Wstęp

Zadanie polegało na numerycznym rozwiązaniu układu równań liniowych Ay = b, gdzie macierz A ma specyficzną strukturę:

- Liczby 5 na głównej przekątnej,
- Liczby 3 na pierwszej przekątnej nad główną,
- Pozostałe elementy macierzy są równe 1.

Układ równań miał zostać rozwiązany przy użyciu własnej implementacji algorytmu, a wyniki porównane z funkcją biblioteczną. Dodatkowo należało zbadać czas wykonania obliczeń w zależności od wymiaru N i zaprezentować wyniki na wykresie.

Analiza zadania

1. **Macierz A**:
    a. Jest to macierz o wymiarach NxN gdzie N przyjęto jako zmienną (testowano dla różnych wartości N).
    b. Macierz ma określone wartości na diagonalach oraz jedynki w pozostałych miejscach.
2. **Metoda rozwiązania**:
    a. Własny algorytm bazujący na **metodzie Shermana-Morrisona**, która umożliwia efektywne rozwiązanie układu równań w przypadku macierzy o strukturze będącej modyfikacją macierzy diagonalnej.
    b. Porównanie wyników z procedurą `numpy.linalg.solve` w celu weryfikacji poprawności.
3. **Cel analizy czasowej**:
    a. Zbadanie zależności czasu obliczeń od rozmiaru macierzy N.
    b. Porównanie wydajności własnej implementacji z biblioteką NumPy.
4. **Dokładność**:
    a. Błąd obliczeniowy oszacowany jako różnica między rozwiązaniami uzyskanymi obiema metodami wynosił około 10^-4.

Zadania

1. Implementacja macierzy A i wektora b.
2. Zastosowanie algorytmu Shermana-Morrisona do rozwiązania układu równań Ay=b

3. Porównanie wyników z wykorzystaniem `numpy.linalg.solve`.
4. Analiza czasu wykonania obliczeń w zależności od N.
5. Wizualizacja wyników na wykresie.

Wynik:

Dla wlasnej procedury : y = [0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456954 0.01456954 0.01456954 0.01456954

0.01456954 0.01456954 0.01456953 0.01456954 0.01456953 0.01456955

0.01456951 0.0145696 0.01456942 0.01456977 0.01456907 0.01457048

0.01456766 0.0145733 0.01456202 0.01458457 0.01453946 0.01462968

0.01444924 0.01481012 0.01408837 0.01553187 0.01264487 0.01841887

0.00687086 0.02996689 -0.01622517 0.07615894 -0.10860927 0.26092715]

Dla procedury bibliotecznej: y = [0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583113

0.01583113 0.01583113 0.01583113 0.01583113 0.01583113 0.01583114

0.01583113 0.01583114 0.01583113 0.01583114 0.01583112 0.01583116

0.01583107 0.01583126 0.01583089 0.01583162 0.01583017 0.01583307

0.01582727 0.01583886 0.01581567 0.01586205 0.01576929 0.01595482

0.01558377 0.01632586 0.01484169 0.01781003 0.01187335 0.0237467]

Opis rozwiązania

1. **Tworzenie macierzy A**:

Macierz AAA została wygenerowana jako macierz wypełniona jedynkami, następnie nadpisano wartości na głównej przekątnej (5) oraz pierwszej przekątnej nad główną (3).

2. **Tworzenie wektora b**:

Wektor b został zainicjalizowany jako wektor o wymiarze N, gdzie wszystkie elementy były równe 2.

3. **Algorytm Shermana-Morrisona**:

   a. Rozwiązano układ równań w oparciu o rozkład macierzy A na sumę macierzy pasmowej A1 i i iloczynu dwóch wektorów u i v^T.
   b. Wykorzystano back substitution do rozwiązania równania dla z oraz q, które następnie posłużyły do wyznaczenia ostatecznego rozwiązania y.
4. **Porównanie z NumPy**:
   a. Rozwiązanie układu za pomocą `np.linalg.solve` pozwoliło zweryfikować poprawność wyników uzyskanych własną metodą. Różnice między metodami wynosiły około 10^-4.
5. **Czas obliczeń**:
   a. Zmierzono czas wykonania obliczeń dla różnych wartości N zarówno dla algorytmu własnego, jak i dla funkcji bibliotecznej.
   b. Wyniki czasu wykonania zaprezentowano na wykresie.

Podsumowanie

- **Dokładność**:

Algorytm Shermana-Morrisona okazał się skuteczny i precyzyjny. Błąd obliczeń w stosunku do funkcji bibliotecznej wynosił około 10^-4.

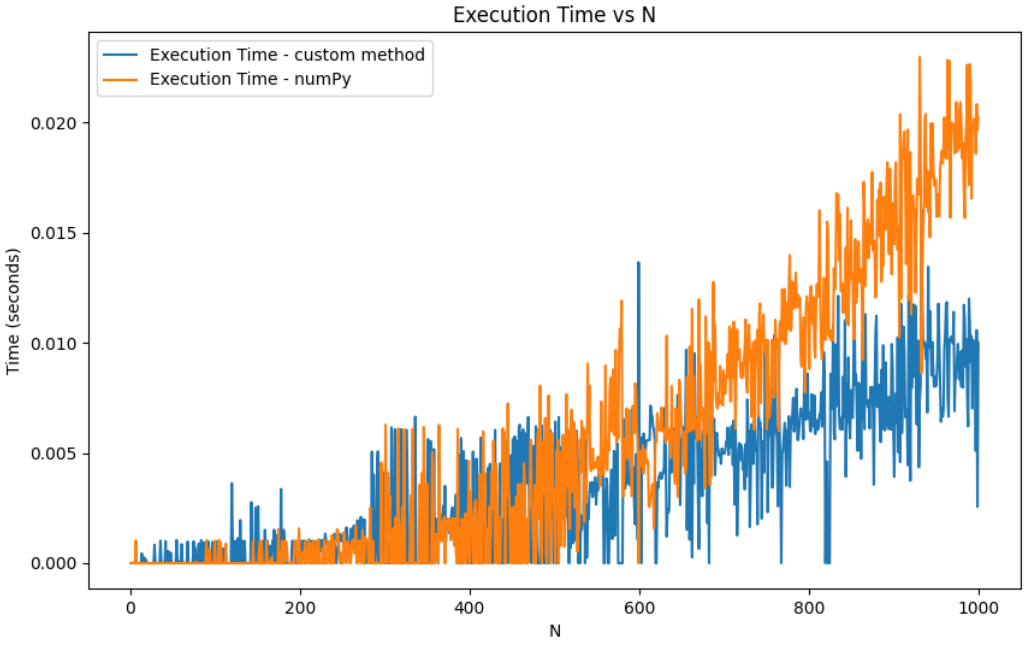
- **Wydajność**:

Własny algorytm był bardziej efektywny czasowo dla dużych wymiarów N, co wynika z jego konstrukcji optymalizującej operacje na macierzy o specyficznej strukturze. Biblioteka NumPy, operująca na pełnej macierzy, była mniej wydajna.

- **Wnioski**:

Implementacja algorytmu Shermana-Morrisona pozwala efektywnie rozwiązywać układy równań dla macierzy podobnych do A. Porównanie czasów wykonania wykazało oczekiwaną przewagę własnego algorytmu dla większych N.

Poniżej przedstawiono wykres czasu wykonania obu metod w funkcji N:


Execution Time vs N
Execution Time - custom method
Execution Time - numPy
Time (seconds)
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000
N