POLITECHNIKA GDAŃSKA

Układy równań liniowych

Metody Numeryczne - Projekt 2

Ruslan Rabadanov 196634

23-04-2024

Spis treści

1. Wstęp

Projekt ma na celu zaimplementowanie oraz porównanie dwóch iteracyjnych metod rozwiązywania układów równań liniowych (Jacobiego i Gaussa-Seidela) oraz metody faktoryzacji LU.

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 macierzą systemową o wymiarach $N \times N$ dla $N = 9cd = 934$, gdzie główna przekątna (a_1) przyjmuje wartości 11, a dwie przesunięte przekątnę (a_2 i a_3) przyjmują wartości -1. Dodatkowo, mamy wektor pobudzenia B o długości N, którego n-ty element jest równy $\sin(7n)$.

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

Pierwszy uklad równań liniowych

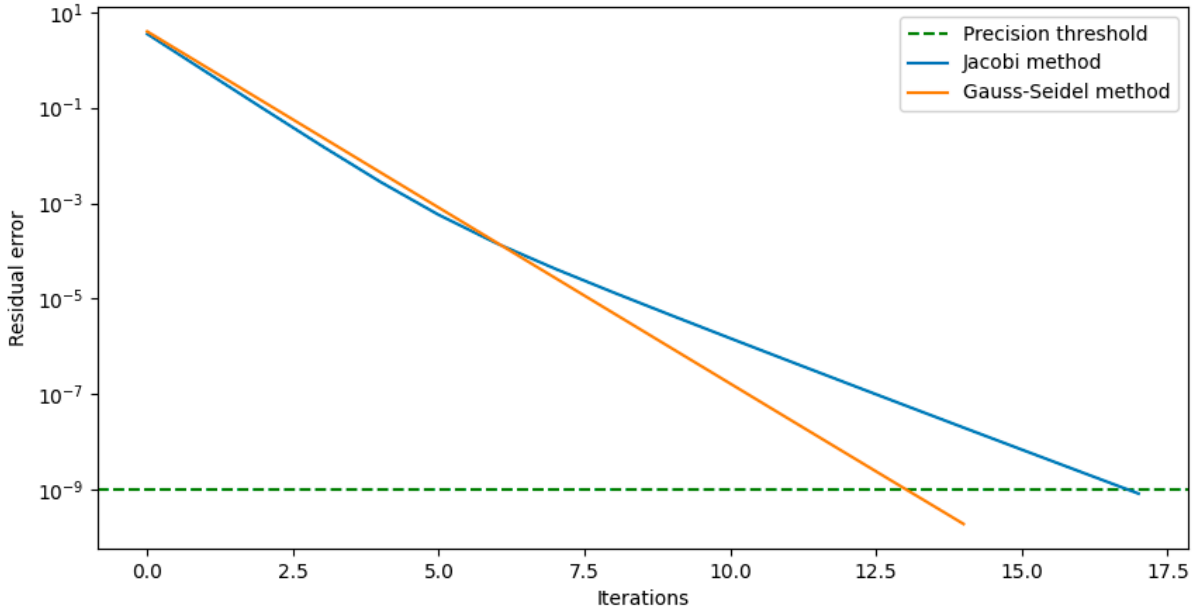
2. Rozwiązywanie domyślnego układu równań liniowych

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 pierwszego 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.



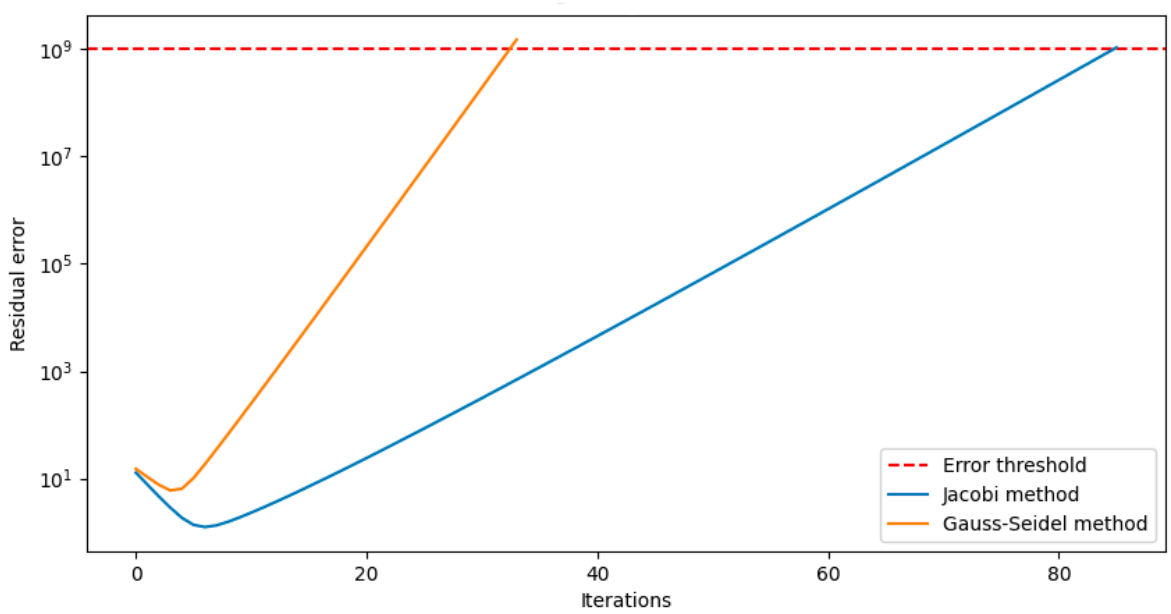
Rysunek 1: Porównanie normy residuum w zależności od iteracji dla metod Jacobiego i Gaussa-Seidela

3. Rozwiązywanie alternatywnego układu równań liniowych

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 systemowej A. Jedyną zmianą było ustawienie wartości głównej przekątnej na $a_1 = 3$. Wektor pobudzenia B pozostaje bez zmian.

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

Drugi uklad równań liniowych



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

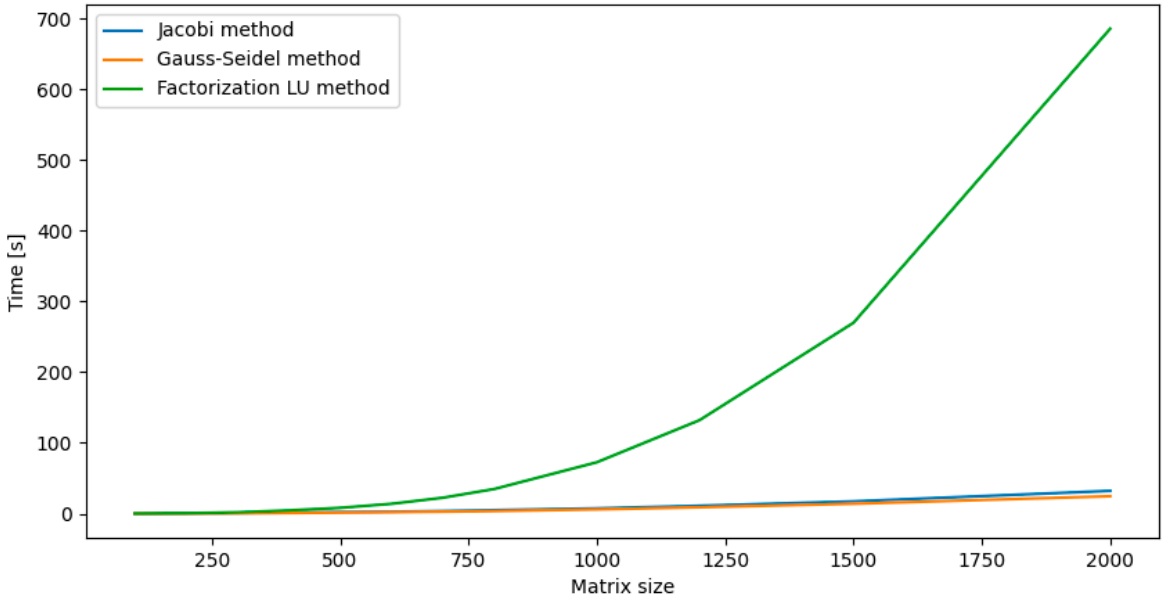
Jak widać z wykresu, dla alternatywnego układu równań liniowych ani metoda Jacobiego, ani metoda Gaussa-Seidela nie były w stanie znaleźć poprawnego rozwiązania. Błąd obliczeń dla obu metod wzrastał już od pierwszych iteracji, osiągając minimalną normę residuum na poziomie 10^0 do 10^1, co wskazuje na to, że metody te nie są zbieżne dla tego układu równań. Szczególnie w przypadku metody Gaussa-Seidela błąd znowu osiągnął wartość graniczną szybciej, niż w metodzie Jacobiego.

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 (58.93s).

4. Zależność czasu obliczeń 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

Z wykresu widać, że metoda faktoryzacji LU jest znacznie wolniejsza od metod iteracyjnych, co staje się szczególnie zauważalne dla dużych macierzy. Metoda Gaussa-Seidela jest trochę szybsza od metody Jacobiego dla wszystkich badanych 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 wynika ze złożoności obliczeniowej ($O(n^3)$ vs $O(n^2)$). Metoda Gaussa-Seidela okazała się o kilkanaście procent szybsza od metody Jacobiego. Pomimo to, metoda faktoryzacji LU była najdokładniejsza, co można zaobserwować na przykładzie alternatywnego układu równań liniowych, który się nie zbiegał dla metod iteracyjnych.

W celu przyspieszenia obliczeń można zastosować hybrydowe metody rozwiązywania układów równań przy wykorzystaniu gotowych narzędzi i szybszego języka programowania. Na przykład, można rozpocząć 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 dla układów równań, które nie zbiegają dla metod iteracyjnych. Dodatkowo, należy korzystać z zewnętrznych bibliotek do obliczeń macierzowych (np. NumPy) lub wykorzystać inny, szybszy od Pythona, język programowania.