Zadanie numeryczne 05

Autor: Eryk Stępień

30.12.2023

Spis treści:

# 1. Problem

Rozwiąż układ równań

$$\begin{pmatrix} 3 & 1 & 0.15 & & & & \\ 1 & 3 & 1 & 0.15 & & & \\ 0.15 & 1 & 3 & 1 & 0.15 & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \\ & & & 0.15 & 1 & 3 & 1 \\ & & & & 0.15 & 1 & 3 \end{pmatrix} x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \dots \\ N-1 \\ N \end{pmatrix}$$

Dla N = 124 za pomocą metod Jacobiego i Gaussa-Seidela. Przedstaw graficznie różnicę pomiędzy dokładnym rozwiązaniem a jego przybliżeniami w kolejnych iteracjach wybierając kilka zestawów punktów startowych. Na tej podstawie porównaj dwie metody

# 2. Program

## 2.1 Użyte narzędzia

Program został napisany w języku Python 3.10. Przy zastosowaniu środowiska PyCharm 2023.2.2. Korzysta on z następujących bibliotek:

- Numpy
- Matplotlib.pyplot

## 2.2 Kompilacja i uruchomienie

W celu kompilacji należy wywołać poniższą komendę w terminalu:

*python NUM5.py*

## 2.3 Opis działania programu

Rozwiązanie układu równań odbywa się z wykorzystaniem iteracyjnych metod Jacobiego i Gaussa-Seidela z wykorzystaniem poniższych formuł:

Dla Jacobiego:

$$x_i^{k+1} = \frac{b_i - \sum_{i \neq j} a_{ij} x_j^k}{a_{ii}}$$

Dla Gaussa-Seidela:

$$x_i^{k+1} = \frac{b_i - \sum_{i<j} a_{ij} x_j^k - \sum_{i>j} a_{ij} x_j^{k+1}}{a_{ii}}$$

Dzięki specyfice macierzy obliczenie jednej iteracji wykonujemy w złożoności O(n)

Przy każdej iteracji program oblicza błąd $\left\| x^d - x^{k+1} \right\|, gdzie\ x^d$ jest dokładnym rozwiązaniem równania obliczonym za pomocą biblioteki *numpy*. Program wykorzystuje wartości błędu do sporządzania wykresu błędu obu metod w zależności od liczby iteracji.

Program przerywa obliczanie kolejnych iteracji po osiągnięciu wartości błędu mniejszej, niż podana tolerancja lub po przekroczeniu podanej maksymalnej liczby iteracji.

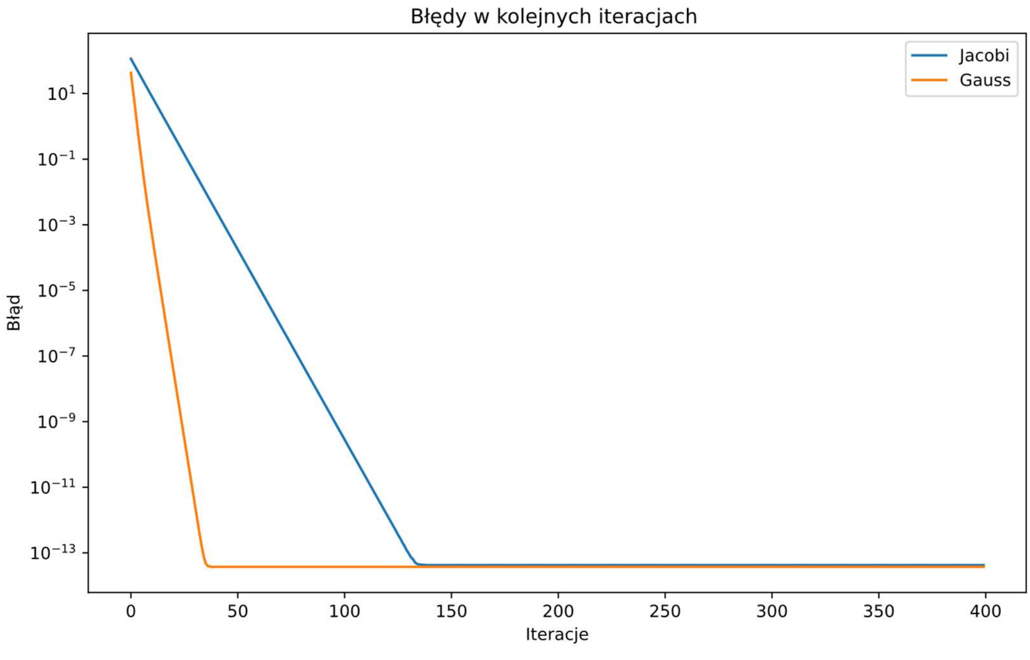
# 3. Analiza wyników działania programu

Program w pierwszej kolejności oblicza rozwiązanie układu równań oraz wyświetla wyniki dla poszczególnych metod. Dla N = 124 i punktu startowego będącego wektorem x wypełnionym zerami mamy:

$$Jacobi = \begin{pmatrix} 0.17801523 \\ 0.38116168 \\ 0.56528409 \\ \dots \\ 21.0615641 \\ 33.1511687 \end{pmatrix} Gauss - Seidel \begin{pmatrix} 0.17801523 \\ 0.38116168 \\ 0.56528409 \\ \dots \\ 21.0615641 \\ 33.1511687 \end{pmatrix} Biblioteka = \begin{pmatrix} 0.17801523 \\ 0.38116168 \\ 0.56528409 \\ \dots \\ 21.0615641 \\ 33.1511687 \end{pmatrix}$$

Rozwiązanie za pomocą metod iteracyjnych jest, więc równe temu otrzymanego za pomocą biblioteki *numpy*.

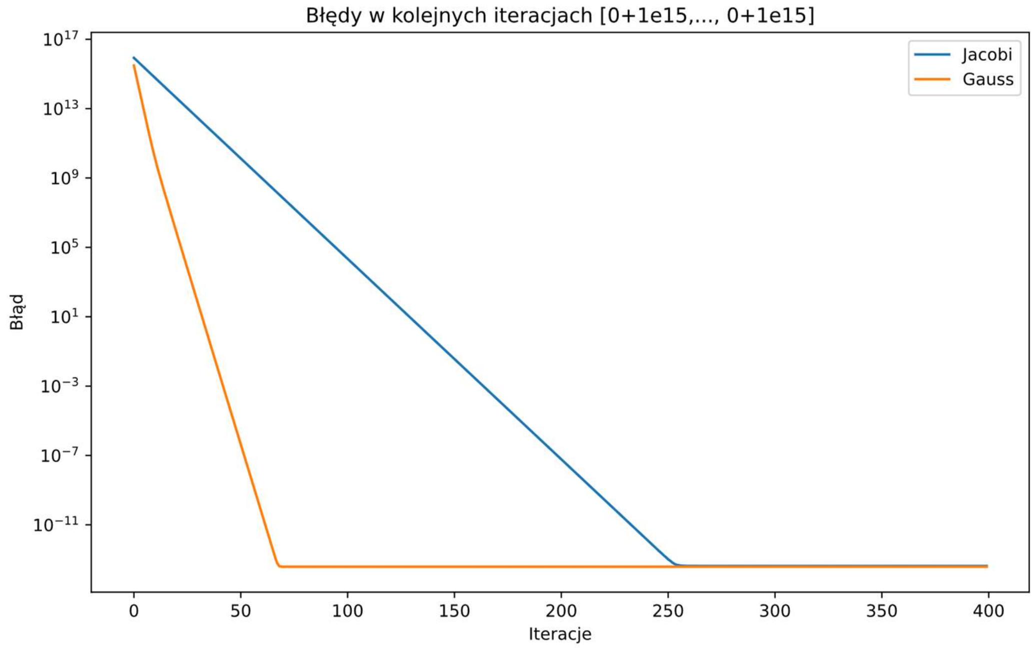
Program tworzy również wykres zależności błędu od iteracji w skali logarytmicznej.

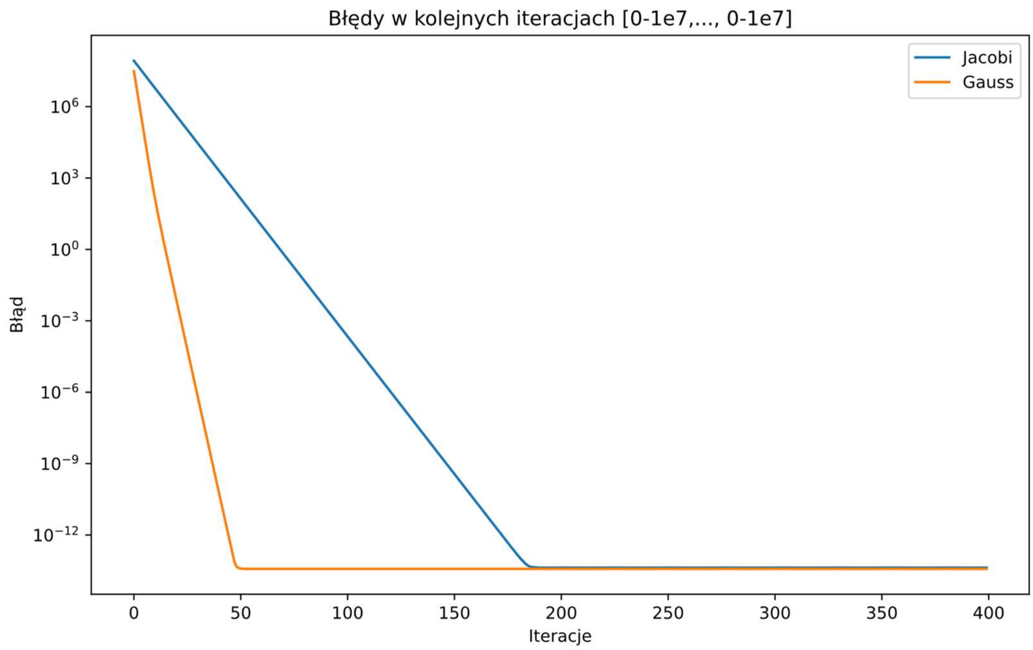


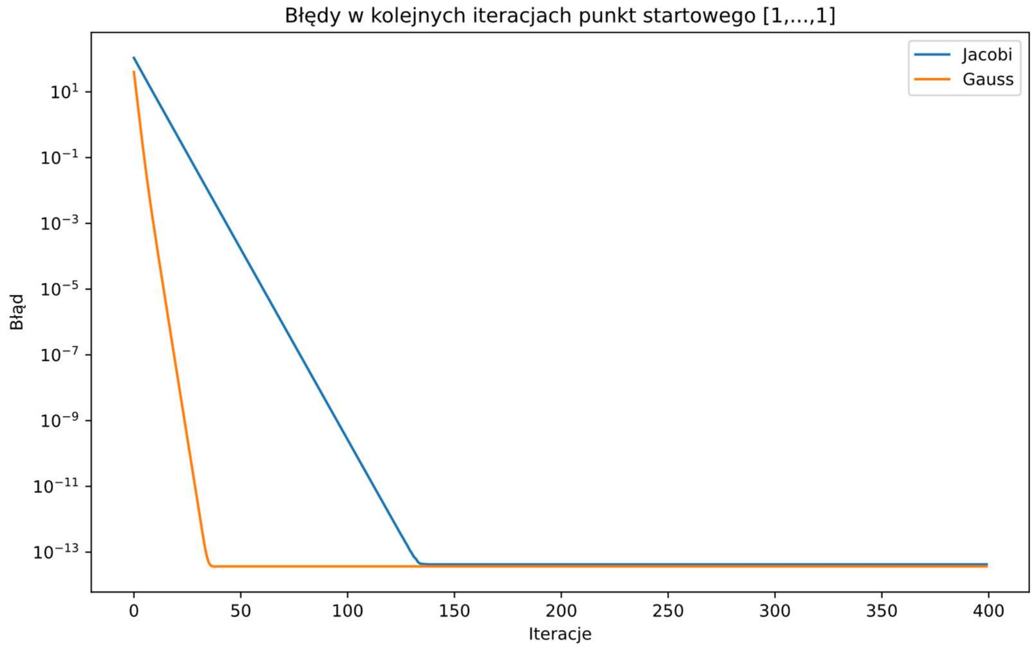

Metoda iteracyjna Gaussa-Seidela jest w naszym przypadku szybsza niż metoda Jacobiego i zbieżność osiągamy już przed 50 iteracją natomiast w przypadku Jacobiego zbieżność osiągamy w okolicy 125 iteracji.

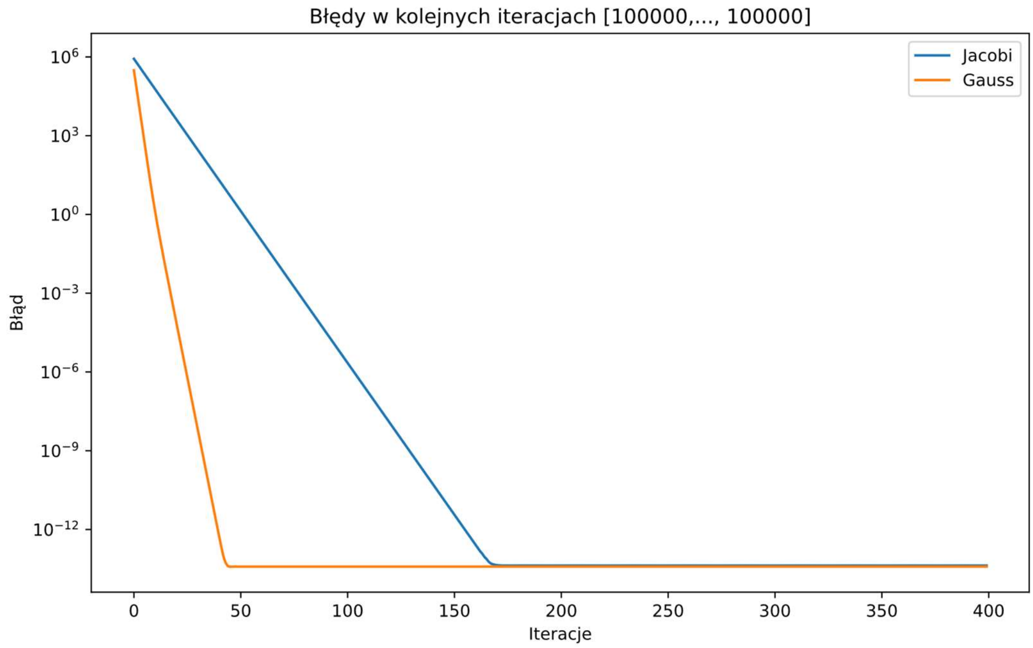
## 3.1 Testowanie różnych punktów startowych

Sprawdźmy jak będzie zachowywała się zbieżność przy różnych wartościach startowych wektora x






Metoda Jacobiego jest bardziej czuła na zmiany punktów startowych i odpowiednie ich dobranie jest w stanie znacząco przyśpieszyć zbieżność. W przypadku Gaussa-Seidela również występuje zmiana ilości iteracji w zależności od wybranych punktów startowych ale nie jest ona taka znacząca jak w przypadku Jacobiego