Układy równań liniowych

Sebastian Kutny, 188586, Informatyka, sem. 4, gr. 3

1. Wstęp

Celem projektu było zaimplementowanie metod iteracyjnych: Jacobiego oraz Gaussa-Seidla i bezpośredniej: Faktoryzacja LU do rozwiązywania układów równań liniowych. Do realizacji wykorzstałem język Python oraz biblioteki: time – zmierzenie czasu wykonywania algorytmu, math – obliczenia matematyczne, matplotlib – wykres prezentujący czasy wykonywania algorytmów dla różnych ilości niewiadomych.

2. Analiza zadania

Zadanie A:

Dla mojego indeksu – 188586 – otrzymujemy wartości a1 = 10, a2 = a3 = -1 dla macierzy A oraz wielkość jej i wektora b: N = 986. N-ty element wektora b wynosi sin(n*(8*1)).

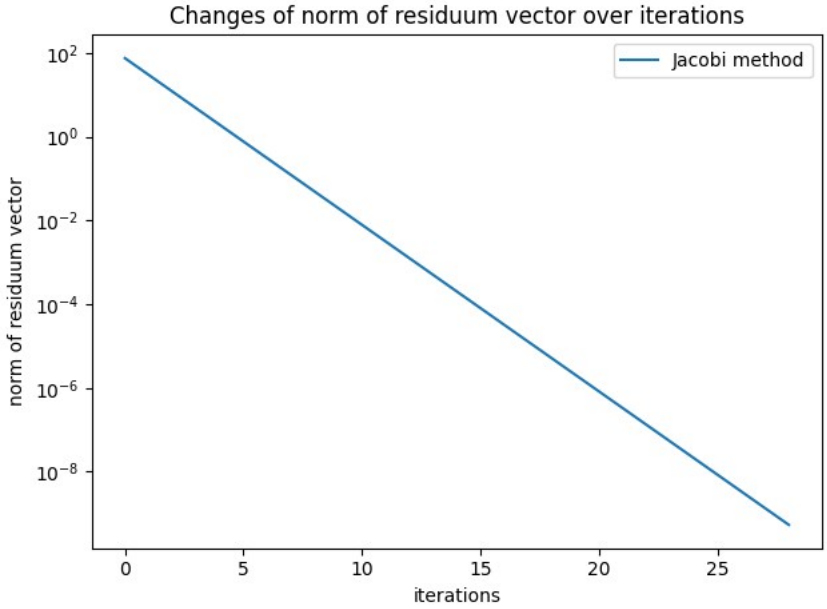
Zadanie B:

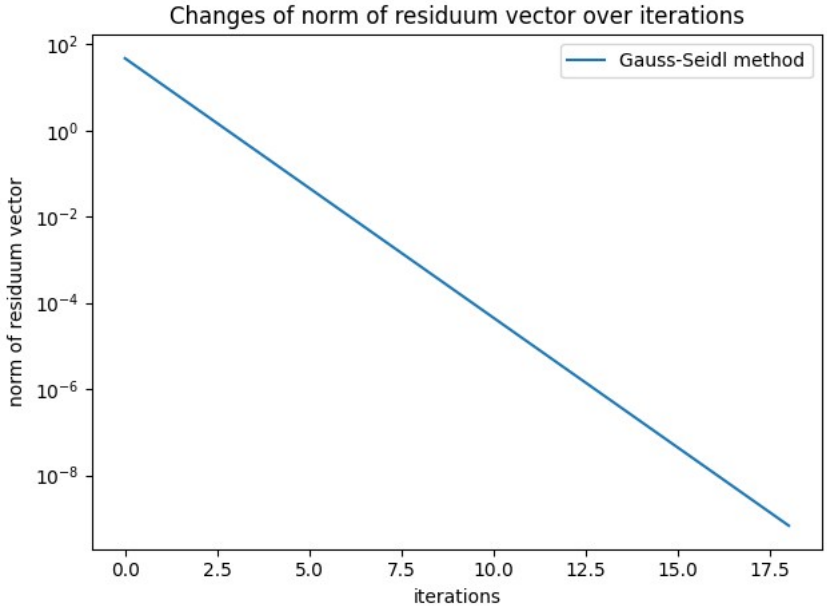
Wyniki dla układu równań stworzonego w punkcie A, przy ustalonym progu dla wartości normy wektora residuum równym 10^-9, dla metody Jacobiego oraz Gaussa-Seidla prezentują się następująco:

```
Jacobi method:
Iterations:  29
Duration [s]:  6.388030290603638
Norm of residual vector:  5.3819420373203 74e-10
Gauss-Seidl method:
Iterations:  19
Duration [s]:  4.203457832336426
Norm of residual vector:  6.769535723141692e-10

Process finished with exit code 0
```

Metoda Gaussa-Seidla sprawdza się lepiej zarówno czasowo jak i dla liczby iteracji potrzebnej do wykonania algorytmu.

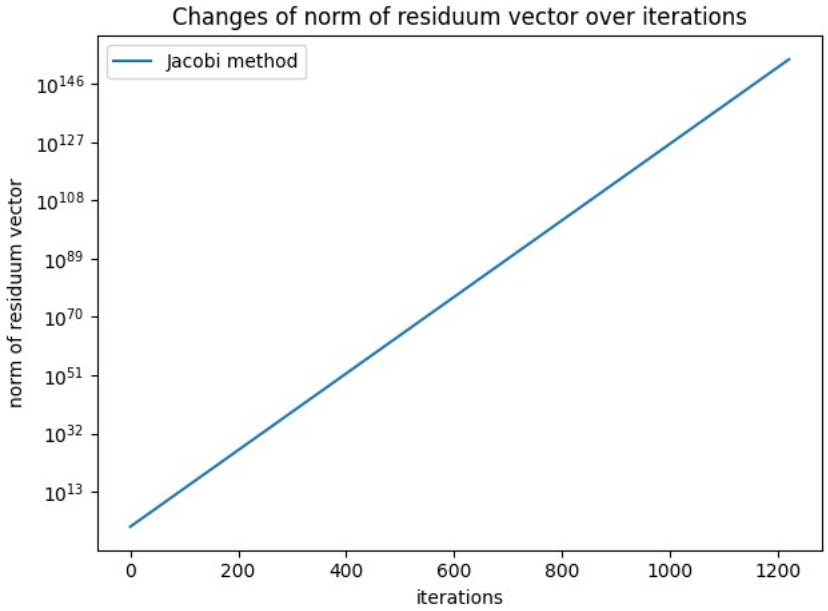

Changes of norm of residuum vector over iterations
Jacobi method
norm of residuum vector
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
0
5
10
15
20
25
iterations

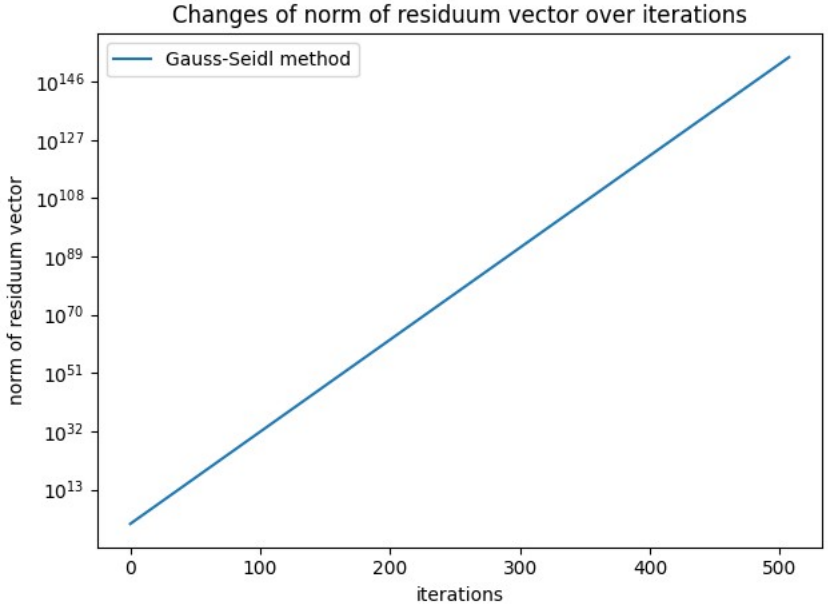

Changes of norm of residuum vector over iterations
Gauss-Seidl method
norm of residuum vector
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
iterations

Zadanie C:
Dla układu równań ze zmienionym parametrem a1 macierzy A na wartość 3 metody iteracyjne nie zbiegają się. Wnioskuję to po nietypowo długim czasie wykonywania się programu, wzroście do nieskończoności normy wektora residuum zamiast jej spadku oraz samym wyjątku spowodowanym przechowywaniem zbyt dużej liczby w pamięci komputera.

```
Traceback (most recent call last):
  File "C:\Users\sebku\Downloads\MN\Projekt\MN2\main.py", line 187, in <module>
    main()
  File "C:\Users\sebku\Downloads\MN\Projekt\MN2\main.py", line 135, in main
    jacobi_result = Jacobi(A, b, treshold)
  File "C:\Users\sebku\Downloads\MN\Projekt\MN2\main.py", line 65, in Jacobi
    res_norm = norm(residuum(A, x, b))
  File "C:\Users\sebku\Downloads\MN\Projekt\MN2\main.py", line 50, in norm
    return math.sqrt(sum(x ** 2 for x in vector))
  File "C:\Users\sebku\Downloads\MN\Projekt\MN2\main.py", line 50, in <genexpr>
    return math.sqrt(sum(x ** 2 for x in vector))
OverflowError: (34, 'Result too large')

Process finished with exit code 1
```





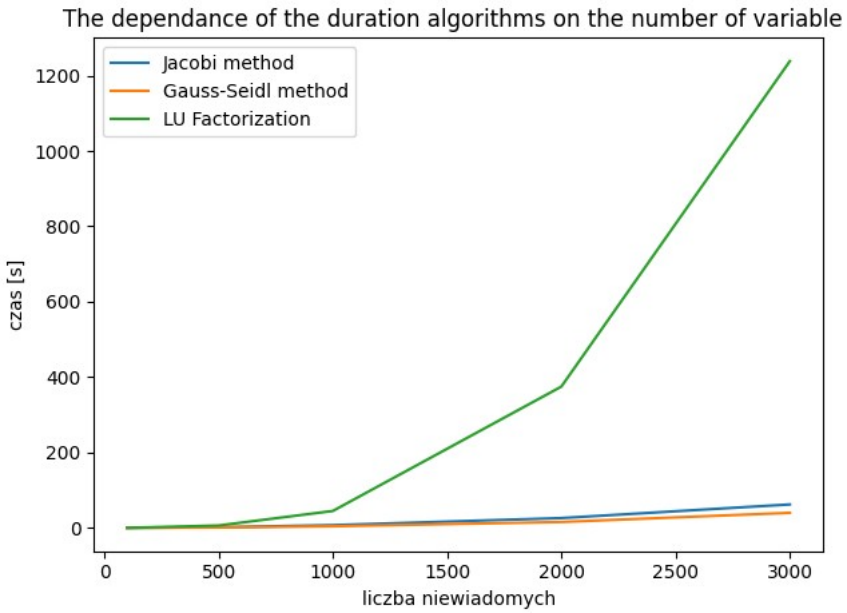
Zadanie D:
Wyniki dla układu równań stworzonego w punkcie C, przy wykorzystaniu metody faktoryzacji LU prezentują się następująco:

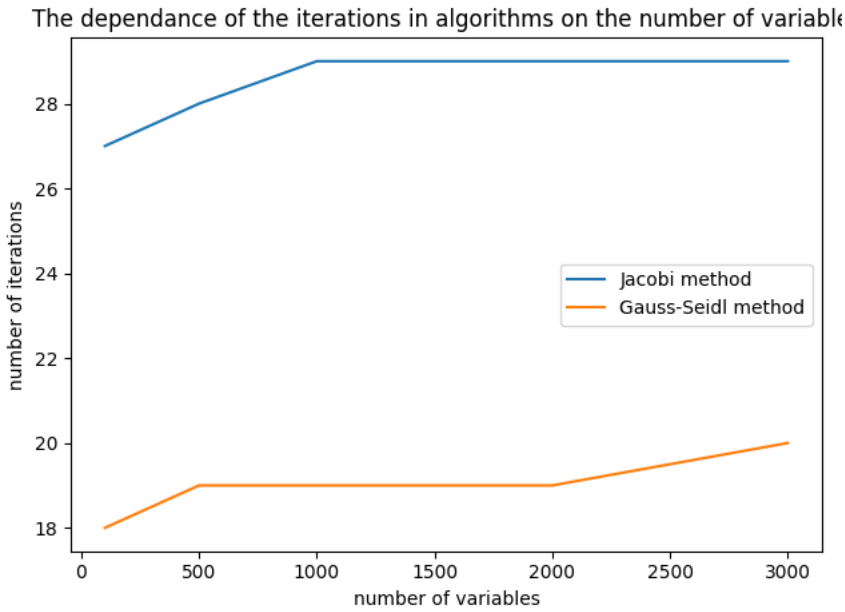
```
LU Factorization method:
Duration [s]:  44.43246054649353
Norm of residual vector:  5.121114426807859e-13

Process finished with exit code 0
```

Wynik normy residuum jest bardzo bliski zera co oznacza wysoką dokładność obliczeń algorytmu.

Zadanie E:
Poniżej prezentuję wykres zależności trwania obliczeń poszczególnych algorytmów zależnie od ilości niewiadomych w podanych liczbach N = {100, 500, 1000, 2000, 3000} dla układu równań z punktu A:




The dependance of the iterations in algorithms on the number of variable
number of iterations
number of variables
Jacobi method
Gauss-Seidl method
18
20
22
24
26
28
0
500
1000
1500
2000
2500
3000

Czas obliczeń każdego algorytmu wzrasta wraz z ilością niewiadomych. Metoda Gaussa-Seidla okazała się najszybsza w każdym przypadku. Faktoryzacja LU jest bardzo kosztowna czasowo i raptownie rośnie wraz ze wzrostem ilości niewiadomych.

```
N:  100
Jacobi method:
Duration:  0.0639655590057373
Gauss-Seidl method:
Duration:  0.04099464416503906
LU Factorization method:
Duration:  0.04595232009887695

N:  500
Jacobi method:
Duration:  1.645258903503418
Gauss-Seidl method:
Duration:  0.9946548938751221
LU Factorization method:
Duration:  6.215157747268677

N:  1000
Jacobi method:
Duration:  7.179279804229736
Gauss-Seidl method:
Duration:  4.2611870765686035
LU Factorization method:
Duration:  44.82562875747681

N:  2000
Jacobi method:
Duration:  25.9583101272583
Gauss-Seidl method:
Duration:  15.596915245056152
LU Factorization method:
Duration:  374.7265827655792

N:  3000
Jacobi method:
Duration:  62.13475275039673
Gauss-Seidl method:
Duration:  39.87706470489502
LU Factorization method:
Duration:  1238.1301608085632


Process finished with exit code 0
```

Zadanie F:

Czas wykonywania obliczeń wzrasta wraz z ilością niewiadomych dla każdego algorytmu, lecz metody iteracyjne dla przypadków zbiegających się pozwalają na obliczenie odpowiednio wymaganego dokładnego wyniku w rozsądnym czasie, w przeciwieństwie do dokładnych, ale długo wykonujących się metod bezpośrednich. Niestety nie każdy układ równań liniowych da się rozwiązać metodami iteracyjnymi, więc trzeba skorzystać z metod bezpośrednich, niezależnie od wymaganego czasu. W metodach iteracyjnych, metoda Gaussa-Seidla w każdym przypadku przeważyła nad metodą Jacobiego co oznacza jej lepszą skuteczność pod względem czasowym oraz liczbą iteracji.