Metody Numeryczne – Projekt 2

Układy równań liniowych

Justyna Dąbrowska, 185872, Inf sem 4

1. Wstęp

Celem projektu było zaimplementowanie metod rozwiązywania układów równań liniowych. Należało zaimplementować dwie metody iteracyjne: Jacobiego i Gaussa-Seidla oraz metodę bezpośrednią czyli faktoryzację LU. Do implementacji zadania wybrałam język Python dołączając biblioteki `matplotlib`, `copy`, `math`, `time` i `decimal`.

2. Zadanie A

Na podstawie mojego numeru indeksu uzyskałam takie własności parametrów wykonania zadania:

```
|============== MATRIX PARAMETERS ==============|
|      a1=13     a2=-1     a3=-1     N=972       |
```

Utworzyłam macierz pasmową `A` o wymiarach 972x972 na podstawie wytycznych projektu i powyższych wartości:

- Na głównej diagonali znajduje się: `a1 = 8 + 5 = 13`
- Na sąsiednich diagonalach są elementy: `a2 = -1`
- Na skrajnych diagonalach występują elementy: `a3 = -1`

Prawa stronę równania, na którym bazujemy stanowi wektor `b` o długości `N = 972`, którego elementy należało policzyć ze wzoru: `n-ty element = sin(n·(5+1))`.

3. Zadanie B

W celu realizacji zadania B zostały zaimplementowane metody iteracyjne: Jacobiego i Gaussa-Seidla, które zostały użyte do rozwiązania układu równań: `Ax = b`, gdzie `A` i `b` zostały wyliczone w zadaniu A, a `x` początkowo jest wektorem o długości `N = 972`, który przyjmuje wartość `1` dla każdego elementu.

Oto wyniki jakie otrzymałam po przeprowadzeniu obliczeń dla danych z zadania A:

```
|============== MATRIX PARAMETERS ==============|
|      a1=13     a2=-1     a3=-1     N=972       |

|**************** Jacobi Method ****************|
|-> Time: 3.4307098388671875 [s]                 |
|-> Iteration: 23                                |
|-> Residuum: 4.691979296758424e-10              |

|************* Gauss-Seidel Method *************|
|-> Time: 2.4680674076080322 [s]                 |
|-> Iteration: 16                                |
|-> Residuum: 3.9624918094717724e-10             |
```

Metoda Gaussa-Seidla jest metodą szybszą od metody Jacobiego oraz wymaga mniejszej liczby iteracji ze względu na to, że wykorzystuje aktualnie dostępne przybliżone rozwiązania, co pozwala na zaoszczędzenie pamięci operacyjnej i zmniejszenie liczby obliczeń.

4. Zadanie C

Warunkiem poprawności działania metod iteracyjnych jest to, aby wektor przybliżonych rozwiązań w każdej następnej iteracji zbiegał się do rozwiązania dokładnego. Aby sprawdzić czy metody zbiegają się dodano dodatkowy warunek zakończenia metody, ograniczając z góry normę z wektora residuum przez wartość 10^9 , po przekroczeniu tej wartości metoda jest zakańczana, gdyż norma z wektora residuum rośnie, co oznacza, że rozwiązanie przybliżone wraz z następną iteracją oddala się od rozwiązania dokładnego.

W celu realizacji tego zadania wektor b pozostał bez zmian, a korzystając z tej samej funkcji tworzącej macierz A utworzyłam macierz o podanych parametrach:

```
|============== MATRIX PARAMETERS ==============|
|      a1=3     a2=-1     a3=-1     N=972       |

|***************** Jacobi Method *****************|
|-> Time: 10.469768524169922 [s]                 |
|-> Iteration: 69                                |
|-> Residuum: Bigger than 12814939418.615734

|************* Gauss-Seidel Method *************|
|-> Time: 4.591810703277588 [s]                  |
|-> Iteration: 29                                |
|-> Residuum: Bigger than 16168943898.25929
```

Zadanej macierzy C nie udaje się zredukować za pomocą metod iteracyjnych, ponieważ zauważa się wzrost normy wektora residuum wraz z każdą kolejną iteracją. Dla metody Jacobiego norma przekroczyła wartość 10^9 po 69 iteracjach, a dla metody Gaussa-Seidla po 29 iteracjach.

5. Zadanie D

W tym zadaniu należało zaimplementować bezpośrednią metodę rozwiązania układu równań – faktoryzację LU. Trzeba było ją przetestować na macierzy z zadania C, czego rozwiązanie nie powiodło się używając metod iteracyjnych.

```
|************** LU  Factorization **************|
|-> Time: 28.163785457611084 [s]                 |
|-> Iteration: 0                                 |
|-> Residuum: 4.3793975541983465e-13             |
```

Faktoryzacja LU wymaga znacznie więcej czasu niż metody iteracyjne, ale pozwala na uzyskanie rozwiązania przybliżonego, które jest zbliżone do rozwiązania dokładnego (na co wskazuje na niska wartość normy wektora residuum). Zastosowanie metod iteracyjnych nie doprowadziło do uzyskania zbliżonego do dokładnego rozwiązania, ponieważ kolejne przybliżenia dla tych metod nie zbiegały się.

6. Zadanie E

Realizacja zadania polegała na stworzeniu wykresu zależności czasu trwania poszczególnych metod rozwiązujących układy równań od liczby niewiadomych dla `N = [100, 500, 1000, 2000, 3000]` dla macierzy utworzonej w zadaniu A.

Wyniki dla macierzy 100x100

```
|============== MATRIX PARAMETERS ==============|
|      a1=13     a2=-1     a3=-1     N=100      |

|**************** Jacobi Method ****************|
|-> Time: 0.04062342643737793 [s]               |
|-> Iteration: 22                               |
|-> Residuum: 4.546085178440576e-10             |

|************* Gauss-Seidel Method *************|
|-> Time: 0.028053760528564453 [s]
|-> Iteration: 15                               |
|-> Residuum: 6.332504949338986e-10             |

|************** LU  Factorization **************|
|-> Time: 0.03960418701171875 [s]               |
|-> Iteration: 0                                |
|-> Residuum: 1.0685446024991062e-15            |
```

Wyniki dla macierzy 500x500

```
|============== MATRIX PARAMETERS ==============|
|      a1=13     a2=-1     a3=-1     N=500      |

|**************** Jacobi Method ****************|
|-> Time: 0.9297289848327637 [s]                |
|-> Iteration: 23                               |
|-> Residuum: 3.339431902163279e-10             |

|************* Gauss-Seidel Method *************|
|-> Time: 0.6617710590362549 [s]                |
|-> Iteration: 16                               |
|-> Residuum: 2.813004672470581e-10             |

|************** LU  Factorization **************|
|-> Time: 3.7476117610931396 [s]                |
|-> Iteration: 0                                |
|-> Residuum: 2.750386095092008e-15             |
```

Wyniki dla macierzy 1000x1000

```
|============== MATRIX PARAMETERS ==============|
|      a1=13     a2=-1     a3=-1     N=1000     |

|**************** Jacobi Method ****************|
|-> Time: 3.7824149131774902 [s]                |
|-> Iteration: 23                               |
|-> Residuum: 4.759300502207373e-10             |

|************* Gauss-Seidel Method *************|
|-> Time: 2.641526937484741 [s]                 |
|-> Iteration: 16                               |
|-> Residuum: 4.018934733680632e-10             |

|************** LU  Factorization **************|
|-> Time: 30.66653347015381 [s]                 |
|-> Iteration: 0                                |
|-> Residuum: 3.869855306645312e-15             |
```

Wyniki dla macierzy 2000x2000

```
|============== MATRIX PARAMETERS ==============|
|      a1=13     a2=-1     a3=-1     N=2000     |

|**************** Jacobi Method ****************|
|-> Time: 15.128153562545776 [s]                |
|-> Iteration: 23                               |
|-> Residuum: 6.756992208618756e-10             |

|************* Gauss-Seidel Method *************|
|-> Time: 11.155796766281128 [s]                |
|-> Iteration: 16                               |
|-> Residuum: 5.715548906046067e-10             |

|************** LU  Factorization **************|
|-> Time: 241.74850726127625 [s]                |
|-> Iteration: 0                                |
|-> Residuum: 5.400202307926871e-15             |
```

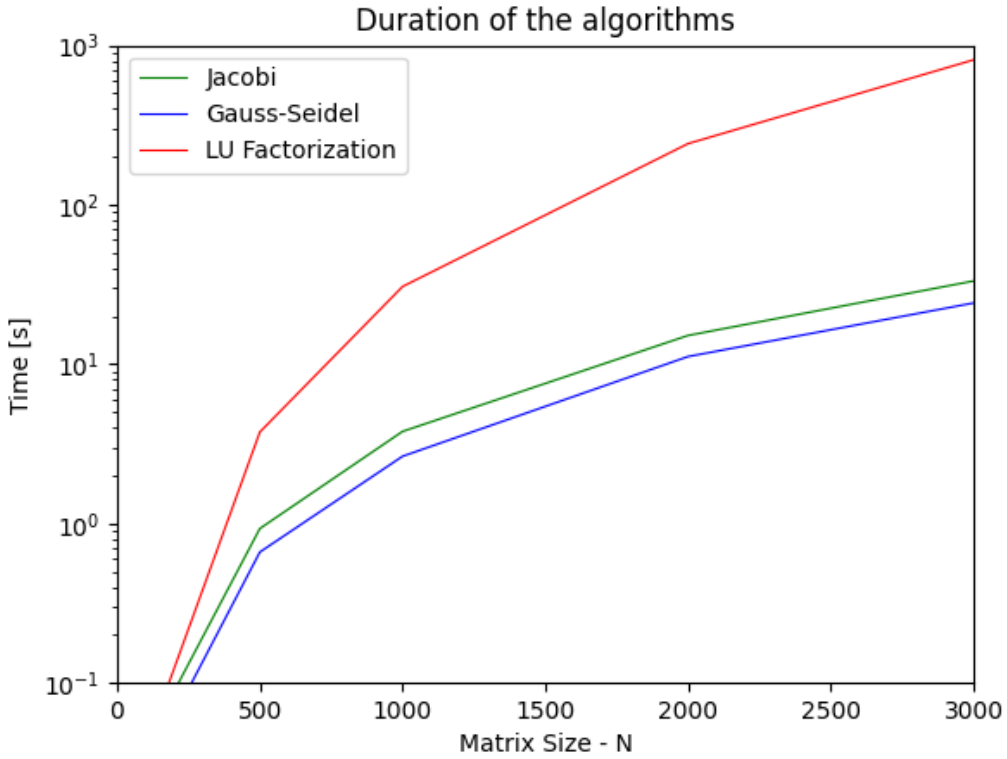
Wyniki dla macierzy 3000x3000

```
|============== MATRIX PARAMETERS ==============|
|      a1=13     a2=-1     a3=-1     N=3000     |

|**************** Jacobi Method ****************|
|-> Time: 33.23600649833679 [s]                 |
|-> Iteration: 23                               |
|-> Residuum: 8.286363208993856e-10             |

|************* Gauss-Seidel Method *************|
|-> Time: 24.183138608932495 [s]                |
|-> Iteration: 16                               |
|-> Residuum: 7.013202186327682e-10             |

|************** LU  Factorization **************|
|-> Time: 810.6885333061218 [s]                 |
|-> Iteration: 0                                |
|-> Residuum: 6.5769921724105e-15               |
```



Wykres 1: Porównanie czasów wykonania algorytmów dla zaimplementowanych metod rozwiązujących układ równań liniowych dla macierzy N = [100, 500, 1000, 2000, 3000]

Wykres został stworzony za pomocą biblioteki `matplotlib`, używając skali logarytmicznej na osi y aby wyniki były lepiej widoczne.

Z wykresu można odczytać, że czas rozwiązania układów równań przez metody iteracyjne jest dużo krótszy niż w przypadku metody faktoryzacji LU, trend wzrostu dla obu metod iteracyjnych jest podobny. Jak widać wraz ze wzrostem liczby niewiadomych dla wszystkich metod czas rośnie znacząco, ale dla metody bezpośredniej wzrasta diametralnie aż do 13,5 minut.

7. Wnioski

Metoda faktoryzacji LU, która jest jednym z wariantów metody eliminacji Gaussa, znajduje zastosowanie w przypadku, gdy układ równań musi być rozwiązany dla wielu różnych prawych stron (wektorów b), przy czym lewa strona równania (macierz A) pozostaje bez zmian. Chociaż obliczenie wyników za pomocą tej metody trwa dłużej, to gwarantuje ona uzyskanie dokładnego rozwiązania w przypadku, gdy dla niektórych wartości macierzy A metody iteracyjne nie są w stanie zapewnić zbieżności wektora z rozwiązaniami przybliżonymi do wektora z rozwiązaniami dokładnymi. Metody iteracyjne są jednak znacznie szybsze niż metody bezpośrednie, dlatego w przypadku braku zbieżności można poprawić ich zbieżność przez dobór odpowiednich parametrów.