POLITECHNIKA GDAŃSKA

Układy równań liniowych

Metody Numeryczne - Projekt 2

Ruslan Rabadanov 196634

22-04-2024

Spis treści

1. Wstęp

Projekt ma na celu zaimplementowanie oraz porównanie różnych iteracyjnych metod rozwiązywania układów równań liniowych oraz metody faktoryzacji LU.

W ramach projektu zostały zaimplementowane metody Jacobiego i Gaussa-Seidela, a także metoda faktoryzacji LU bez korzystania z zewnętrznych bibliotek do obliczeń macierzowych.

Następnie przeprowadzono testy wydajnościowe, które umożliwiły porównanie czasu potrzebnego do rozwiązania układów równań liniowych w zależności od zastosowanej metody i rozmiaru danych.

Do realizacji tego zadania użyto języka programowania Python oraz biblioteki Matplotlib do wizualizacji wyników. Wszystkie operacje na macierzach wykonano przy użyciu własnoręcznie napisanej klasy `Matrix`.

Zgodnie z treścią Zadania A na początku projektu utworzono układ równań liniowych reprezentowany przez macierze A i B.

Macierz A jest kwadratową macierzą pasmową o wymiarach $N \times N$ dla $N = 9cd = 934$, gdzie $a_1 = 5 + 6 = 11$, $a_2 = a_3 = -1$. a_1 to główna przekątna, a_2 to przekątne przesunięte o 1 względem głównej przekątnej itd. Dodatkowo, mamy wektor B o długości N, którego n-ty element jest równy $\sin(n \cdot (f + 1)) = \sin(7n)$

$$A = \begin{pmatrix} 11 & -1 & -1 & 0 & \dots & 0 & 0 & 0 & 0 \\ -1 & 11 & -1 & -1 & \dots & 0 & 0 & 0 & 0 \\ -1 & -1 & 11 & -1 & \dots & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 11 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 11 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & \dots & -1 & 11 & -1 & -1 \\ 0 & 0 & 0 & 0 & \dots & -1 & -1 & 11 & -1 \\ 0 & 0 & 0 & 0 & \dots & 0 & -1 & -1 & 11 \end{pmatrix} \quad B = \begin{pmatrix} 0.657 \\ 0.9906 \\ 0.8367 \\ 0.2709 \\ \vdots \\ 0.9728 \\ 0.8857 \\ 0.3627 \\ -0.3388 \end{pmatrix}$$

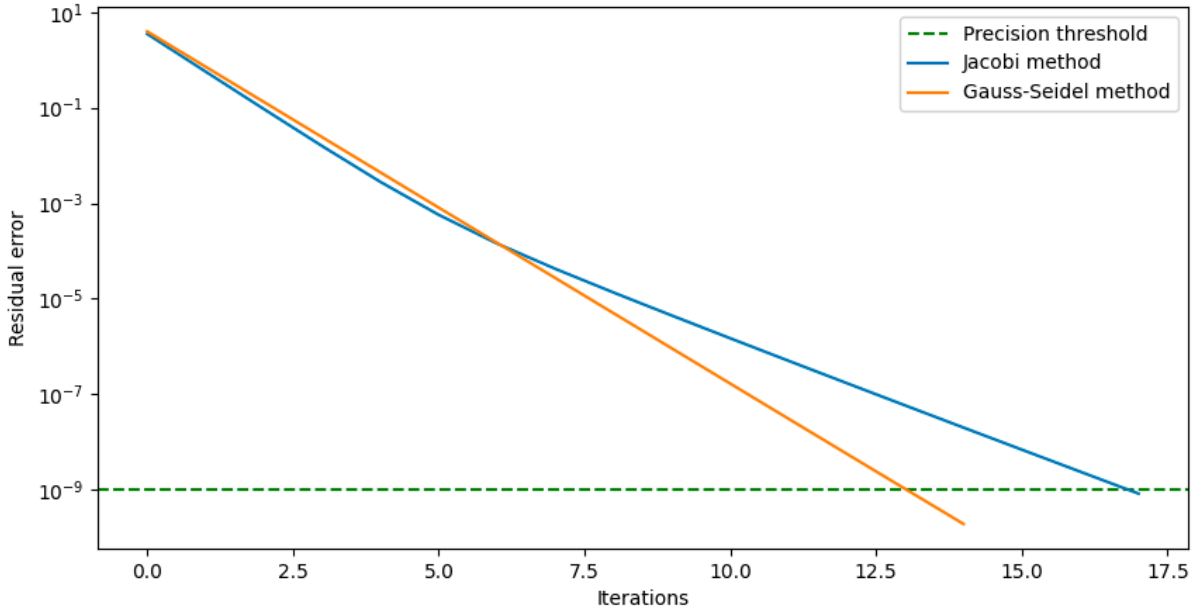
2. Porównanie metod rozwiązywania układów równań liniowych dla domyślnego układu równań

W ramach Zadania B zaimplementowano metody Jacobiego i Gaussa-Seidela, które zostały wykorzystane do rozwiązania układu równań liniowych opisanego we wstępie tego sprawozdania.

Metoda	Liczba iteracji	Czas obliczeń [s]	Norna residuum
Jacobi	17	6.463	$8.243249338856146 \times 10^{-10}$
Gauss-Seidel	14	5.497	$1.905168171052384 \times 10^{-10}$
Faktoryzacja LU	-	59.82	$3.4073565309548815 \times 10^{-15}$

Tabela 1: Porównanie metod Jacobiego, Gaussa-Seidela i faktoryzacji LU dla domyślnego układu równań

Dla przedstawionego układu równań liniowych metoda Jacobiego wymagała 17 iteracji, podczas gdy metoda Gaussa-Seidela tylko 14 iteracji. Wyniki te wskazują, że metoda Gaussa-Seidela jest szybsza od metody Jacobiego (5.497 s vs 6.463 s). Z wykresu widać, że po szóstej iteracji spadek błędu obliczeniowego dla metody Jacobiego spowalnia w porównaniu z metodą Gaussa-Seidela.

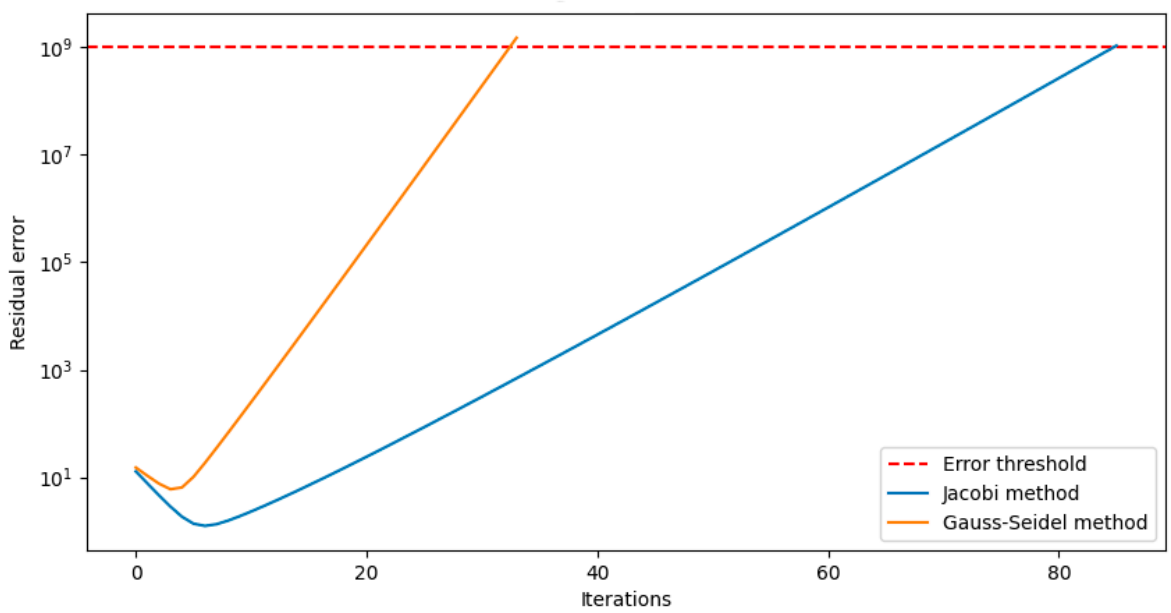


Rysunek 1: Porównanie normy residuum w zależności od iteracji dla metod Jacobiego i Gaussa-Seidela

3. Porównanie metod rozwiązywania układów równań liniowych dla alternatywnego układu równań

W ramach Zadania C i Zadania D wygenerowano alternatywny układ równań liniowych, który różni się od domyślnego układu równań liniowych opisanego we wstępie jedynie wartościami elementów macierzy A. Jedyną zmianą było ustawienie wartości głównej przekątnej na $a_1 = 3$.

$$A = \begin{pmatrix} 3 & -1 & -1 & 0 & \dots & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & \dots & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & \dots & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 3 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 3 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & \dots & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & \dots & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & \dots & 0 & -1 & -1 & 3 \end{pmatrix} \quad B = \begin{pmatrix} 0.657 \\ 0.9906 \\ 0.8367 \\ 0.2709 \\ \vdots \\ 0.9728 \\ 0.8857 \\ 0.3627 \\ -0.3388 \end{pmatrix}$$



Rysunek 2: Porównanie normy residuum w zależności od iteracji dla metod Jacobiego i Gaussa-Seidela dla alternatywnego układu równań

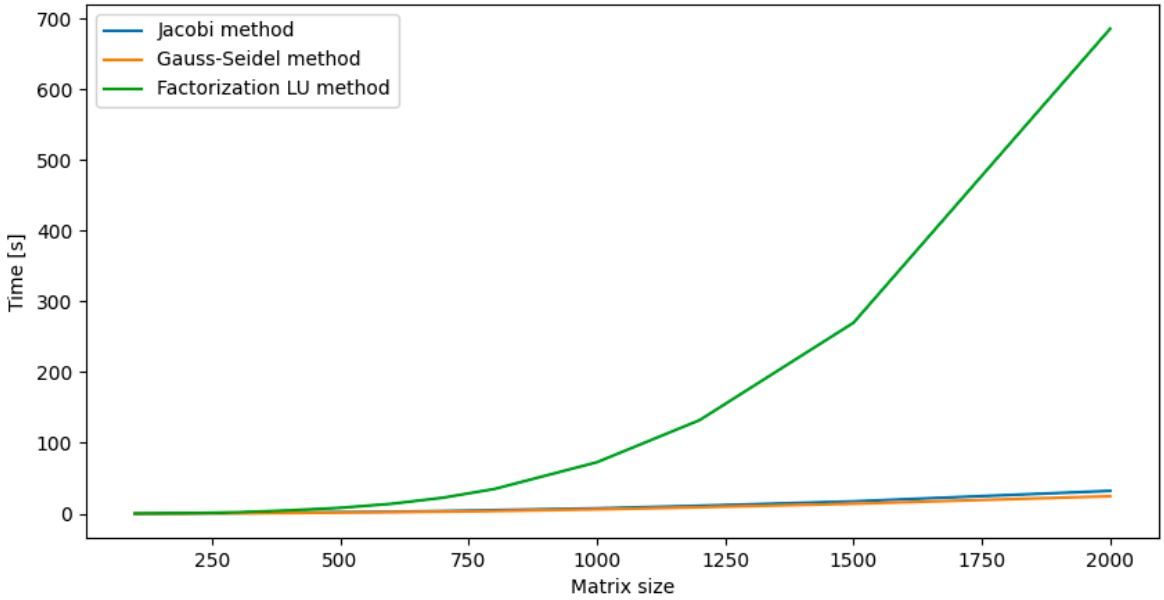
Niestety, dla alternatywnego układu równań liniowych ani metoda Jacobiego, ani metoda Gaussa-Seidela nie były w stanie znaleźć poprawnego rozwiązania. Błąd obliczeń dla obu metod wzrastał już od pierwszych iteracji, osiągając normę residuum na poziomie 10^0 do 10^1, co wskazuje na to, że metody te nie są zbieżne dla tego układu równań. Szczególnie w przypadku metody Gaussa-Seidela błąd wzrastał znacznie szybciej, co można zaobserwować na wykresie normy residuum w zależności od iteracji.

W Zadaniu D, dla metody faktoryzacji LU, błąd obliczeń wyniósł $1.954704687433127 \times 10^{-12}$, co wskazuje na bardzo wysoką dokładność tej metody dla tego układu równań, chociaż kosztem czasu obliczeń.

4. Czas obliczeń w zależności od rozmiaru macierzy

W ramach Zadania E przeprowadzono testy wydajnościowe dla metod Jacobiego, Gaussa-Seidela i faktoryzacji LU dla układów równań liniowych o różnych rozmiarach.

Testy wydajnościowe przeprowadzono dla macierzy kwadratowych o rozmiarach $N = \{100, 200, \dots, 700, 800, 1000, 1200, 1500\}$. Dla każdego rozmiaru macierzy wygenerowano układ równań liniowych zgodnie z opisem we wstępie tego sprawozdania.



Rysunek 3: Porównanie czasu obliczeń w zależności od rozmiaru macierzy dla metod Jacobiego, Gaussa-Seidela i faktoryzacji LU

Na wykresie czasu obliczeń w zależności od rozmiaru macierzy dla metod Jacobiego, Gaussa-Seidela i faktoryzacji LU można zauważyć, że metoda faktoryzacji LU jest zdecydowanie wolniejsza od metod iteracyjnych, co staje się szczególnie zauważalne dla dużych macierzy. Metoda Gaussa-Seidela jest trochę szybsza od metody Jacobiego dla wszystkich rozmiarów macierzy.

5. Podsumowanie

W ramach projektu zaimplementowano metody Jacobiego i Gaussa-Seidela oraz metodę faktoryzacji LU. Przeprowadzone testy wydajnościowe pozwoliły na porównanie czasu obliczeń rozwiązań układów równań liniowych w zależności od zastosowanej metody oraz rozmiaru macierzy.

Podczas testów wydajnościowych zauważono, że metoda faktoryzacji LU jest znacznie wolniejsza od metod iteracyjnych (nawet 20 razy wolniejsza w przypadku macierzy 2000×2000), co jest szczególnie istotne dla dużych macierzy. Metoda Gaussa-Seidela okazała się szybsza od metody Jacobiego dla wszystkich rozmiarów macierzy. Pomimo to, metoda faktoryzacji LU była najdokładniejsza, co można zaobserwować na przykładzie alternatywnego układu równań liniowych, który nie zbiegał dla metod iteracyjnych.

Wnioskiem ogólnym jest to, że przy rozwiązywaniu różnych klas układów równań liniowych zaleca się rozpoczęcie od metody Gaussa-Seidela, a w przypadku braku zbieżności (co można stwierdzić, zauważając, że błąd zamiast maleć przy kolejnych iteracjach wzrasta) można przejść do metody faktoryzacji LU. Takie podejście pozwala uzyskać dokładne wyniki w krótszym czasie, pod warunkiem, że układ równań jest zbieżny, oraz umożliwia uzyskanie wyników dla układów równań, które nie zbiegają dla metod iteracyjnych.

Dodatkowo, należy zauważyć, że zastosowanie zewnętrznych bibliotek do obliczeń macierzowych (np. NumPy) lub wykorzystanie szybszego języka do implementacji może znacząco przyspieszyć obliczenia, szczególnie istotne dla dużych macierzy. Testowana implementacja została napisana w czystym języku Python, który nie należy do szybkich języków, co znacząco wpłynęło na czas obliczeń, zwłaszcza dla metody faktoryzacji LU, która zajęła nawet 11 minut dla macierzy o rozmiarze 2000×2000.