Sprawozdanie z Projektu 2

Jakub Bronowski, 193208, gr. 2

Wstęp

Niniejszy dokument opisuje wyniki analiz metod rozwiązywania macierzowych układów równań linowych metodami: Jacobiego, Gaussa-Seidla oraz LU (2 iteracyjne i 1 bezpośrednia). Zadania projektowe zaimplementowano w C++ z użyciem obiektów i metod. Wykresy stworzono w Pythonie (*matplotlib)* na podstawie wyznaczonych danych. Użyto biblioteki *cmath*, która dostarcza funkcje *sin()* oraz *sqrt()*. Nie wykorzystano domyślnych implementacji klasy *Matrix* wraz z algebrą macierzy oraz metod rozwiązujących zadanie problemy. Czas mierzono przy użyciu *high_resolution_clock::now()* z biblioteki *chrono.*

Teoria

Zadanie projektowe polegało na rozwiązaniu macierzowych układów równań liniowych postaci *Ax = b* gdzie *A* – macierz współczynników, *x* – wektor rozwiązań, *b* – wektor wyrazów wolnych.

Metoda Jacobiego

Metoda Jacobiego polega na iteracyjnym poprawianiu wektora rozwiązań *x.* Początkowo wektor *x* zawiera same 0. Iteracyjnie wyliczamy nowe wartości dla wektora *x* do momentu, aż osiągniemy warunek zbieżności zadany przez normę błędu mniejszą niż 10^{-9}. Warto nadmienić, że metoda nie zawsze jest zbieżna, więc nie zawsze otrzymamy rozwiązanie zadanego problemu.

$$x'_i = \frac{1}{a_{i,i}}\left(b_i - \sum_{j \neq i} a_{i,j} * x_j\right) dla\ i = 1,2,3 \dots, n$$

$$x'_i - wartość\ i - tego\ elementu\ wektora\ z\ następnej\ iteracji$$

Wzór 1: Wyliczanie elementów wektora *x* korzystając z wartości z poprzedniej iteracji.

Metoda Gaussa-Seidela

Metoda Gaussa-Seidela jest bardzo podobna do wyżej opisanej metody Jacobiego z tą tylko różnicą, że do wyliczenia *k-tego* elementu wektora *x* korzystamy z już wyliczonych elementów z tej samej iteracji (a nie z iteracji poprzedniej jak w Jacobim), o ile takowe już istnieją. Jeśli nie, korzystamy z sumy z poprzedniej iteracji. Metoda nie gwarantuje zbieżności i rozwiązania problemu.

$$x'_i = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j} * x'_j - \sum_{j=i+1}^{n} a_{i,j} * x_j\right) dla\ i = 1,2,3 \ldots, n$$

$$x'_i - wartość\ i - tego\ elementu\ wektora\ z\ następnej\ iteracji$$

Wzór 2: Wyliczanie elementów wektora *x* korzystając z wartości z tej samej i poprzedniej iteracji.

Faktoryzacja LU

Jest to jedyna badana metoda bezpośrednia polegająca na podzieleniu macierzy *A* na 2: *L* oraz *U*, takie że *L* – macierz trójkątna górna, *U* – macierz trójkątna dolna oraz spełniony jest warunek *A = L * U*. Wtedy równanie *Ax = b* przyjmuje postać *LUx = b.* Co następnie można rozwiązać przez podstawianie w przód dla równania pomocniczego *Ly = b* oraz podstawiania w tył dla rozwiązania równania *Ux = y*. Wektory *x* oraz *y* obliczono używając poniższych wzorów.

$$y_i = b_i - \sum_{j=1}^{i-1} L_{i,j} * y_j \quad dla\ i = 1,2,3,\ldots,n$$

$$x_i = \frac{y_i - \sum_{j-i+1}^{n} U_{i,j} * x_i}{U_{i,i}}\ dla\ i = n, n-1, n-2, \ldots 1$$

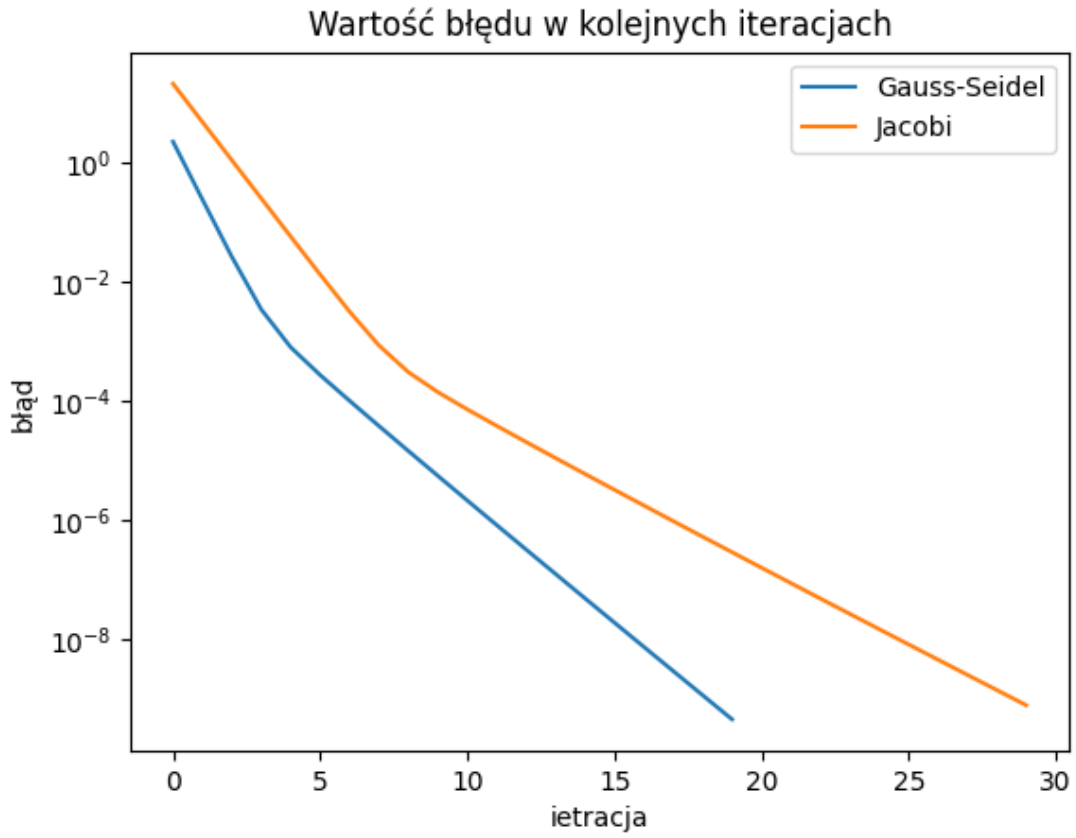
Wzór 3: Wzory wyliczające poszczególne elementy wektorów *x* oraz *y*.

Zadanie A i B

Stworzono macierz *A* o rozmiarach 908 x 908, na głównej przekątnej wartość 7, a na dwóch poniżej i powyżej -1. Wektor *b* o rozmiarach 908 x 1 zawiera wartość wyrażenia *sin(4i)* dla *i*-tego elementu wektora. Oczekiwany błąd rzędu 10^{-9}. Rezultaty pracy metod Jacobiego oraz Gaussa-Seidela opisuje tabela:

	Jacobi	Gauss-Seidel
Ilość iteracji	30	20
Czas [s]	3,921	1,390
Błąd [* 10^{-10}]	7,99685	4,63498

Metoda Jacobiego cechuje się nieco mniejszym błędem niż metoda Gaussa-Seidela, choć obie są o rząd wielkości mniejsze niż oczekiwano.



Wykres 1: Zbieżność metod iteracyjnych dla równania macierzowego z zadania A i B. Oś OY w skali logarytmicznej.

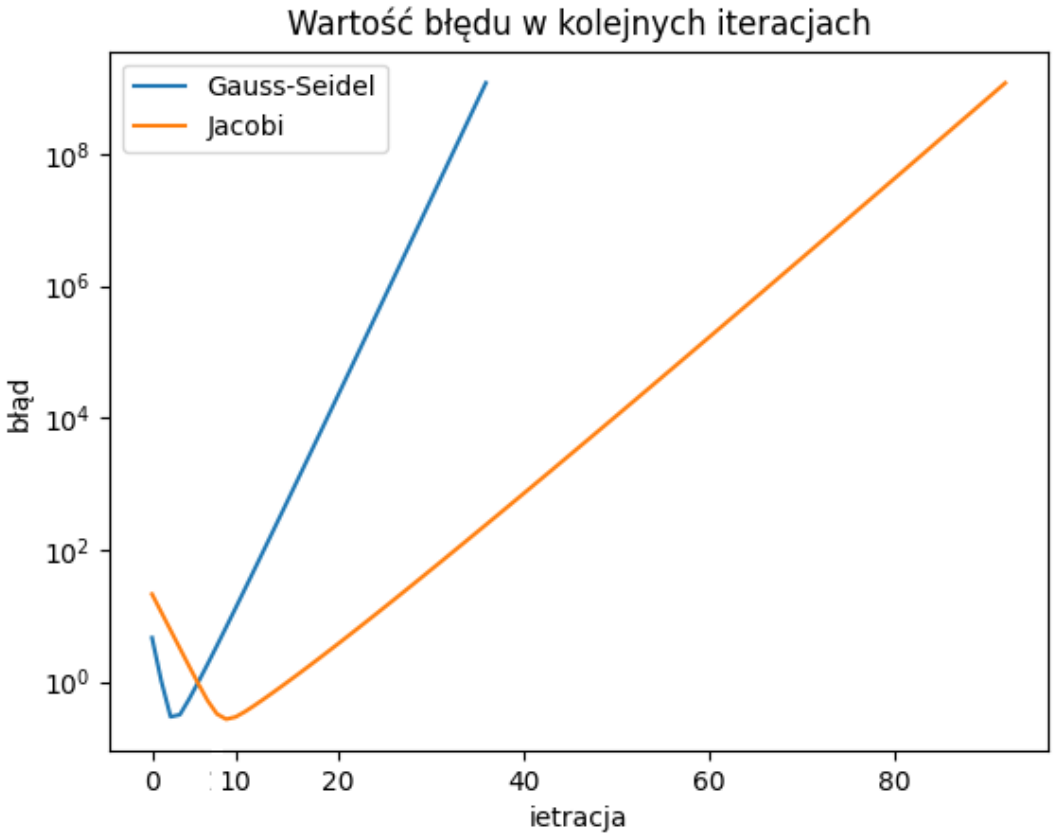
Obie metody są zbieżne, błąd maleje szybciej w przypadku metody Gaussa-Seidela.

Zadanie C

Stworzono macierz *A* o rozmiarach 908 x 908, na głównej przekątnej wartość 3, a na dwóch poniżej i powyżej -1. Wektor *b* o rozmiarach 908 x 1 zawiera wartość wyrażenia *sin(4i)* dla *i*-tego elementu wektora. Oczekiwany błąd rzędu 10^{-9}. Przerwanie pracy przy osiągnięciu błędu na poziomie 10^{9}. Rezultaty pracy metod Jacobiego oraz Gaussa-Seidela opisuje tabela:

	Jacobi	Gauss-Seidel
Ilość iteracji	93	37
Czas [s]	7,155	2,491
Błąd [* 10^{9}]	1,17727	1,18183

Każda z metod szybko przekroczyła górną granicę błędu. Metoda Jacobiego przekroczyła ją mocniej.



Wykres 2: Zbieżność metod iteracyjnych dla równania macierzowego z zadania A i B. Oś OY w skali logarytmicznej.

Żadna z metod nie zbiegła się do dopuszczalnej wartości błędu, co równoznaczne z brakiem znalezienia rozwiązania. Najmniejszy błąd wystąpił w 3 iteracji dla metody Gaussa-Seidela i 9 iteracji dla metody Jacobiego.

Zadanie D

Stworzono macierz *A* o rozmiarach 908 x 908, na głównej przekątnej wartość 3, a na dwóch poniżej i powyżej -1. Wektor *b* o rozmiarach 908 x 1 zawiera wartość wyrażenia *sin(4i)* dla *i*-tego elementu wektora. Oczekiwany błąd rzędu 10^{-9}, choć spodziewamy się lepszego wyniku. Nie występuje przerwanie pracy po przekroczeniu poziomy błędu. Zaimplementowano również metodę bezpośrednią rozwiązywania równań faktoryzacją LU. Wyniki pracy przedstawiono poniżej.

	Faktoryzacja LU
Ilość iteracji	1 (metoda bezpośrednia)
Czas [s]	11,125
Błąd [* 10^{-13}]	3,6075

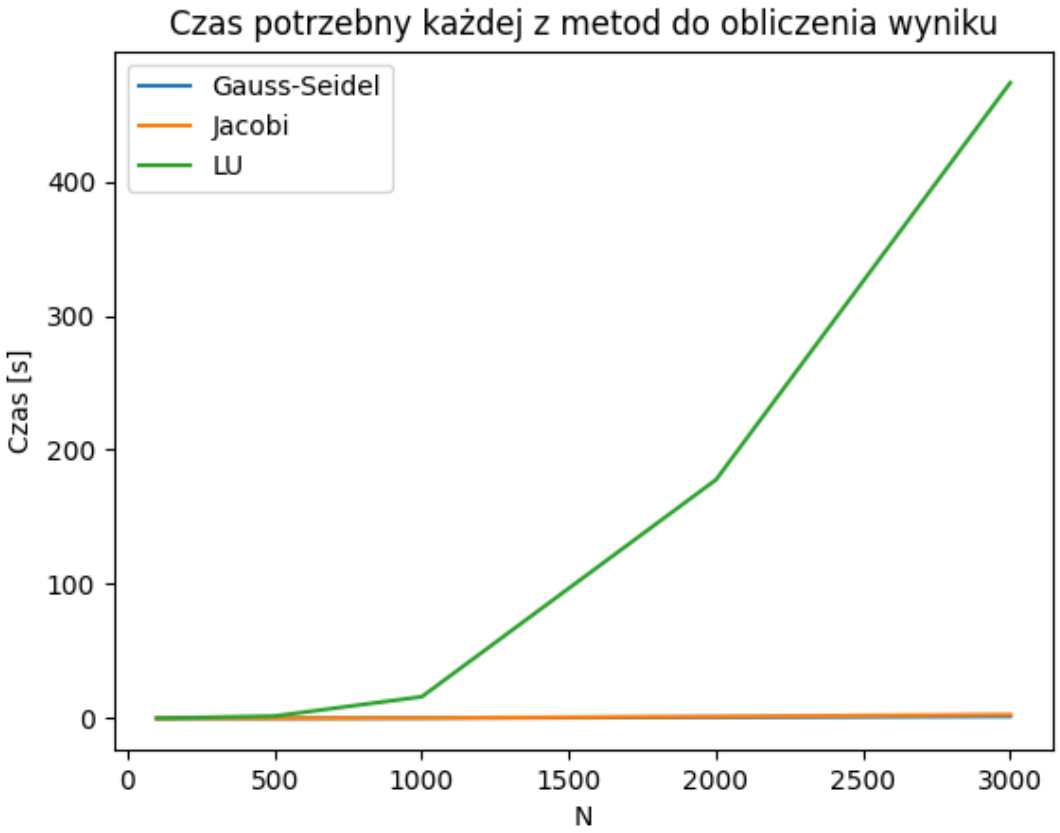
Jak widać, dla tego samego równania *Ax = b* co w zadaniu C otrzymaliśmy rozwiązanie problemu, czego nie potrafiły osiągnąć metody iteracyjne. Więcej niż zadowalająca jest też dokładność rozwiązania na poziomie ok. 35 bilionowych, blisko 4 rzędy wielkości mniej niż oczekiwany poziom błędu.

Zadanie E

Ponownie stworzono macierz *A* o rozmiarach 908 x 908, na głównej przekątnej wartość 7, a na dwóch poniżej i powyżej -1. Wektor *b* o rozmiarach 908 x 1 zawiera wartość wyrażenia *sin(4i)* dla *i*-tego elementu wektora. Oczekiwany błąd rzędu 10^{-9}, przy którym metody iteracyjne zatrzymują swoją pracę. Wykonano testy wydajnościowe dla wyżej opisanych macierzy o różnych rozmiarach N. Zebrane dane zaprezentowano w tabeli, a czas potrzebny na wyliczenie poszczególnych danych przedstawiono na wykresie.

Testy uruchomiono na procesorze Intel i5-5200.

N	Jacobi [s]	Gauss-Seidel [s]	Faktoryzacja LU [s]
100	0,003	0,010	0,014
500	0,062	0,049	1,696
1000	0,323	0,166	16,082
2000	1,317	0,829	177,765
3000	2,825	1,485	473,523



Wykres 3: czasy wykonywania testów dla poszczególnych rozmiarów macierzy (Intel i5).

Tak długie czasy dla faktoryzacji LU są spowodowane złożonością metody (jest ona bardzo obliczeniochłonna). Nie pomaga też fakt implementacyjny, dane w macierzy są składowane w strukturze *std::vector< std::vector<double> >* co może nie sprzyjać cache pamięci a zatem zwiększać czas dostępu do danych.

Wnioski końcowe

Metoda LU jest najwolniejszą z badanych, za to potrafi rozwiązać każdy z przypadków testowych. Metody iteracyjne są szybsze, ale nie zawsze gwarantują znalezienie poprawnego rozwiązania, residuum błędu może się nie zbiegać do wartości tolerancji. Nasuwa się na myśl strategia, aby macierzowe układy równań liniowych rozwiązywać najpierw metodą Gaussa-Seidlera, a gdy ta zawiedzie wykorzystać metodę LU ze względu na złożoność obliczeniową rzędu $O(n^3)$.