Układy równań liniowych

Metody Numeryczne - Projekt 2

Stanisław Nieradko 193044

2024-04-02

Spis treści

1. Wstęp

Celem projektu jest zaimplementowanie i porównanie iteracyjnych metod rozwiązywania układów równań liniowych oraz metody faktoryzacji LU.

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

Następnie zostaną przeprowadzone testy wydajnościowe, które pozwolą na porównanie czasu wyznaczenia rozwiązań układów równań liniowych w zależności od zastosowanej metody.

Do zrealizowania zadania wykorzystany zostanie język programowania Python, środowisko Jupyter Notebook oraz biblioteka Matplotlib do wizualizacji wyników. Wszelkie operacje na macierzach będą wykonywane przy użyciu własnoręcznie napisanej klasy `Matrix`.

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

Macierz A jest macierz pasmową kwadratową o wymiarach $N \times N$ dla $N = 9cd = 944$, $a_1 = 5 + 3 = 5$, $a_2 = a_3 = -1$. Wartości a_i indukują wartości przekątnych o przesunięciu względem głównej przekątnej macierzy offset $= i - 1$ (także a_1 to główna przekątna, a_2 to przekątne przesunięte o 1 względem głównej przekątnej itd.). Mamy też wektor B o długości N, którego n-ty element jest równy $\sin(n \cdot (f + 1)) = \sin(n)$

$$A = \begin{pmatrix} 5 & -1 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ -1 & 5 & -1 & -1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 5 & -1 & -1 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 5 & -1 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 5 & \dots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 5 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 & 5 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 & -1 & 5 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & -1 & -1 & 5 & -1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & -1 & -1 & 5 \end{pmatrix} \quad B = \begin{pmatrix} -0.7568 \\ 0.9894 \\ -0.5366 \\ -0.2879 \\ 0.9129 \\ \vdots \\ 0.4675 \\ 0.3635 \\ -0.9426 \\ 0.8688 \\ -0.1931 \end{pmatrix}$$

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

W ramach Zadania B zaimplementowane zostały metody Jacobiego i Gaussa-Seidela, które posłużyły do rozwiązania układu równań liniowych zdefiniowanego we wstępie.

Metoda	Liczba iteracji	Czas obliczeń [s]	Błąd
Jacobi	67	11.7006	8.830506921606741e-10
Gauss-Seidel	39	6.4401	8.212466420029578e-10
Faktoryzacja LU	-	35.8186	1.9124294068938155e-15

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

Dla podanego układu równań liniowych metoda Jacobiego wymagała 67 iteracji, a Gaussa-Seidela 39 iteracji, co wskazuje na to, że metoda Gaussa-Seidela jest szybsza od metody Jacobiego (6.44 s vs 11.70 s). Błąd obliczeń jest na bardzo podobnym poziomie dla obu metod z delikatnym zwycięstwem metody Gaussa-Seidela.

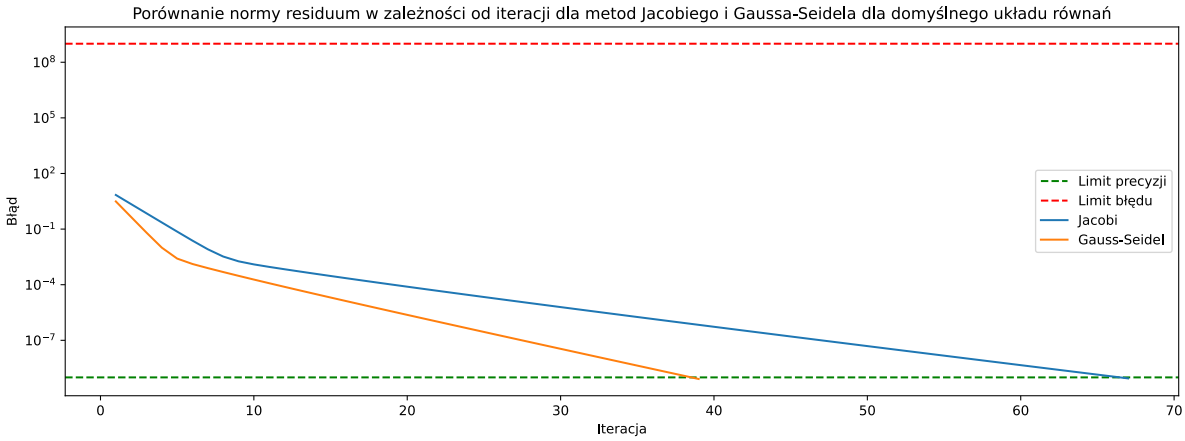

Figure 1: Porównanie normy residuum w zależności od iteracji dla metod Jacobiego i Gaussa-Seidela

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

W ramach Zadania C i Zadanie D wygenerowano alternatywny układ równań liniowych, który różni się od domyślnego układu równań liniowych z wstępu jedynie wartościami elementów macierzy A. Jedyną zmianą były wartości głównej diagonalnej na $a_1 = 3$.

$$A = \begin{pmatrix} 3 & -1 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & -1 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 3 & -1 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 3 & \dots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 3 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 & 3 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & -1 & -1 & 3 \end{pmatrix} \quad B = \begin{pmatrix} -0.7568 \\ 0.9894 \\ -0.5366 \\ -0.2879 \\ 0.9129 \\ \vdots \\ 0.4675 \\ 0.3635 \\ -0.9426 \\ 0.8688 \\ -0.1931 \end{pmatrix}$$

Dla takich danych poszczególne metody rozwiązywania układów równań liniowych osiągnęły następujące wyniki:

Metoda	Liczba iteracji	Czas obliczeń [s]	Błąd	Najmniejszy błąd (iteracja)
Jacobi	93	16.5827	894576398.9636	0.18136286954970482 (9)
Gauss-Seidel	36	6.3301	596229785.312	0.28223629372829717 (3)
Faktoryzacja LU	-	30.2689	6.13589416330441e-13	-

Table 2: Porównanie metod Jacobiego, Gaussa-Seidela i faktoryzacji LU dla alternatywnego układu równań

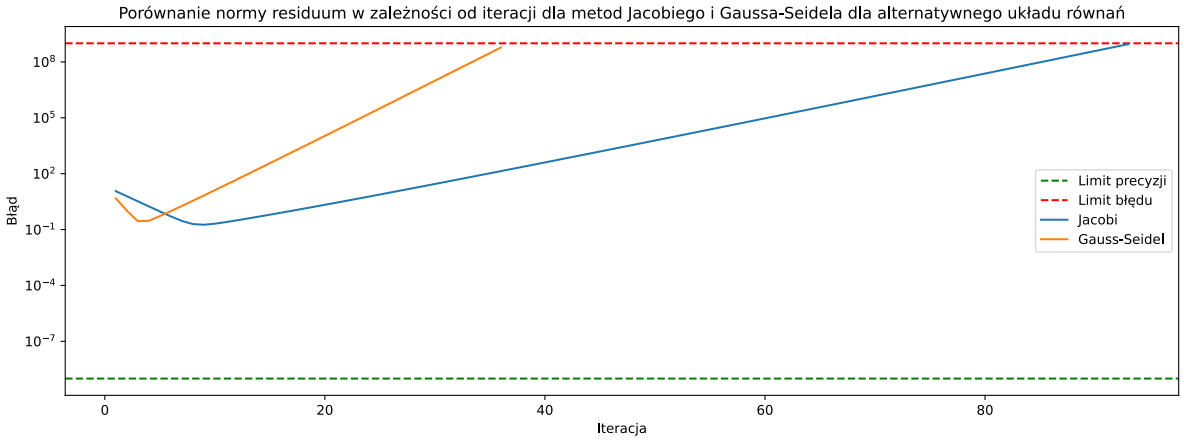

Figure 2: Porównanie normy residuum w zależności od iteracji dla metod Jacobiego i Gaussa-Seidela dla alternatywnego układu równań

Niestety dla alternatywnego układu równań liniowych zarówno metoda Jacobiego jak i metoda Gaussa-Seidela nie były w stanie znaleźć poprawnego rozwiązania. Błąd obliczeń dla obu metod rośnie od odpowiednio 9 i 3 iteracji (osiągając błąd na poziomie 10^{-1}), co wskazuje na to, że metody te nie są zbieżne dla tego układu równań. W przypadku metody Gaussa-Seidela błąd wzrasta zdecydowanie szybciej, co widać na wykresie normy residuum w zależności od iteracji.

W przypadku metody faktoryzacji LU błąd obliczeń jest na poziomie $6.13589416330441e-13$, co wskazuje na to, że metoda ta jest bardzo dokładna dla tego układu równań 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 $N = \{100, 500, 1000, 2000, 3000\}$. Dla każdego rozmiaru macierzy wygenerowano układ równań liniowych zgodnie z wstępem.

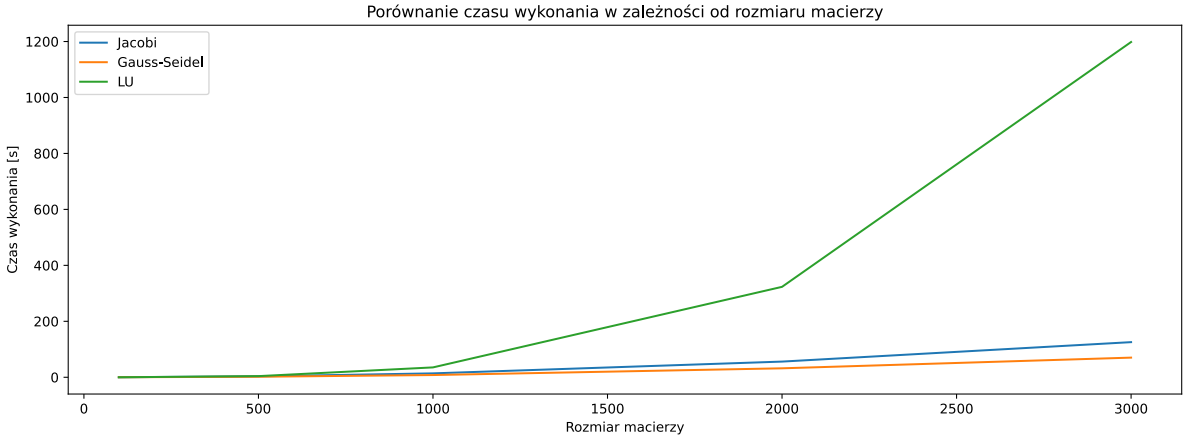

Figure 3: Porównanie czasu obliczeń w zależności od rozmiaru macierzy dla metod Jacobiego, Gaussa-Seidela i faktoryzacji LU

Jak widać na wykresie czasu obliczeń w zależności od rozmiaru macierzy dla metod Jacobiego, Gaussa-Seidela i faktoryzacji LU, metoda faktoryzacji LU jest zdecydowanie wolniejsza od metod iteracyjnych co jest mocno

odczuwalne dla dużych macierzy. Metoda Gaussa-Seidela jest szybsza od metody Jacobiego dla wszystkich rozmiarów macierzy.

5. Podsumowanie

W ramach projektu zaimplementowane zostały metody Jacobiego i Gaussa-Seidela oraz metoda faktoryzacji LU. Przeprowadzone testy wydajnościowe pozwoliły na porównanie czasu wyznaczenia rozwiązań układów równań liniowych w zależności od zastosowanej metody oraz rozmiaru macierzy. Podczas testów wydajnościowych zauważono, że metoda faktoryzacji LU jest zdecydowanie wolniejsza od metod iteracyjnych (nawet 9 razy wolniejszy w przypadku macierzy 3000×3000), co jest mocno odczuwalne dla dużych macierzy. Metoda Gaussa-Seidela jest szybsza od metody Jacobiego dla wszystkich rozmiarów macierzy. Mimo tego metoda faktoryzacji LU jest najdokładniejsza, co widać na przykładzie alternatywnego układu równań liniowych, który nie był zbieżny dla metod iteracyjnych.

W ogólności zdaje się, że przy rozwiązywaniu dowolnej klasy układów równań liniowych należałoby na początek skorzystać z metody Gaussa-Seidela, a w przypadku braku zbieżności (co można wykryć poprzez zauważenie, że błąd zamiast maleć przy kolejnych iteracjach wzrasta) zastosować metodę faktoryzacji LU. Powinno to pozwolić na uzyskanie dokładnych wyników w krótkim czasie tak długo jak układ równań jest zbieżny, oraz dać możliwość uzyskania wyników dla układów równań, które nie są zbieżne dla metod iteracyjnych.

Ponadto warto zauważyć, że zastosowanie zewnętrznych bibliotek do obliczeń macierzowych (np. NumPy) lub zastosowanie szybszego języka do implementacji powinno pozwolić na znaczne przyspieszenie obliczeń, co jest szczególnie ważne dla dużych macierzy. Testowana implementacja była napisana w czystym języku Python, który nie jest językiem szybkim, co **znacząco** wpłynęło na czas obliczeń (zwłaszcza metody faktoryzacji LU, która zajęła nawet 20 min dla macierzy 3000×3000).