Metody numeryczne - Projekt nr 2

Hubert Szymczak

WSTĘP

Niniejszy projekt skupia się na implementacji oraz analizie dwóch metod iteracyjnych – metody Jacobiego i metody Gaussa-Seidla – oraz jednej metody bezpośredniej, a mianowicie faktoryzacji LU, do rozwiązywania układów równań liniowych. Projekt został zrealizowany przy użyciu języka Python, który dzięki swojej uniwersalności oraz rozbudowanym bibliotekom numerycznym, stanowi doskonałe narzędzie do obliczeń naukowych. Analiza wyników oraz prezentacja graficzna danych zostały wykonane z wykorzystaniem biblioteki Matplotlib, co umożliwia efektywne tworzenie wykresów kluczowych dla oceny zbieżności i skuteczności zaimplementowanych metod. W celu dokładnego pomiaru czasu obliczeń wykorzystano moduł time, który pozwala na ocenę wydajności algorytmów, natomiast moduł math został użyty do przeprowadzenia niezbędnych obliczeń matematycznych.

ANALIZA

ZADANIE A

W ramach tego projektu przeprowadzana jest analiza układu równań liniowych opisanych równaniem macierzowym $Ax=b$, gdzie A reprezentuje macierz systemową, b jest wektorem stymulacji, a x jest poszukiwanym wektorem rozwiązań.

Zgodnie z numerem indeksu 193316, dokonuje modyfikacji wartości parametrów w macierzy A oraz wektorze b. Macierz A to macierz pasmowa o wymiarach $N \times N$ z $N=916$. Jest ona zdefiniowana pięcioma diagonalami, z główną diagonalną wypełnioną wartościami $a1=8$. Diagonale bezpośrednio sąsiadujące z główną mają wartości $a2=-1$, a te leżące obok nich, nazywane drugimi sąsiednimi diagonalami, mają wartości $a3=-1$. Taka konfiguracja parametrów skutkuje stworzeniem macierzy o charakterystycznej, pasmowej strukturze, z większością elementów poza wspomnianymi diagonalami jako zerami.

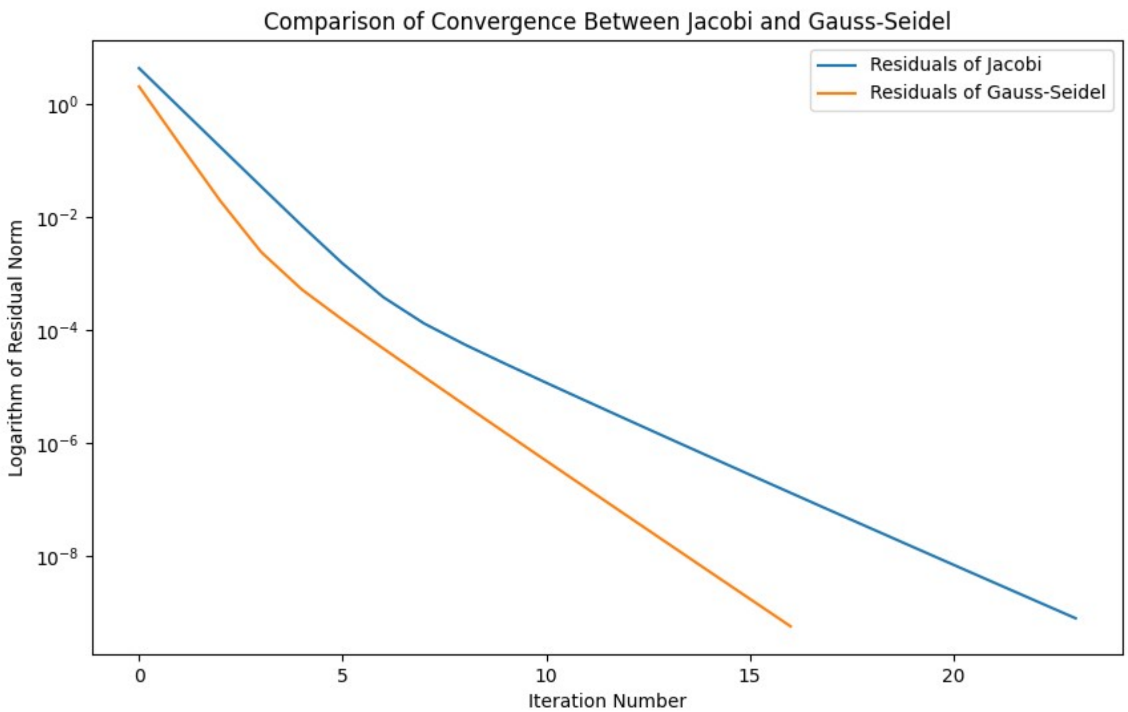
Długość wektora b jest równa N, a każdy z jego elementów, $b[n]$, jest definiowany jako $\sin(n \cdot (3+1))$. Elementy wektora b są wynikiem funkcji sinus dla wartości podwojonych indeksów poszczególnych elementów, co przekłada się na dynamiczne zmiany wartości zależne od pozycji elementu w wektorze.

ZADANIE B

Analiza Wykresu

Na przedstawionym wykresie widać zbieżność obu metod w zależności od liczby iteracji, reprezentowanej na osi poziomej, oraz logarytmicznej skali normy residuum na osi pionowej. Oto kluczowe obserwacje:

- **Zbieżność**: Metoda Gaussa-Seidla wykazuje szybszą zbieżność do rozwiązania w porównaniu z metodą Jacobiego, co widać po stromszej krzywej spadku normy residuum. Ta metoda osiągnęła żądaną precyzję w mniejszej liczbie iteracji.
- **Kształt krzywych**: Krzywe zbiegają płynnie, co wskazuje na stabilność obu metod dla danego układu równań.



Analiza Wydajności

```
Jacobi Method: Iterations = 24, Execution Time = 3.45 seconds
Gauss-Seidel Method: Iterations = 17, Execution Time = 2.43 seconds
```

Dane dotyczące wykonania metod dostarczają dodatkowych informacji na temat ich wydajności:

- **Liczba Iteracji**:
  - **Jacobi**: 24 iteracje
  - **Gauss-Seidel**: 17 iteracji To wskazuje, że metoda Gaussa-Seidla jest bardziej efektywna w osiąganiu wymaganej dokładności (norma residuum mniejsza niż 10^(−9)), co potwierdza obserwacje z wykresu.
- **Czas Wykonania**:
  - **Jacobi**: 3.45 sekundy

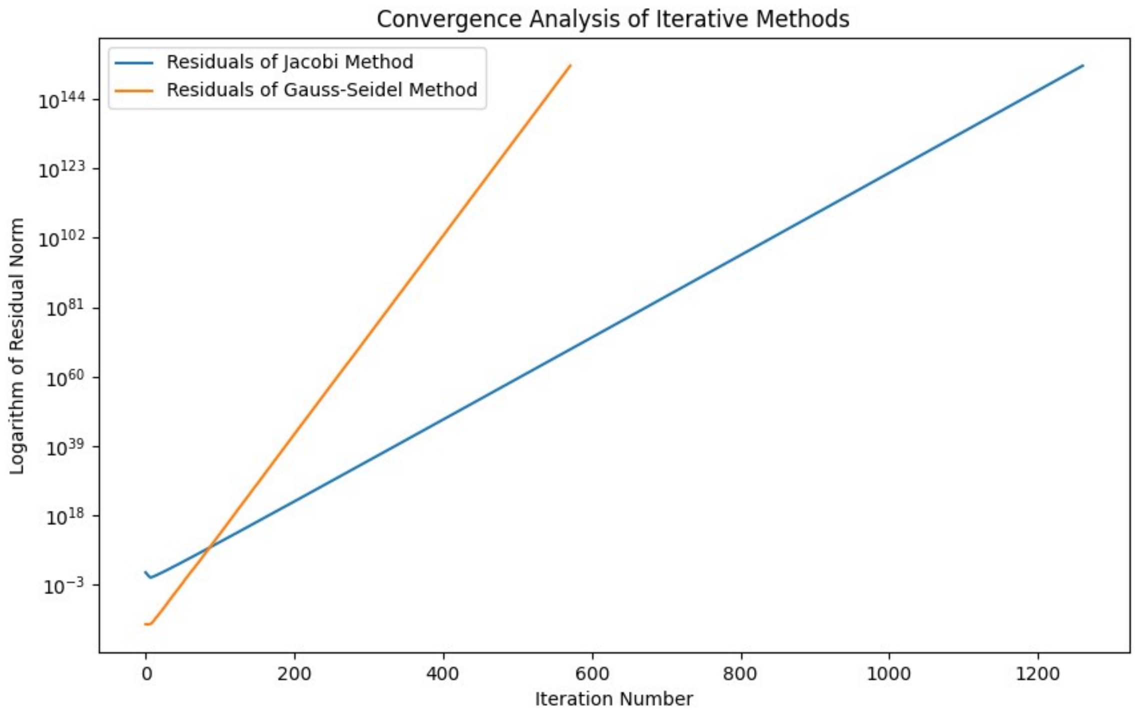
- **Gauss-Seidel**: 2.43 sekundy Również pod względem czasu wykonania metoda Gaussa-Seidla okazuje się być bardziej wydajna. Wynika to z mniejszej liczby potrzebnych iteracji, a także z możliwej większej efektywności samego algorytmu w przetwarzaniu danych.

ZADANIE C

```
Traceback (most recent call last):
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 104, in <module>
    jacobi_solution, jacobi_iters, jacobi_residuals, time_for_jacobi = measure_performance(solve_jacobi, A, b)
                                                                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 99, in measure_performance
    solution, iterations, residuals = solver(system_matrix, vector_b)
                                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 30, in solve_jacobi
    norm_of_residuals = compute_residual_norm(residuals)
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 18, in compute_residual_norm
    return math.sqrt(sum(res ** 2 for res in residuals))
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 18, in <genexpr>
    return math.sqrt(sum(res ** 2 for res in residuals))
                         ~~~~^^~~
OverflowError: (34, 'Result too large')

Traceback (most recent call last):
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 107, in <module>
    seidel_solution, seidel_iters, seidel_residuals, time_for_seidel = measure_performance(perform_gauss_seidel, A, b)
                                                                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 99, in measure_performance
    solution, iterations, residuals = solver(system_matrix, vector_b)
                                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 59, in perform_gauss_seidel
    residual_norm = calculate_residual_norm(matrix, current_estimate, target_vector)
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 49, in calculate_residual_norm
    return math.sqrt(sum(res ** 2 for res in residuals))
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ginko\OneDrive\Pulpit\Programowanko\studia\semestr 4\metody numeryczne\projekt2\projekt2\ZADANIE_C.py", line 49, in <genexpr>
    return math.sqrt(sum(res ** 2 for res in residuals))
                         ~~~~^^~~
OverflowError: (34, 'Result too large')
```

W trakcie obliczeń za pomocą metod Jacobiego i Gaussa-Seidla wystąpił błąd `OverflowError: (34, 'Result too large')`, który objawił się w obu przypadkach. Taki błąd sygnalizuje potencjalne problemy z zbieżnością tych metod dla analizowanego układu równań. Po dokładnej analizie sytuacji można stwierdzić, że zastosowane metody iteracyjne mogą nie być odpowiednie dla tego układu równań z uwagi na dane parametry macierzy A. Zarówno metoda Jacobiego, jak i Gaussa-Seidla, wymagają, aby macierz była przekątniowo dominująca lub spełniała inne specyficzne warunki zapewniające zbieżność. W analizowanym przypadku, gdzie $a1=3$ a $a2=a3=-1$, suma wartości bezwzględnych elementów pozadiagonalnych w każdym wierszu ($|-1|+|-1|+|-1|+|-1|=4$) przewyższa wartość elementu na przekątnej ($|3|$), co zwykle uniemożliwia zbieżność metod iteracyjnych ze względu na brak dominacji przekątniowej w macierzy.



Obserwacje z wykresu

1. **Wzrost residuum:** Analiza wykresu ujawnia, że norma residuum dla obu metod zwiększa się znacząco. Na wykresie zastosowano skalę logarytmiczną, na której linie reprezentujące normę residuum szybko pną się w górę już we wczesnych etapach iteracji.
2. **Porównanie efektywności metod:** Obie metody prezentują tendencję wzrostową w zakresie normy residuum, jednakże metoda Gaussa-Seidla rozpoczyna proces z niższym poziomem residuum i charakteryzuje się wolniejszym tempem wzrostu w porównaniu do metody Jacobiego. To wskazuje, że Gauss-Seidla mogła początkowo wydawać się bardziej skuteczna, ale ostatecznie zarówno ona, jak i metoda Jacobiego, nie osiągają zbieżności.

ZADANIE D

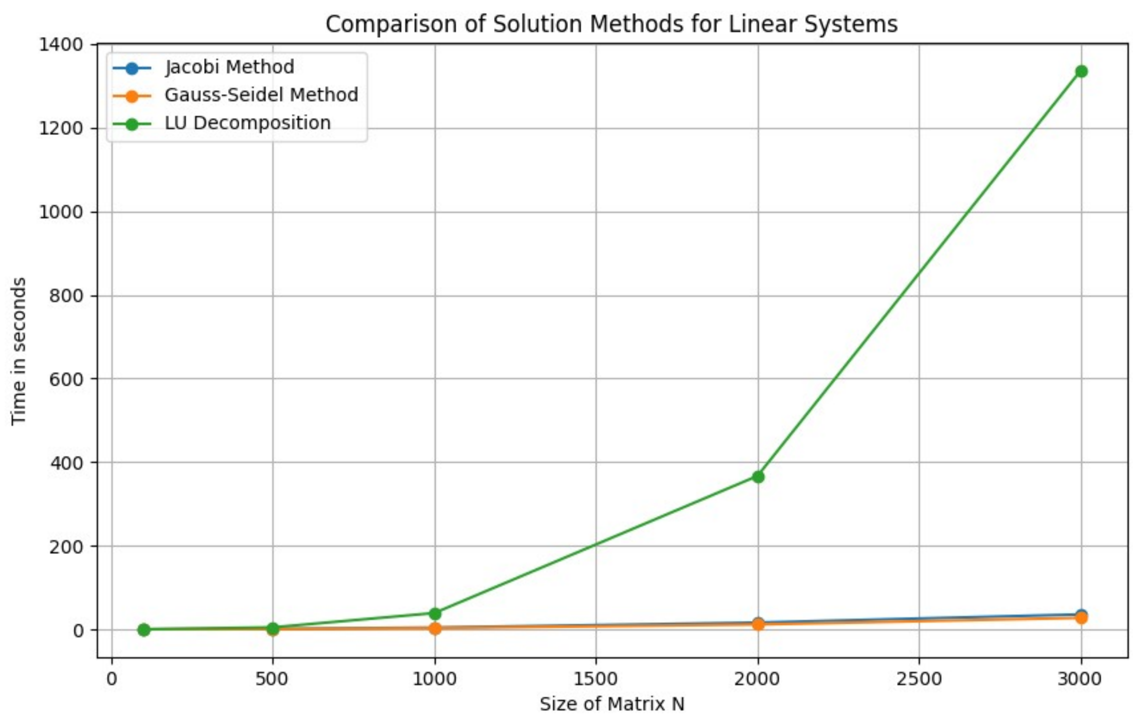
```
residuum norm: 1.760231600176464e-15
```

Skuteczność metody bezpośredniej w porównaniu do metod iteracyjnych: W poprzednim zadaniu, zarówno metoda Jacobiego jak i Gaussa-Seidla nie osiągnęły zbieżności, co było związane z brakiem dominacji przekątniowej macierzy systemowej. Wynikające z tego problemy z zbieżnością w tych metodach iteracyjnych mogły przyczyniać się do narastających błędów i niestabilności numerycznej, co manifestowało się w eksplodujących wartościach norm residuum.

W przypadku użycia metody faktoryzacji LU, będącej metodą bezpośrednią, uzyskane rezultaty były znacznie lepsze. Norma residuum osiągnęła wartość rzędu około 1.76×10^−15, co jest bardzo blisko zera i świadczy o wysokiej dokładności uzyskanego rozwiązania. Ta różnica podkreśla większą stabilność i niezawodność metod bezpośrednich w porównaniu z metodami iteracyjnymi w kontekście analizowanych układów równań.

Zalety faktoryzacji LU: Metoda faktoryzacji LU, realizująca rozwiązanie układu równań bez potrzeby iteracji, zapewnia stabilne i precyzyjne rozwiązania nawet w przypadkach, gdy macierz systemowa nie spełnia warunków koniecznych dla efektywności metod iteracyjnych. Jest efektywna dla każdej macierzy kwadratowej, która jest nieosobliwa, bez względu na jej charakterystykę dominacji przekątnej.
Koszt obliczeniowy: Mimo że metoda faktoryzacji LU jest bardziej wiarygodna i dokładna, wiąże się to z większymi kosztami obliczeniowymi, szczególnie w przypadku dużych macierzy, gdzie proces faktoryzacji LU może być czasochłonny.

ZADANIE E



Analiza Wyników

Metoda Jacobiego i Gaussa-Seidla:

- Zarówno dla metody Jacobiego, jak i Gaussa-Seidla, czas wykonania pozostaje stosunkowo stabilny niezależnie od wielkości macierzy. Wynika to z faktu, że obie te metody iteracyjne wykazują dobrą efektywność przy mniejszych rozmiarach macierzy, a ich złożoność obliczeniowa nie wzrasta tak dynamicznie jak w przypadku metody faktoryzacji LU.

Metoda Faktoryzacji LU:

- Czas potrzebny na przeprowadzenie faktoryzacji LU zwiększa się eksponencjalnie wraz z rosnącym rozmiarem macierzy. Jest to oczekiwany rezultat, biorąc pod uwagę złożoność obliczeniową tej metody, która wynosi $O(N$^3). To sprawia, że metoda ta staje się mniej efektywna w przypadku obsługi dużych macierzy. Znaczne wydłużenie czasu obliczeń dla macierzy o rozmiarach 2000 i 3000 podkreśla intensywne wymagania obliczeniowe tej metody.

ZADANIE F

Zadanie A

W Zadaniu A skonstruowano macierz pasmową z określonymi wartościami na przekątnych, co posłużyło jako baza do analizy metod rozwiązywania układów równań liniowych. Zastosowanie funkcji sinusoidalnej w wektorze b wprowadziło dodatkową złożoność, wpływając na wyniki obliczeń numerycznych. Układ równań zdefiniowano z różnicowaniem wartości poszczególnych elementów macierzy, co miało wpływ na takie właściwości macierzy jak dominacja przekątniowa.

Zadanie B

W Zadaniu B zaimplementowano i przetestowano metody iteracyjne Jacobiego i Gaussa-Seidla, korzystając z macierzy zdefiniowanej w Zadaniu A. Stwierdzono, że te metody napotkały trudności z konwergencją przy specyficznych wartościach elementów macierzy, co było szczególnie widoczne w Zadaniu C. Mimo to, udało się osiągnąć konwergencję przy nieco innych wartościach macierzy. Analiza zbieżności oraz czasu wykonania wskazała, że metody iteracyjne mogą być efektywne, ale ich skuteczność zależy od specyficznych właściwości macierzy.

Zadanie C

Testowanie metod iteracyjnych przy zmienionych wartościach elementów macierzy (a1=3) wykazało brak zbieżności. Te obserwacje pokazały, że zmiana wartości na przekątnej macierzy istotnie wpływa na zbieżność metod iteracyjnych, co podkreśla znaczenie odpowiedniego doboru wartości elementów macierzy w kontekście tych metod.

Zadanie D

W Zadaniu D zastosowano metodę faktoryzacji LU, co umożliwiło bezpośrednie rozwiązanie układu równań, które okazało się problematyczne w metodach iteracyjnych z Zadania C. Uzyskana bardzo niska norma residuum (rzędu 10^−15) potwierdziła wysoką dokładność tej metody, ukazując jej przewagę w sytuacjach, gdy metody iteracyjne zawodzą.

Zadanie E

Analiza czasu obliczeń dla różnych metod w zależności od rozmiaru macierzy wykazała, że czas wykonania metody faktoryzacji LU znacząco wzrasta wraz ze wzrostem rozmiaru macierzy, co jest zgodne z oczekiwaniami złożoności obliczeniowej $O(N^3)$. Metody iteracyjne utrzymały niski czas wykonania przez wszystkie testowane rozmiary, co sugeruje, że mogły nie osiągnąć konwergencji dla większych macierzy.

Ogólne Wnioski

- Metody iteracyjne są przydatne i efektywne dla macierzy o odpowiednich właściwościach, ale ich skuteczność zależy od dominacji przekątniowej i specyfiki macierzy.
- Metoda faktoryzacji LU jest bardziej niezawodna dla złożonych układów równań, ale wiąże się z większym kosztem obliczeniowym, co staje się wyzwaniem przy dużych macierzach.
- Wybór metody rozwiązywania układów równań liniowych powinien uwzględniać zarówno rozmiar problemu, jak i właściwości macierzy, co ma kluczowe znaczenie dla osiągnięcia efektywności obliczeniowej i dokładności wyników.