Juliusz Radziszewski
Informatyka, sem IV
s193504, grupa 1

Metody Numeryczne - Projekt 2 - Układy równań liniowych

1. Wstęp

Celem omawianego projektu jest zbadanie działania metod rozwiązywania układów równań liniowych. Pod lupę wzięte będą dwa typy metod - iteracyjne (tj. metoda Jacobiego i Gaussa-Seidla) i bezpośrednie (fraktoryzacja LU). Te iteracyjne polegają na stopniowym zbliżaniu się wyników cząstkowych do rozwiązania rzeczywistego. Algorytm jest przerywany wtedy, kiedy norma błędu residualnego osiągnie satysfakcjonującą dla nas niską wartość (kryterium stopu). Fraktoryzacja LU jest metodą bezpośrednią, czyli taką, która pozwala na uzyskanie wyniku po określonej liczbie działań arytmetycznych.
Rozpatrywany przez nas układ równań liniowych ma postać:

$$\mathbf{Ax = b}$$

gdzie A to macierz pierwotna (systemowa), x to wektor rozwiązań, a b to wektor pobudzenia

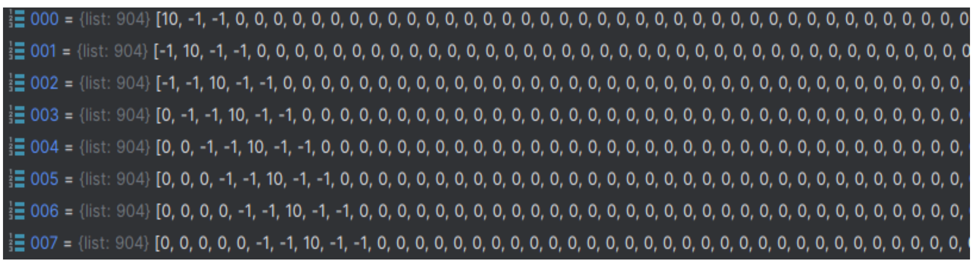
$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{B}.$$

Aby określić jaki błąd wnosi obliczone przez nas rozwiązanie w trakcie trwania algorytmów iteracyjnych, należy obliczyć tak zwany wektor residuum. Obliczając jego normę euklidesową uzyskujemy wartość naszego błędu. Najczęściej kryterium stopu ustala się na poziomie 10^{-6} lub 10^{-9}

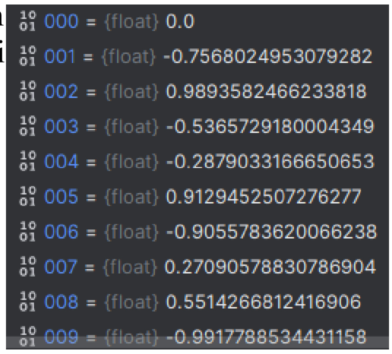
$$\mathbf{res^{(k)} = Ax^{(k)} - b}$$

2. Zadanie A

Celem pierwszego zadania było zdefiniowanie i utworzenie składowych do rozwiązania równania macierzowego. Długość macierzy i wektora jest z góry ustalona i zależna od numeru indeksu. W moim przypadku N = 904. Nasza macierz pierwotna przyjęła postać macierzy diagonalnej o rozmiarze N x N, która posiada na swoich przekątnych z góry ustalone, małe wartości (a1 = 3 - główna przekątna, a2 = a3 = -1 - przekątne poboczne). Co za tym idzie, zdecydowana większość pól w macierzy wynosi 0, a więc jest to macierz rzadka. b jest wektorem o długości N, gdzie jego kolejne elementy przyjmują wartość sin(4n), czyli od -1 do 1.



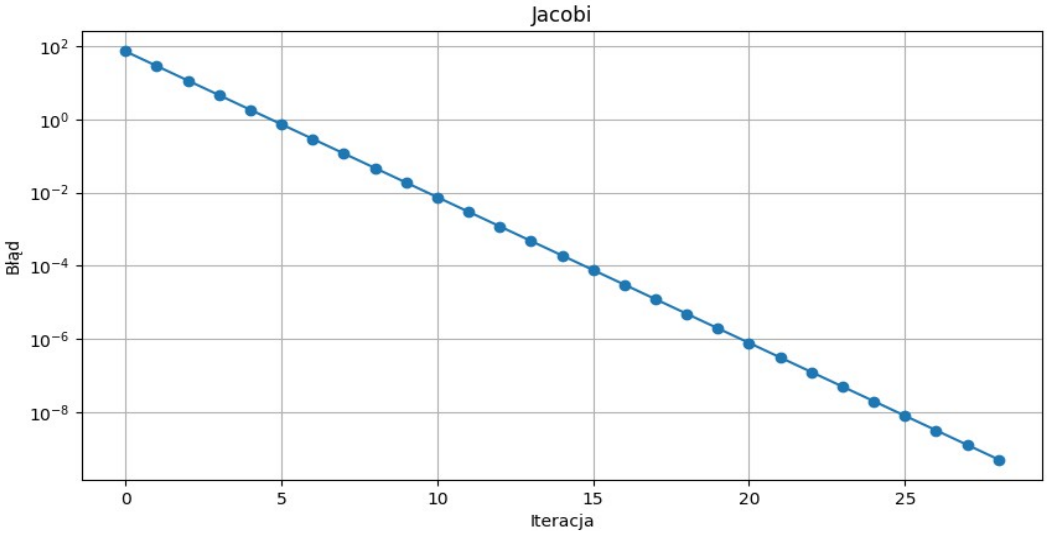
macierz A



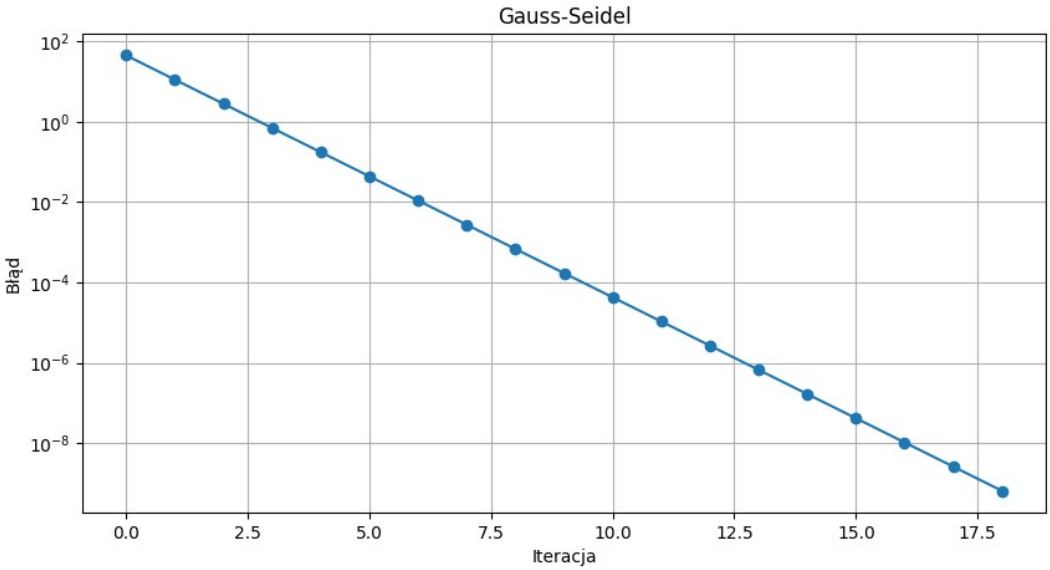
wektor B

3. Zadanie B

Celem zadania była implementacja obu metod iteracyjnych rozwiązywania równań macierzowych - metoda Jacobiego i Gaussa. Wszystkie funkcje odpowiedzialne za mnożenie macierzy, wektorów, sumowanie macierzy, mnożenie przez skalar, były zaimplementowane ręcznie, a więc również macierzą były listy list. Implementacja wykonana została w języku Python, co za tym idzie, naturalnie prędkość wykonywania obliczeń nie będzie tak szybka jak w języku C++. Poniżej zaprezentowane są wykresy błędu residualnego w zależności od liczby iteracji oraz czas obliczeń dla obu rozpatrywanych metod. Kryterium stopu zostało ustalone na wartości 10^{-9}.



0:00:06.306665

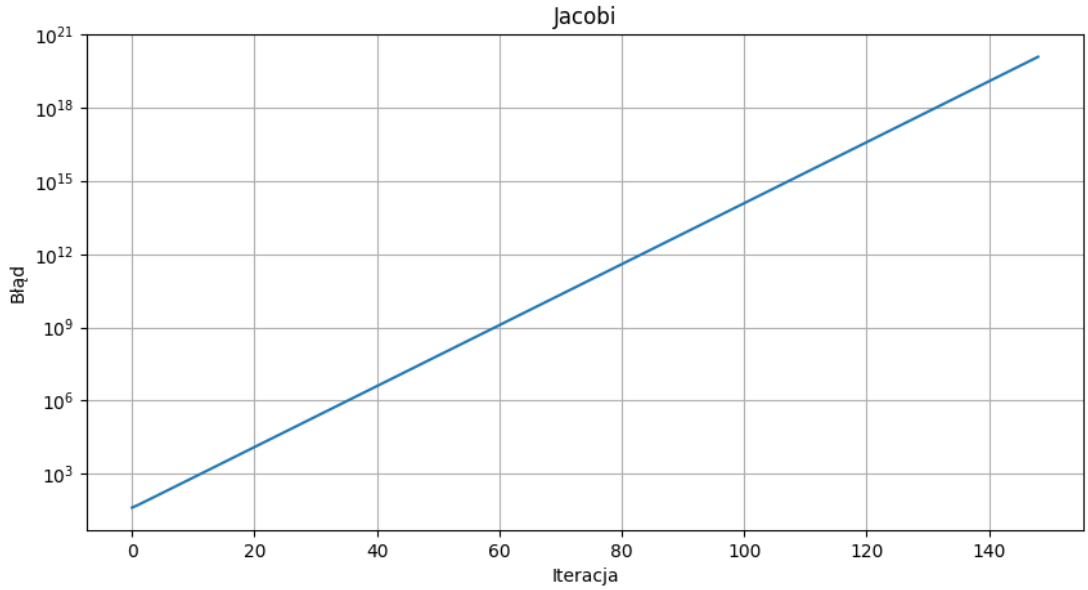


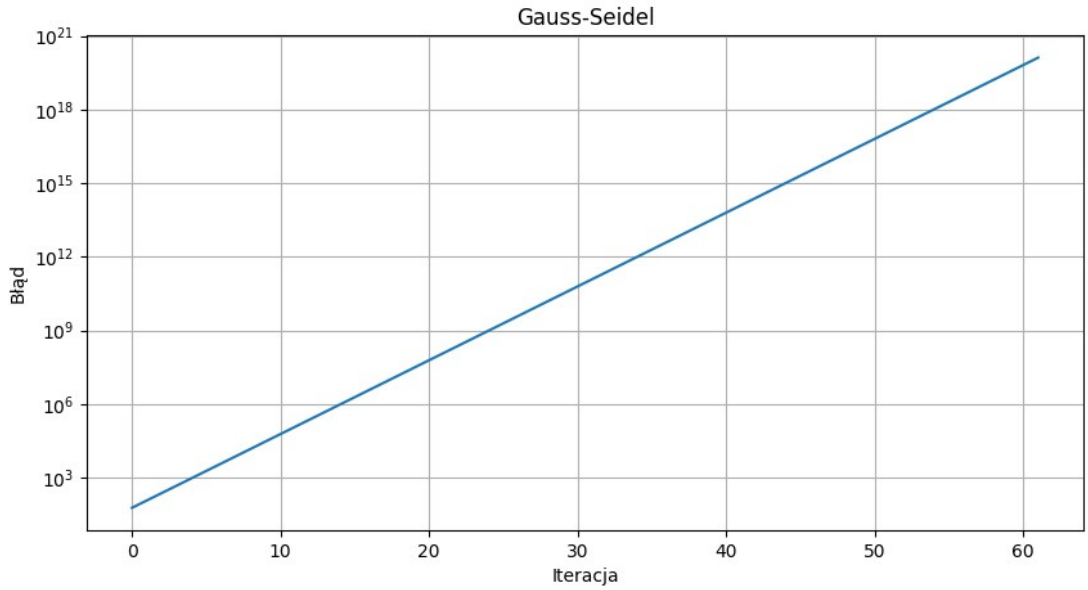
0:00:02.894151

Jak możemy zauważyć, metoda Gaussa-Seidla jest zdecydowanie (ponad dwukrotnie) szybsza dla macierzy o rozmiarach 904 x 904. Wymaga również mniej, bo jedynie 19 iteracji w przeciwieństwie do 29 przy metodzie Jacobiego. Kolejną obserwacją jest to, że norma błędu residualnego zmniejsza się dla obu metod w tempie logarytmicznym (wykresy na górze są właśnie w takiej skali, tak więc wykres liniowy potwierdza logarytmiczne tempo spadku).

4. Zadanie C

Naszym zadaniem było ponowne rozpatrzenie wyników dla napisanych przez nas algorytmów, lecz teraz dla nieco zmienionych danych układu równania macierzowego. W tym wypadku a1 = 3, a2 = a3 = -1. Oto wyniki dla wykonanych obliczeń:





Można łatwo zauważyć, iż tym razem norma residuum nie dąży do 0, lecz do nieskończoności dla obu rozpatrywanych metod. Wzrost ten jest również w tempie logarytmicznym. Dodatkowym przyjętym kryterium stopu ustalona została wartość 10^{20}. Ta została osiągnięta dla metody Jacobiego przy 149 iteracjach, a dla metody Gaussa-Seidla przy 62. Okazuje się bowiem, że metody iteracyjne nie zawsze dają dobry wynik dla określonych danych początkowych układu równań macierzowych. Aby metoda Gaussa-Seidla zbiegała się, macierz A musi być symetryczna, dodatnio określona oraz diagonalnie dominująca (ta właśnie cecha została w tym przypadku naruszona, gdyż w niektórych wierszach suma wartości bezwzględnych poza główną przekątną była większa od tych na tej przekątnej). Metoda Jacobiego również wymaga, by macierz A była diagonalnie dominująca, stąd również tam wystąpił owy błąd. Tym samym nie zawsze powinno się stosować metody iteracyjne.

5. Zadanie D

Kolejnym zadaniem była implementacja metody bezpośredniego rozwiązania układów równań liniowych, czyli metody fraktoryzacji LU.
Na początku należy znaleźć macierze L - macierz dolną trójkątną oraz U - macierz górną trójkątną, które później przydadzą się do dalszych obliczeń.

Schemat działania algorytmu:
1. Tworzenie macierzy L, U takich, że **LUx = b**
2. Utworzenie wektora pomocniczego **y = Ux**
3. Rozwiązywanie układu równań **Ly = b** (podstawienie w przód)
4. Rozwiązywanie układu równań **Ux = y** (podstawienie wstecz)

```
def solveLU(A, b):
    start_time = datetime.now()
    L, U = LU_Factorization(A)
    y = forward_substitution(L, b)
    x = backward_substitution(U, y)
    end_time = datetime.now()
    time = end_time - start_time
    norm_residual = errNorm(A, x, b)
    return norm_residual, time
```

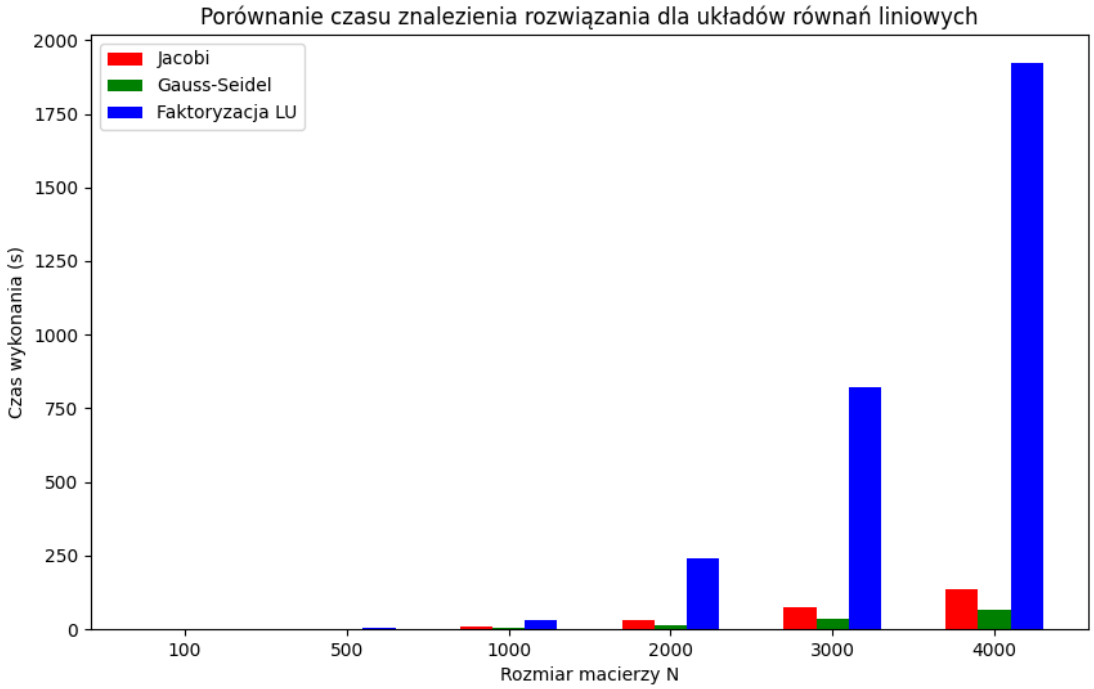
Dla danych z zadania C otrzymane wyniki to:

```
LU factorization
Norma residuum:  1.5019583493569575e-13
Czas obliczeń:  0:00:22.172244
```

Tak więc mimo tego, że metody iteracyjne nie dały nam dobrego rozwiązania, metoda bezpośrednia bardzo dobrze poradziła sobie z zadaniem, i w nieco ponad 22 sekundy znalazła praktycznie idealne rozwiązanie (gdyż błąd wynosi jedynie $1.5 * 10^{-13}$).

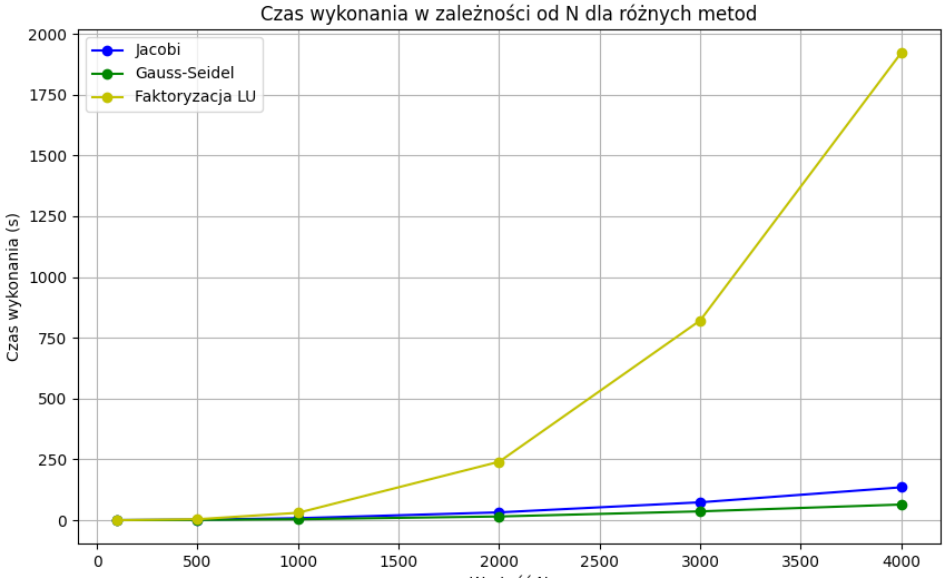
6. Zadanie E

W tym zadaniu zajmiemy się kompleksowo analizą czasów działania wszystkich powyższych trzech algorytmów, gdzie algorytmy iteracyjne zbiegają się. Szukane rozwiązania układów liniowych będą przebiegały dla macierzy o rozmiarach N x N dla N kolejno: 100, 500, 1000, 2000, 3000, 4000 dla danych z zadania A. Oto uzyskane wyniki przedstawione na diagramie słupkowym.



Dokładne czasy wykonania:

Rozmiar A	100x100	500x500	1000x1000	2000x2000	3000x3000	4000x4000
Jacobi	0.068816	1.840668	7.981138	31.999883	73.355693	135.077222
Gauss-Seidel	0.033909	0.95212	3.664479	15.050234	36.126139	64.168357
Fraktoryzacja LU	0.030917	3.836675	30.179068	240.165795	821.549072	1922.664468



Jak można zauważyć, czas wykonania dla metody bezpośredniej (faktoryzacji LU) rośnie zdecydowanie szybciej od metod iteracyjnych. Złożoność czasowa algorytmów iteracyjnych to $O(n^2)$, gdy złożoność fraktoryzacji LU to $O(n^3)$. Do rozmiaru macierzy 500 x 500 czasy działania są znikome i dla każdej metody są bliskie maksymalnie kilku sekund. Czas metody Jacobiego rośnie w podobnym tempie (kilka procent szybszym) do metody Gaussa-Seidla, lecz mimo tego jest ona około dwukrotnie wolniejsza od tej drugiej. Mimo tego wypadają one znacznie lepiej w porównaniu do fraktoryzacji LU, lecz, że metoda bezpośrednia daje dokładniejsze wyniki.

7. Obserwacje - Zadanie F

Najważniejszym wnioskiem, który się nasuwa, jest to, że główną zaletą korzystania z metod iteracyjnych przy szukaniu rozwiązania układu równań liniowych jest czas wykonania. Metoda Gaussa-Seidla wypada w tej kategorii zdecydowanie najlepiej, gdyż czas wykonania jest ponad dwukrotnie niższy niż w przypadku metody Jacobiego oraz praktycznie 30-krotnie szybszy niż dla metody bezpośredniej. Ogromną wadą algorytmów iteracyjnych jest jednak to, że nie zawsze błąd będzie zbiegał do zera, tak więc trzeba wiedzieć, kiedy należy używać owych metod. Dla danych z zadania C niemożliwym było znalezienie rozwiązania dzięki metodom iteracyjnym, lecz udało się tego dokonać czy pomocy fraktoryzacji LU. Wynikało to z tego, że macierz A nie była diagonalnie dominująca, tak więc trafnym wnioskiem jest to, że metoda bezpośrednia jest bardziej odporna na zróżnicowaną zawartość macierzy A. Oprócz tego metody bezpośrednie dają bardzo dokładny wynik, gdyż błędy są rzędu 10^{-13}, lecz czy to rekompensuje bardzo długi czas wykonania, pozostaje w gestii ludzi odpowiedzialnych za obliczenia. Warto podkreślić, że wybór odpowiedniej metody zależy od konkretnych wymagań problemu oraz tolerancji na błędy obliczeń. W niektórych przypadkach, gdzie precyzja jest kluczowa, metody bezpośrednie mogą być preferowane pomimo dłuższego czasu wykonania. Natomiast dla zadań, gdzie liczy się szybkość, metody iteracyjne mogą być bardziej odpowiednie.