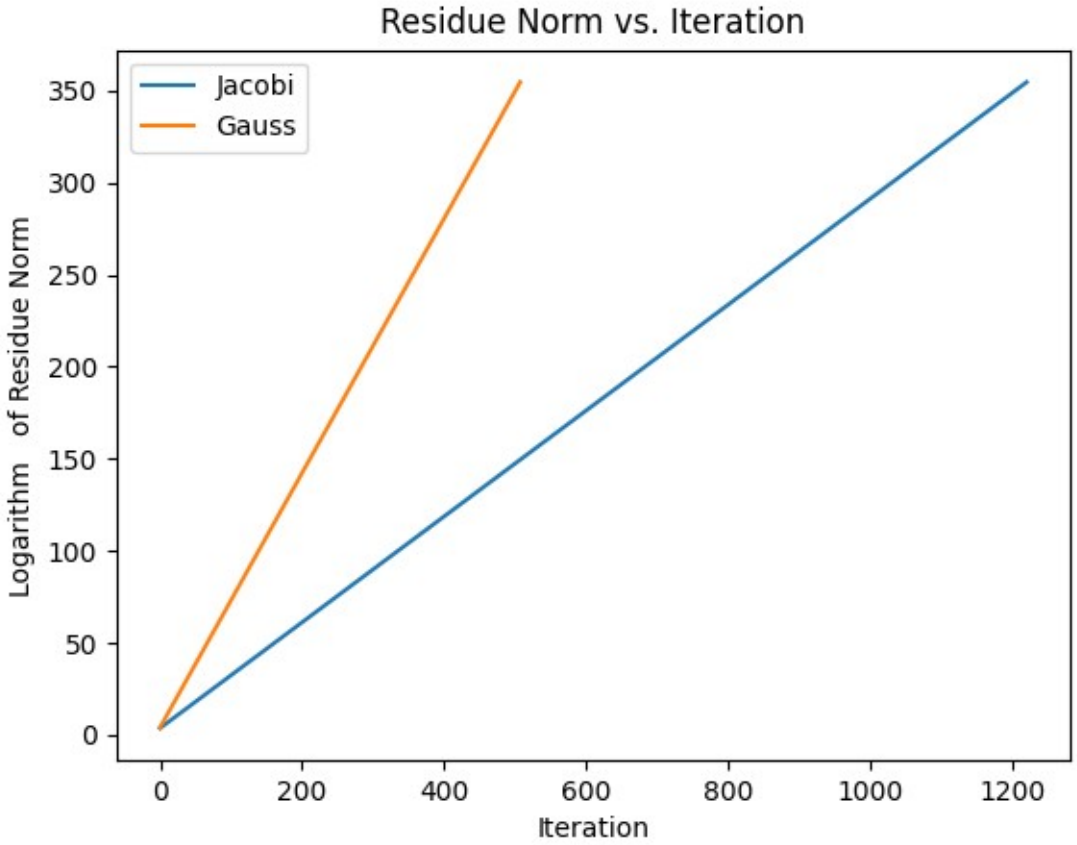
number of iterations: 1046

Warning: Residual norm became NaN.

duration [ms]: 1943.58

==========

Wynik ten oznacza, iż metody iteracyjne dla takich wartości elementów macierzy A **nie** zbiegają się. Dominacja diagonalna – pożądana własność dla metod iteracyjnych, takich jak metoda Jacobiego i Gaussa-Seidela jest wprost proporcjonalna do wartości a1, wobec czego taki obrót spraw jest poniekąd zgodny z oczekiwaniami. Tutaj potwierdza się również nasza hipoteza o przypadkowości stosunku czasu do iteracji w poprzednim zadaniu – tutaj mamy 2.96ms/iterację dla Jacobiego, 1.82ms/iterację dla G-S. Gauss-Seidel pozostaje szybszy.\



Zadanie D

Zaimplementowano metodę bezpośredniej faktoryzacji LU i zastosowano do danych zadania C

==========

LU decomposition method:

norm of residual vector: 1.08486e-13

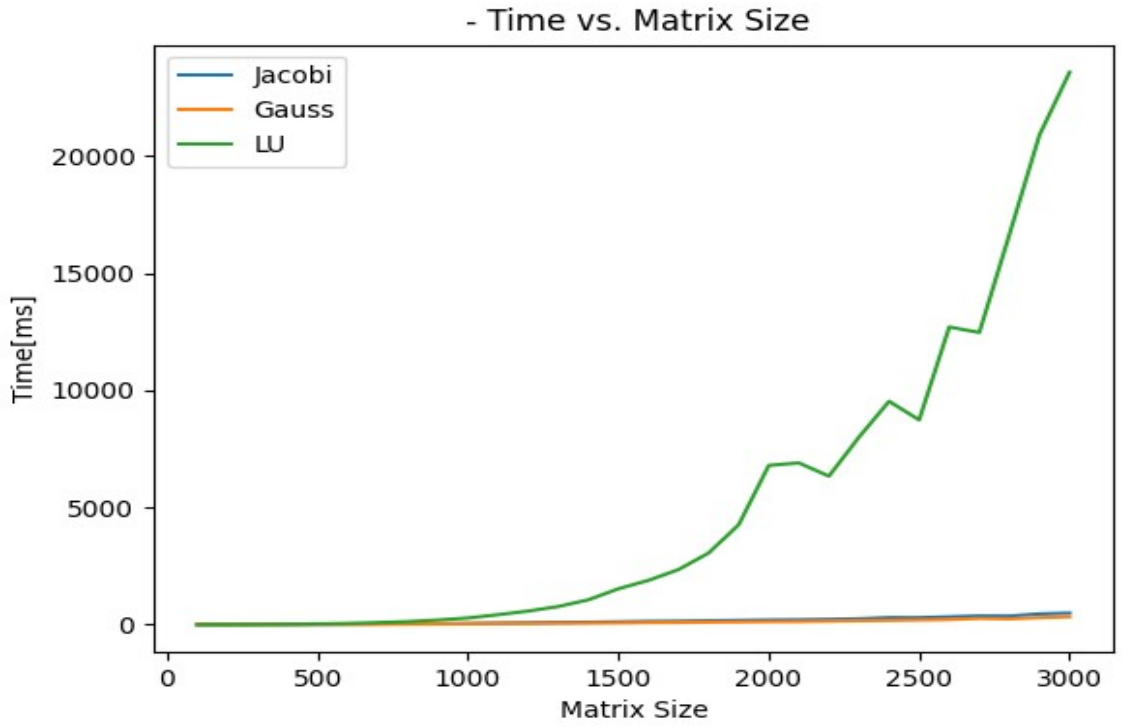
duration [ms]: 293.705

==========

Jak zatem widać dla tego układu rozwiązanie można znaleźć pomimo jego rozbieżności dla metod iteracyjnych. Warte uwagi jest również to, iż osiągnęliśmy tutaj bardzo dobry – do bardzo niski rząd wielkości, o 5 rzędów niższy niż w zadaniu B – i odbyło się to w zadowalającym czasie, porównywalnym z czasami metod Jakobiego i G-S w zadaniu B, oraz znacznie krótszym niż czas, którego potrzebowały owe metody w zadaniu C na rozpoznanie rozbieżności(zapewne można by owy czas skrócić za pomocą jakiejś mechaniki w kodzie)

Zadanie E

Wytworzyłem wykres czasu trwania poszczególnych metod w zależności od liczby niewiadomych

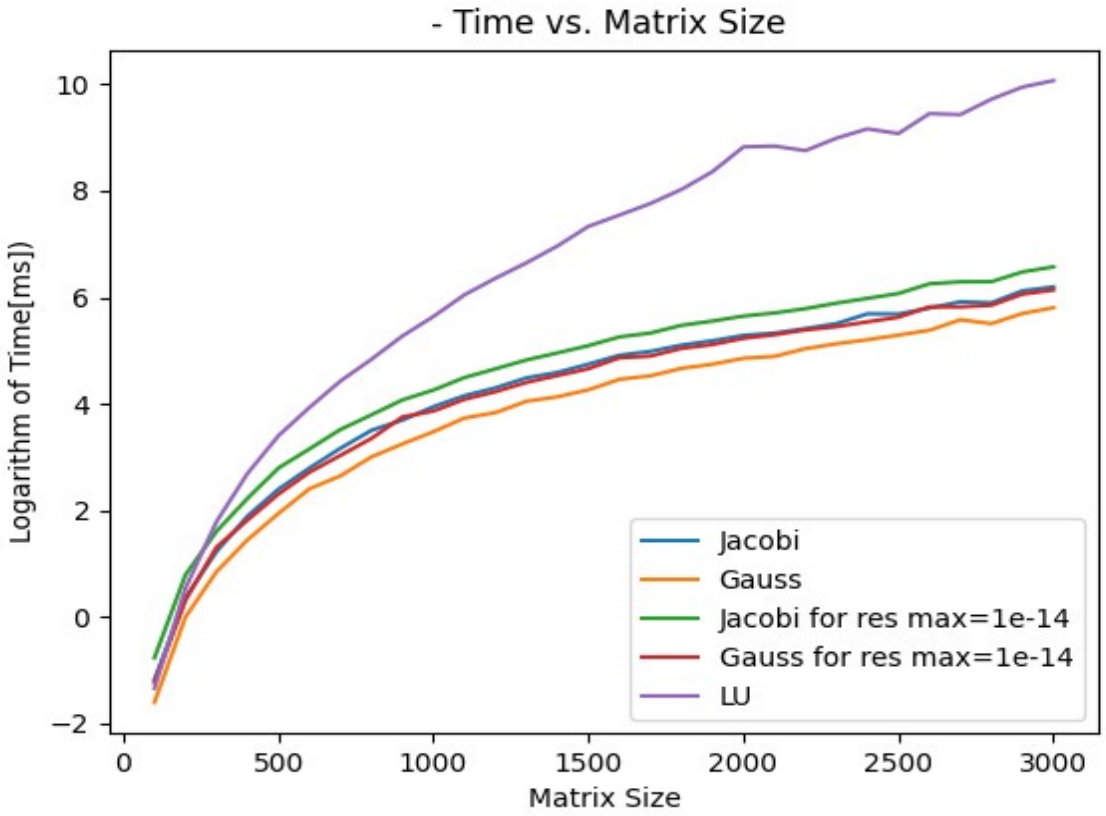


{100,200,300,400,500...3000} , pozostałe dane pozostają identyczne z punktem A.

Dla każdego z algorytmów czas trwania rośnie wraz ze zwiększaniem liczby niewiadomych N, czyli ze wzrostem rozmiaru macierzy. Już tu możemy zauważyć, że metoda faktoryzacji LU jest znacznie wolniejsza od metod iteracyjnych – czas rośnie $O(n^3)$, natomiast dla metod iteracyjnych jest to $O(n^2)$. Zastanawiające stają się jednak punkty dla rozmiarów 2000,2400,2600 – gdzie odnotowujemy nagłe spowolnienie- co może wynikać z czasu dostępu do procesora.

Zadanie F

Dla dokładniejszego omówienia sprawy dokonano dodatkowych obliczeń i wykonano poniższy wykres, który celem większej czytelności korzysta z skali logarytmicznej.



Dla wszystkich trzech metod, czas wykonywania obliczeń wzrasta wraz z liczbą niewiadomych. Metoda Gaussa-Seidla oraz Jacobiego są jednak szybsze od metody faktoryzacji LU – są tym szybsze im większy jest rozmiar macierzy. W naszym eksperymencie metoda faktoryzacji LU osiągała wartości rzędu e-15, wobec czego dokonałem obliczeń dla Jakobiego i G-S z tym wymaganiem, by osiągnęli ten sam rząd wielkości – i metody te dokonały tego w czasie znacznie niższym niż metoda faktoryzacji. Metoda LU ma jednak swoje bardzo mocne strony – co zaobserwowaliśmy w zadaniach C i D – mimo swojej czasochłonności jest ona niezawodna, co w niektórych przypadkach może być ważniejsze od prędkości.

W oczy rzuca się zbieżność na wykresie linii Jacobiego dla res max=1e-9 i Gaussa dla res max=1e-14 – Metoda Gaussa powtarzalnie daje lepsze wyniki czasowe od metody Jakobiego – dzięki czemu zdobywa ona miano najszybszej z tych trzech metod , a my potwierdzamy kolejną z naszych hipotez z zadania B.

W ogólności zdaje się, że ciężko jest jednoznacznie stwierdzić czy lepsza jest metoda Gaussa-Seidla czy metoda Faktoryzacji LU, natomiast łatwo można stwierdzić, że obie mają lepsze walory niż metoda Jakobiego. Gdybyśmy mieli bowiem chcieć użyć metody Jacobiego lepiej od razu użyć Gaussa-Seidla. Rozbieżność ta najpewniej wynika z tego, iż podczas gdy w metodzie Jakobiego aktualizujemy od razu cały wektor na podstawie jego poprzedniego stanu, tak w G-S aktualizujemy jego pojedyncze wartości biorąc pod uwagę również te zaktualizowane przed momentem – co prowadzi do większego dynamizmu wektora, więc i do szybszej zbieżności.