AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Metody obliczeniowe w nauce i technice

Laboratorium 5

Jan Gołda, WIET 2018/2019

1. Wprowadzenie

Laboratorium miało na celu zapoznanie się z iteracyjnymi metodami rozwiązywania równań układów liniowych.

Zadanie wykonano w Jupyter Notebook, pisząc w Pythonie i używając biblioteki do obliczeń naukowych o nazwie NumPy.

Wykorzystany kod oraz wyniki pomiarów są dostępne w repozytorium.

Dla uproszczenia przyjęto że macierz współczynników jest kwadratowa.

1.1. Funkcje pomocnicze

W programie wykorzystano następujące funkcje pomocnicze:

```
# sprawdza czy macierz jest dominująca
def is_diagonally_dominant(A):
    for i in range(A.shape[0]):
        if 2 * abs(A[i,i]) < np.sum(np.abs(A[i,:])):
            return False
    return True
```

```
# tworzy losową dominującą macierz
def random_diagonally_dominant(n):
    A = np.random.rand(n,n) * 2 - 1
    for i in range(n):
        A[i,i] = np.sum(np.abs(A[i,:]))
    return A
```

1.2. Macierze pomocnicze

$$A = L + D + U$$

```
def matrix_L(A):
    return np.tril(A, -1)

def matrix_D(A):
    return np.diag(np.diag(A))

def matrix_U(A):
    return np.triu(A, 1)
```

$$R = L + U$$

```
def matrix_R(A):
    return matrix_L(A) + matrix_U(A)
```

$$S = (L + D)^{-1}$$

```
def matrix_S(A):
    return np.linalg.inv(matrix_L(A) + matrix_D(A))
```

$$N = D^{-1}$$

```
def matrix_N(A):
    return np.linalg.inv(np.diag(np.diag(A)))
```

3. Metoda Gaussa-Seidela

Metoda Gaussa-Seidela dana jest następującym wzorem:

$$L_* X^{(k+1)} = B - UX^{(k)}$$

Co jest równoważne:

$$X^{(k+1)} = -(L+D)^{-1}UX^{(k)} + (L+D)^{-1}B$$

Co korzystając z wcześniejszych macierzy wygląda:

$$X^{(k+1)} = -SUX^{(k)} + SB$$

Jako że $-SU$ i SB nie są zależne od X, są stałe. Zatem dla przyspieszenia obliczeń obliczymy je tylko raz:

$$T_2 = -SU$$

$$C_2 = SB$$

$$X^{(k+1)} = T_2 X^{(k)} + C_2$$

3.1. Implementacja

```
def matrix_T2(A):
    return -1 * np.dot(matrix_S(A), matrix_U(A))
```

```
def matrix_C2(A, B):
    return np.dot(matrix_S(A), B)
```

```
def gauss_seidel_compute(T2, C2, X):
    return np.dot(T2, X) + C2
```

```
def gauss_seidel(A, B, X, iterations):
    if not is_diagonally_dominant(A):
        raise ValueError("Given matrix is not diagonally dominant")
    T2 = matrix_T2(A)
    C2 = matrix_C2(A, B)
    for i in range(iterations):
        X = gauss_seidel_compute(T2, C2, X)
    return X
```

4. Testy

Podczas manualnych testów porównano wyniki obydwu metod, przyrównując je do wyniku dokładnego, wyznaczonego przy pomocy funkcji bibliotecznej.

Za każdym razem dokonano 25 iteracji.

4.1. Test 1: Macierz 2x2

Układ wejściowy:

A		B
2	1	11
5	7	13

Wyniki:

Jacobi	Gauss-Seidel	NumPY
7.11110202	7.11111111	7.11111111
-3.22220342	-3.22222222	-3.2222222

4.2. Test 2: Macierz 4x4

Układ wejściowy:

A				B
10	-1	2	0	6
-1	11	-1	3	25
2	-1	10	-1	-11
0	3	-1	8	15

Wyniki:

Jacobi	Gauss-Seidel	NumPY
1	1	1
2	2	2
-1	-1	-1
1	1	1

4.3. Test 3: Macierz 3x3

Układ wejściowy:

A			B
4	-1	-1	3
-2	6	1	9
-1	1	7	-6

Wyniki:

Jacobi	Gauss-Seidel	NumPY
1	1	1
2	2	2
-1	-1	-1

4.4. Test 4: Macierz nie spełniająca warunku

Układ wejściowy:

A			B
4	-1	-1	3
-2	1	1	9
-1	1	7	-6

Wyniki:

Jacobi	Gauss-Seidel	NumPY
5.9710198	5.99999762	6
24.56950449	24.49999439	24.5
-3.48213643	-3.49999954	-3.5

4.5. Wnioski

Jak widać obydwie metody działają poprawnie o ile spełniony jest warunek że macierz jest przekątniowo dominująca.

Co ciekawe dla testu w którym warunek nie jest spełniony obydwie metody nie znacznie się pomyliły.

5. Porównanie zbieżności

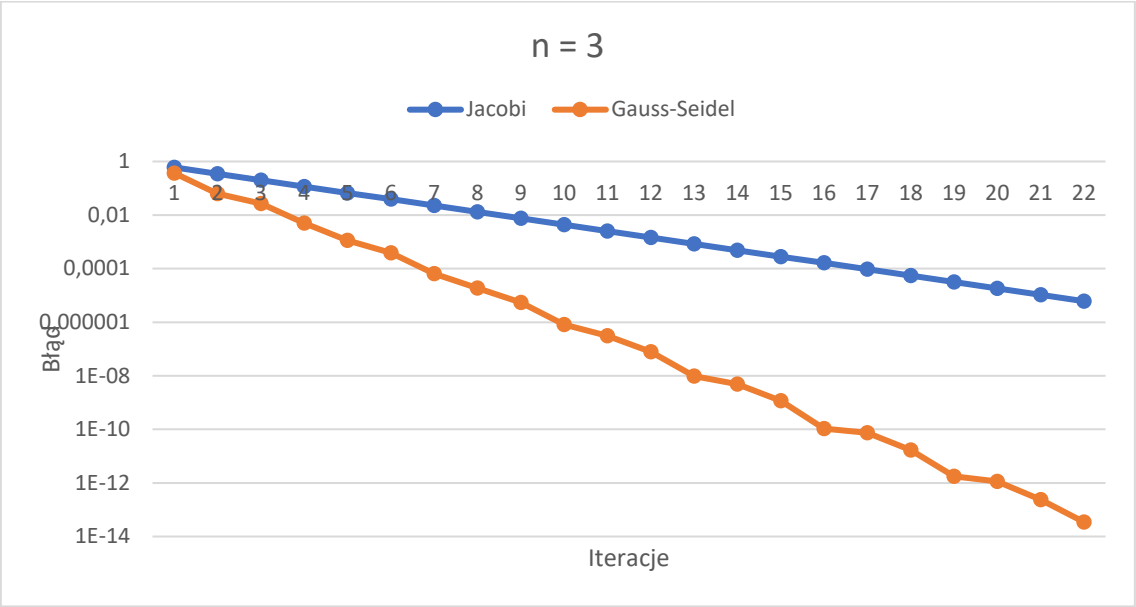
Do porównania zbieżności wykorzystano następującą funkcję:

```
def test(n):
    A = random_diagonally_dominant(n)
    B = np.random.rand(n,1)
    T1 = matrix_T1(A)
    T2 = matrix_T2(A)
    C1 = matrix_C1(A, B)
    C2 = matrix_C2(A, B)
    Xj = np.zeros((n,1))
    Xg = np.zeros((n,1))
    X = np.linalg.solve(A, B)

    data = list()

    while (not np.allclose(Xj, X)) or (not np.allclose(Xg, X)):
        Xj = jacobi_compute(T1, C1, Xj)
        Xg = gauss_seidel_compute(T2, C2, Xg)

        print(str(np.sum(np.abs(X - Xj))) + "," + str(np.sum(np.abs(X
- Xg))))
```

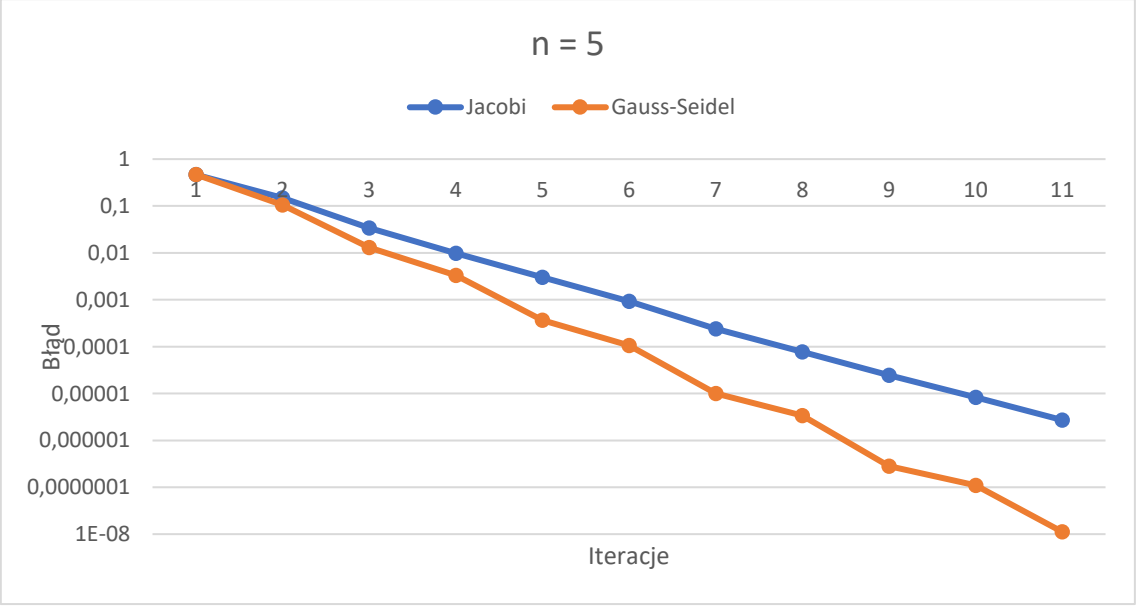
5.1. Wyniki

Wyniki zostały przedstawione w **skali logarytmicznej**, tak aby było widać różnicę pomiędzy wynikami.

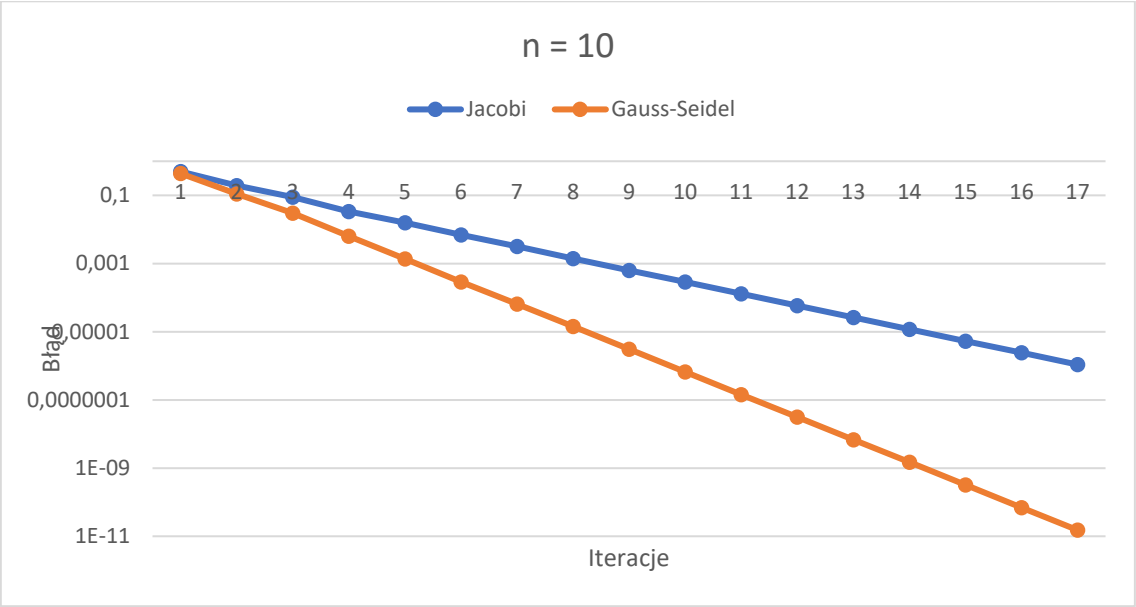
5.2. Macierz 3x3



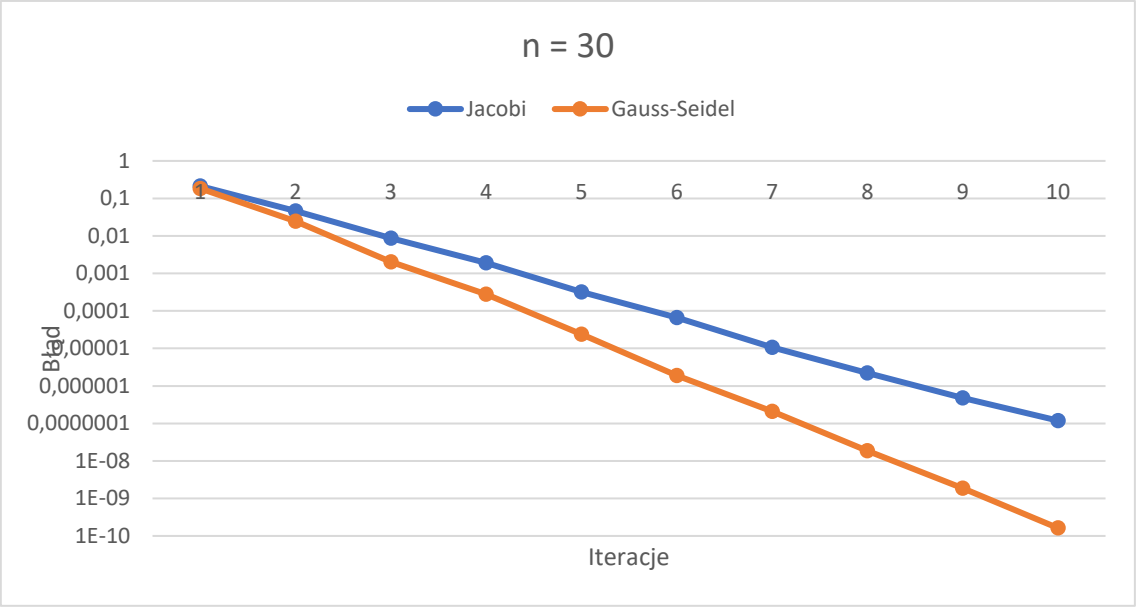
5.3. Macierz 5x5



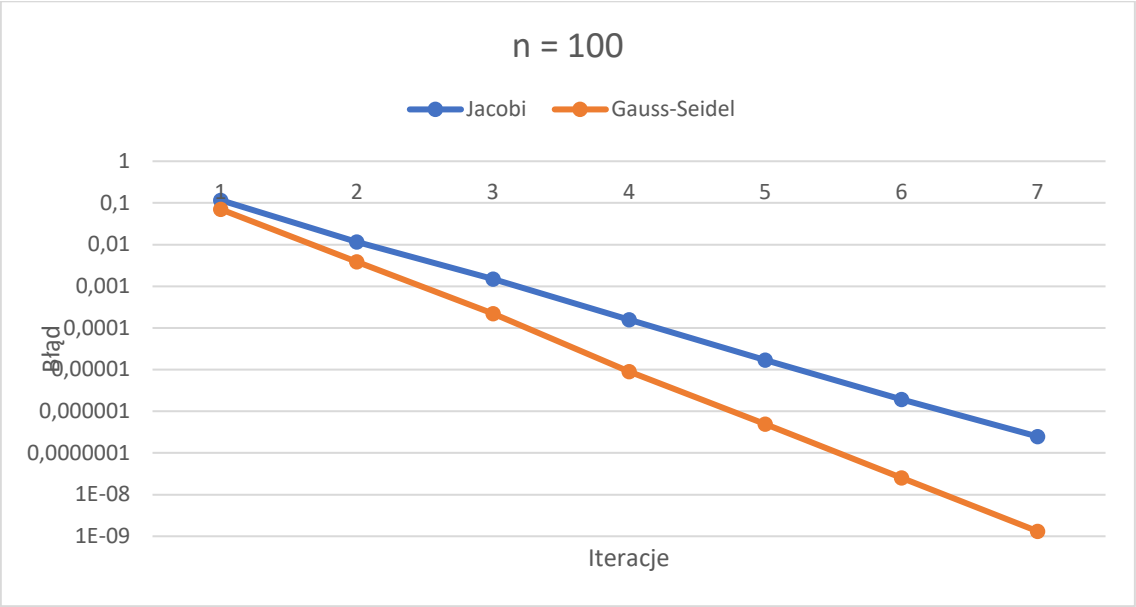
5.4. Macierz 10x10



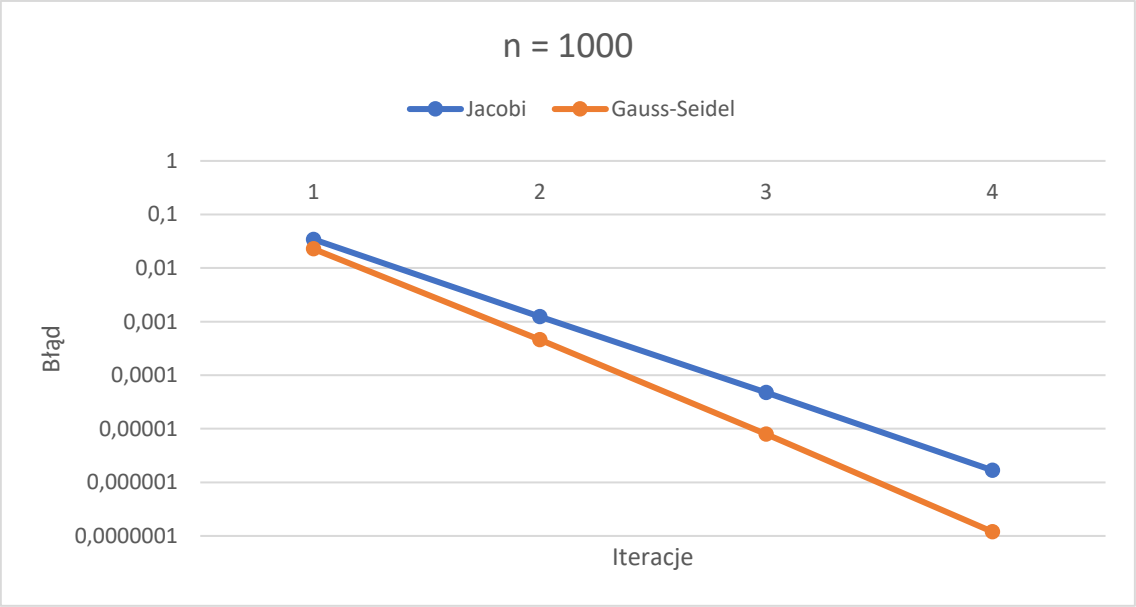
5.5. Macierz 30x30



5.6. Macierz 100x100



5.7. Macierz 1000x1000



6. Wnioski

Jak widzimy na powyższych wykresach metoda Gaussa-Seidla zbiega znacząco szybciej niż metoda Jacobiego.

Warto również zauważyć, że im większy rozmiar macierzy tym mniej iteracji jest potrzebnych do osiągnięcia oczekiwanej wartości. Dlatego też metody iteracyjne stosuje się dla bardzo dużych układów równań, a nie stosuje się ich dla małych macierzy gdyż korzystniej jest od razu obliczyć dokładny wynik używając metod z poprzednich laboratoriów.