Savchenko Yehor, Metody Numeryczne, zadanie NUM 5

Instrukcja do programy:

1. Zainstalować python: https://www.python.org/downloads/
2. Zainstalować biblioteki NumPy: **pip install numpy**
3. Wpisać do terminalu: **python "FileName"**

Wstęp:

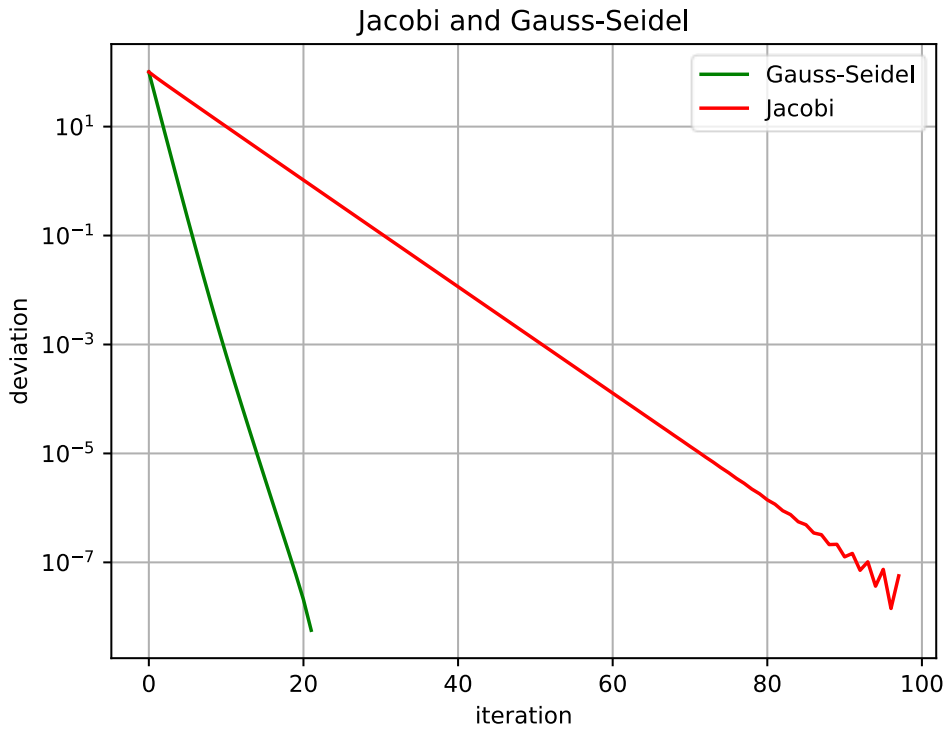
Musiałem rozwiązać układ równań:

$$\begin{pmatrix} 3 & 1 & 0.2 & & & & \\ 1 & 3 & 1 & 0.2 & & & \\ 0.2 & 1 & 3 & 1 & 0.2 & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & 0.2 & 1 & 3 & 1 \\ & & & & 0.2 & 1 & 3 \end{pmatrix} x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \dots \\ N-1 \\ N \end{pmatrix}$$

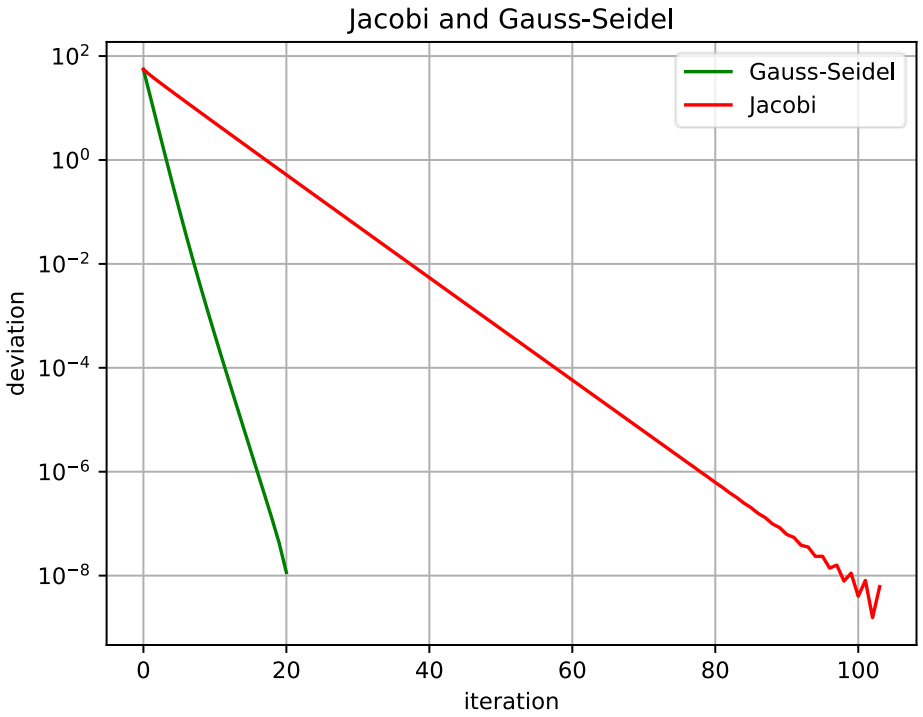
dla **N = 100** za pomocą metod Jacobiego i Gaussa-Seidela. Także musiałem przedstawić różnicę pomiędzy dokładnym rozwiązaniem a jego przybliżeniami w kolejnych iteracjach wybierając kilka zestawów punktów startowych.
Zauważyłem, że podana macierz jest silnie diagonalna i struktura macierzy daje możliwość nie przechowywać całą macierz, a tylko elementy różne od zera. Jako zestawy punktów startowych wziąłem wektory, gdzie wszystkie elementy dla **x1 = 1, x2 = 10, x3 = 100**.
W końcu wyniki które otrzymałem za pomogą iteracyjnych metod porównałem z wynikiem z rozwiązania za pomocą algebraicznych bibliotek.

Wyniki:

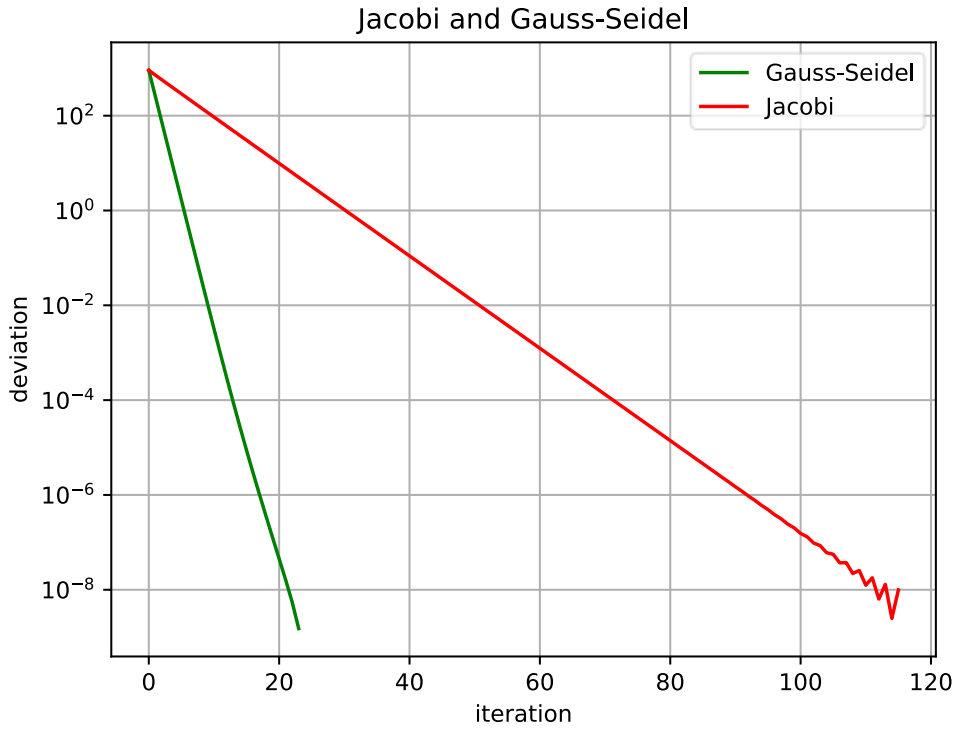
Dla x1:



Dla x2:


Jacobi and Gauss-Seidel
Gauss-Seidel
Jacobi
deviation
iteration
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
0
20
40
60
80
100

Dla x3:


Jacobi and Gauss-Seidel
Gauss-Seidel
Jacobi
deviation
iteration
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
0
20
40
60
80
100
120

Wektor x = (0.17126009, 0.37523974, 0.55489993, 0.74060385, 0.9260231, 1.11108743**,** 1.29629727, 1.48148292, 1.66666609, 1.85185195, 2.03703705, 2.22222221, 2.40740741, 2.59259259, 2.77777778, 2.96296296, 3.14814815, 3.33333333, 3.51851852, 3.7037037, 3.88888889, 4.07407407, 4.25925926, 4.44444444, 4.62962963, 4.81481481, 5.0000000, 5.18518519, 5.37037037, 5.55555556, 5.74074074, 5.92592593, 6.11111111, 6.2962963, 6.48148148, 6.66666667, 6.85185185, 7.03703704, 7.22222222, 7.40740741, 7.59259259, 7.77777778, 7.96296296, 8.14814815, 8.33333333, 8.51851852, 8.7037037, 8.88888889, 9.07407407, 9.25925926, 9.44444444, 9.62962963, 9.81481481, 10.000000, 10.18518519, 10.37037037, 10.55555556, 10.74074074, 10.92592593, 11.11111111, 11.2962963, 11.48148148, 11.66666667, 11.85185185, 12.03703704, 12.22222222, 12.40740741, 12.59259259, 12.77777778, 12.96296296, 13.14814815, 13.33333333, 13.51851852, 13.7037037, 13.88888889, 14.07407407, 14.25925926, 14.44444444, 14.62962963, 14.81481481, 15.0000000, 15.18518518, 15.37037037, 15.55555556, 15.74074072, 15.92592603, 16.11111094, 16.29629569, 16.48148656, 16.66665111, 16.85185669, 17.03722139, 17.22130055, 17.40924191, 17.59631865, 17.73605398, 18.1074402, 18.03115407, 16.95603806, 26.47924371**)**

Przedyskutowanie wyników:

Dla otrzymania takich wyników rozwiązywałem to dwoma metodami: metoda Jacobiego i metoda Gaussa-Seidela. Metoda Jacobiego jest zbieżna, bo macierz jest silnie diagonalnie dominująca. Metoda Gaussa-Seidela jest zbieżna, ponieważ macierz A jest symetryczna i dodatnio określona.

Dla metod iteracyjnych macierz można przedstawić w postaci:

$$A = A_1 + A_2$$
$$Ax = b$$
$$(A_1 + A_2)x = b \qquad (1)$$

Metoda Jacobiego będzie miała taki rozkład macierzy:

$$A = L + D + U = D + L + U$$

gdzie L – pod diagonalna macierz, D – diagonalna macierz, U – nad diagonalna macierz. Dla wzoru (1) będziemy mieć, że **$D = A_1$, $L+U = A_2$**.
Stąd mamy taki wzór na wektor **x**:

$$x_i^{(n+1)} = \frac{b_i - \sum_{k<i} a_{ik}\, x_k^{(n)} - \sum_{k>i} a_{ik} x_k^{(n)}}{a_{ii}}$$

Jak rozpiszemy ten wzór, to będzie widoczne przez to, gdzie możemy użyć strukturę macierzy. Że suma $\sum_{k<i} a_{ik}\, x_k^{(n)}$ będzie używała pod diagonalne elementy, a suma $\sum_{k>i} a_{ik} x_k^n$ będzie używała nad diagonalne elementy.

Metoda Gaussa-Seidela będzie miała taki rozkład macierzy:

$$A = L + D + U$$

gdzie L – pod diagonalna macierz, D – diagonalna macierz, U – nad diagonalna macierz. Dla wzoru (1) będziemy mieć, że **$L+D = A_1$, $U = A_2$**.
Stąd mamy taki wzór na wektor **x**:

$$x_i^{(n+1)} = \frac{b_i - \sum_{k<i} a_{ik}\, x_k^{(n+1)} - \sum_{k>i} a_{ik} x_k^{(n)}}{a_{ii}}$$

Licząc przez te metody wektor **x** używałem **precyzję**, która u mnie jest równa **10^ (-10)** co liczy dokładny wynik wektora **x**. Próbowałem także policzyć dla większej precyzję i wychodził już wynik przybliżony do dokładnego.

Warunek stopu dla metod miałem taki: $x^{(n+1)} - x^{(n)} < precyzja$

Z wykresów widać, że metoda Gaussa-Seidela zbiega szybciej niż metoda Jacobiego.