SPRAWOZDANIE
Z PIĄTEGO ZADANIA NUMERYCZNEGO

MACIEJ WÓJCIK

Spis treści:

1. Polecenie zadania

6. **(zadanie numeryczne NUM5)** Rozwiąż układ równań

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

dla $N = 100$ za pomocą metod Jacobiego i Gaussa-Seidela. Przedstaw graficznie różnicę pomiędzy pomiędzy dokładnym rozwiązaniem a jego przybliżeniami w kolejnych iteracjach wybierając kilka zestawów punktów startowych. Na tej podstawie porównaj dwie metody.

Zadanie to sprowadza się do rozwiązania równania **Ax = b**, gdzie za **A** przyjmujemy macierz wstęgową, a z **b** wektor składający się z rosnących liczb.

2. Instrukcja uruchomienia

Aby uruchomić program należy skorzystać z polecenia:

make run

Lub

python3 NUM5.py

3. Sposób obliczenia

Do rozwiązania naszego równania posłużymy się metodami Gaussa-Seidela i Jacobiego.

Obydwie te metody są metodami iteracyjnymi. Oznacza to, że nie wyznaczają one dokładnej wartości, a dokonują przybliżenia z każdą iteracją. Im więcej iteracji wykonamy, tym dokładniejsze będzie przybliżenie.

Dokonajmy kilku przekształceń.

$$A\,x = b$$

Macierz A możemy rozłożyć na sumę macierzy A_1 i A_2:

$$(A_1 + A_2)x = b$$
$$A_1 x = -A_2 x + b$$

Następnie zapisujemy równanie iteracyjne:

$$A_1 x^{n+1} = -A_2 x^n + b$$

Macierz A rozpisujemy na $D + L + U$, gdzie:

D - część diagonalna
L - macierz wypełniona tylko poniżej diagonali
U - macierz wypełniona tylko powyżej diagonali

Teraz w zależności od wymaganej metody przyjmiemy inne oznaczenia:

a) Metoda Jacobiego

$A_1 = D$ i $A_2 = L + U$ i podstawiając do powyższego równania otrzymujemy:

$$Dx^{n+1} = -(L + U)x^n + b$$

A po rozpisaniu na składowe otrzymujemy wzór:

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

A uwzględniając jeszcze fakt, że nasza macierz ma w sobie wiele zer, możemy wzór uprościć do:

$$x_i^{n+1} = \frac{b_i - x_{i-1}^n - 0.2x_{i-2}^n - x_{i+1}^n - 0.2x_{i+2}^n}{3}$$

b) Metoda Gaussa-Seidela

$A_1 = L + D$ i $A_2 = U$ i podstawiając do wcześniejszego równania otrzymujemy:

$$(L + D)x^{n+1} = - Ux^n + b$$

A po rozpisaniu na składowe otrzymujemy wzór:

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

Tak samo jak w przypadku poprzedniej metody, bierzemy pod uwagę charakterystykę budowy macierzy A i otrzymujemy uproszczony wzór:

$$x_i^{n+1} = \frac{b_i - x_{i-1}^{n+1} - 0.2x_{i-2}^{n+1} - x_{i+1}^n - 0.2x_{i+2}^n}{3}$$

4. Zakończenie obliczania

Założeniem metod iteracyjnych jest iterowanie do nieskończoności, dlatego musimy jakoś wybrać moment, w którym zaprzestaniemy iterowanie. Do tego można posłużyć się normą euklidesową. Polega to na tym, że porównujemy normę z aktualnej iteracji i z poprzedniej. Zależy nam na tym, żeby różnica między nimi była mniejsza niż wybrany przez nas wcześniej współczynnik precyzji.

5. Wyniki dla precyzji 10^{-10}

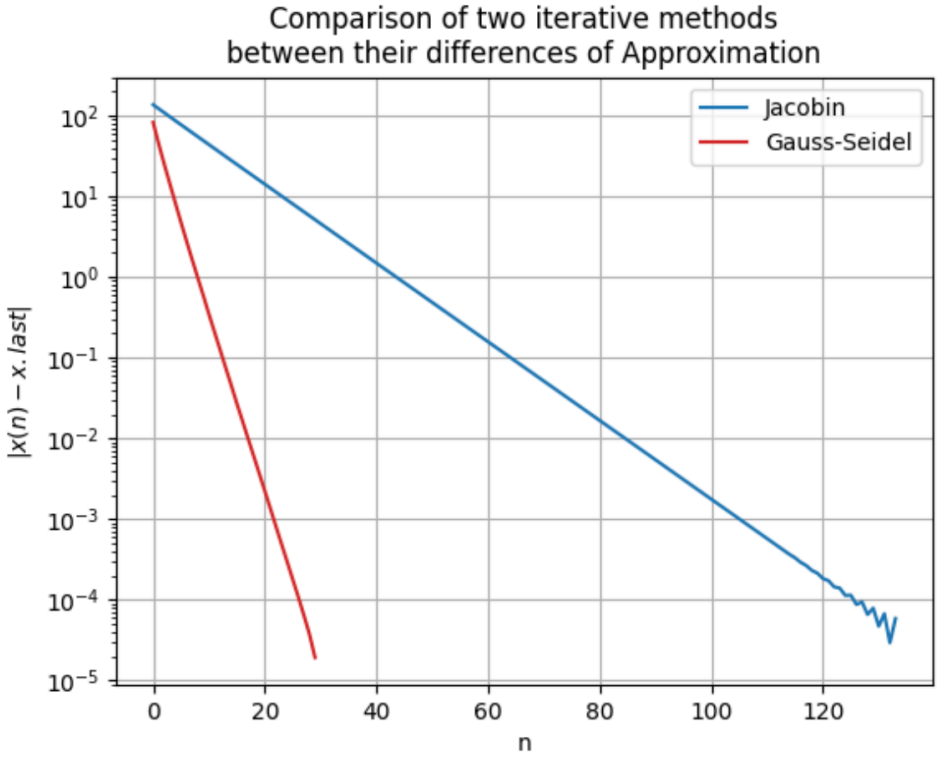
Metoda Jacobiego:

[0.1712600924903588, 0.37523973744936984, 0.5548999253656824,
0.7406038489199288, 0.9260230950909856, 1.111087426354369,
1.2962972717575567, 1.4814829213555454, 1.666666089804832,
1.851851945285181, 2.0370370477279383, 2.2222222114424945,
2.4074074103564507, 2.592592592354478, 2.777777777620944,
2.962962963016558, 3.148148148121301, 3.3333333333181066,
3.5185185185046497, 3.7037037036879137, 3.8888888888731494,
4.07407407405799, 4.259259259242868, 4.444444444427804,
4.629629629612757, 4.814814814797733, 4.999999999982731,
5.185185185167749, 5.370370370352782, 5.555555555537829,
5.740740740722888, 5.925925925907958, 6.1111111110930345,
6.296296296278119, 6.4814814814632085, 6.666666666648301,
6.8518518518333975, 7.037037037018494, 7.222222222203591,
7.407407407388689, 7.592592592573783, 7.777777777758879,
7.962962962943973, 8.148148148129065, 8.333333333314156,
8.518518518499244, 8.703703703684333, 8.888888888869419,
9.074074074054503, 9.25925925923959, 9.444444444424677,
9.629629629609761, 9.81481481479485, 9.999999999979945,
10.18518518516504, 10.370370370350141, 10.555555555535248,
10.740740740720357, 10.925925925905482, 11.111111111090606,
11.296296296275747, 11.481481481460891, 11.666666666646057,
11.851851851831228, 12.037037037016418, 12.222222222201617,
12.407407407386847, 12.592592592572082, 12.777777777757345,
12.962962962942626, 13.14814814812793, 13.333333333313263,
13.518518518498615, 13.703703703683992, 13.888888888869493,
14.074074074054701, 14.25925925924018, 14.444444444428896,
14.629629629599679, 14.814814814811427, 15.000000000071962,
15.185185184561599, 15.370370371988043, 15.555555556010162,
15.740740716808686, 15.925926030877994, 16.11111094103337,
16.296295691220063, 16.481486556830017, 16.666651111154014,
16.85185669431732, 17.037221385426975, 17.22130054522282,
17.409241909482912, 17.596318652558807, 17.736053983215623,
18.107440202724174, 18.03115406571765, 16.956038061790633,
26.479243708353007]

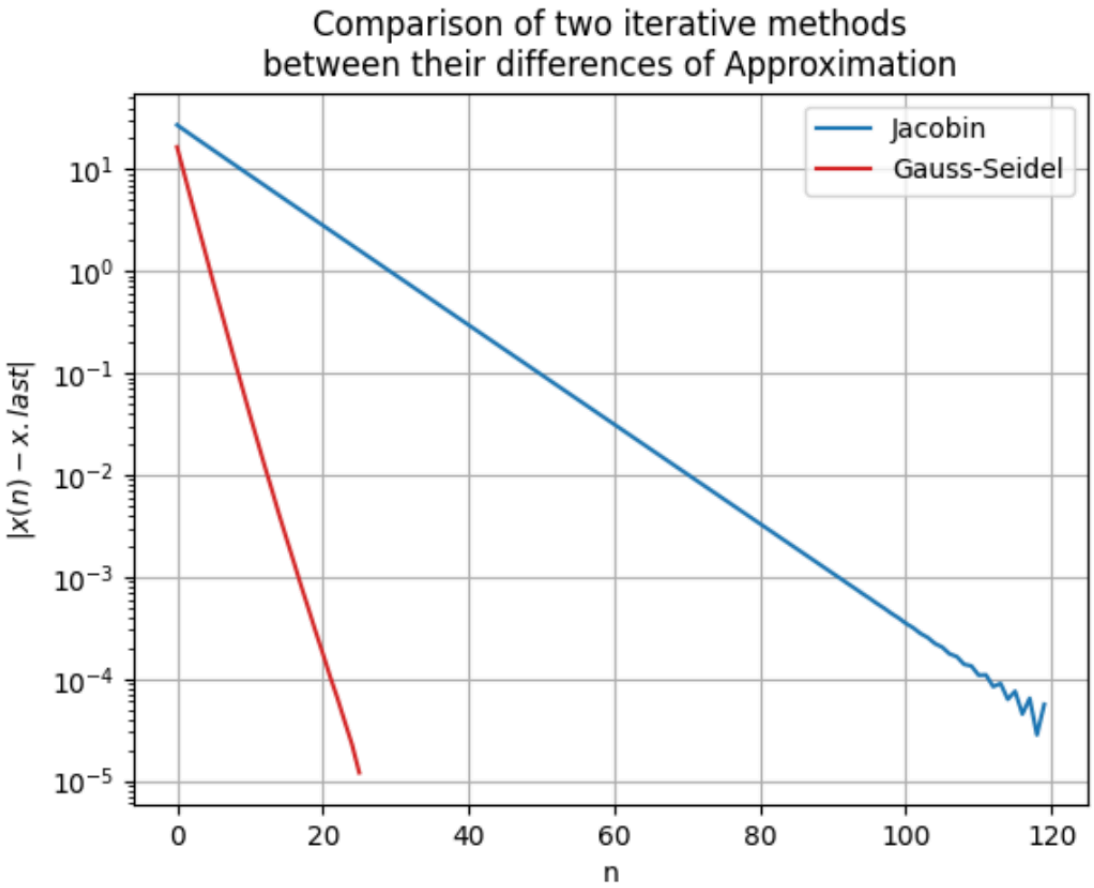
Metoda Gaussa-Seidela:

[0.17126009248976437, 0.37523973745242895, 0.5548999253699801, 0.7406038489214541, 0.9260230950987417, 1.1110874263600972, 1.2962972717629713, 1.4814829213654193, 1.6666660898134407, 1.8518519452929674, 2.0370370477415487, 2.2222222114504557, 2.4074074103701495, 2.592592592366724, 2.7777777776341743, 2.96296296302803, 3.148148148140391, 3.3333333333264332, 3.518518518524608, 3.703703703701311, 3.888888888889051, 4.074074074073708, 4.259259259260909, 4.444444444442662, 4.62962962962902, 4.814814814819204, 4.999999999993368, 5.1851851851903294, 5.370370370369366, 5.55555555555334, 5.740740740743219, 5.925925925924102, 6.111111111113863, 6.2962962962924225, 6.4814814814832395, 6.6666666666696885, 6.851851851846317, 7.037037037039934, 7.222222222224352, 7.407407407402336, 7.592592592596709, 7.777777777776833, 7.962962962961501, 8.148148148149767, 8.333333333332403, 8.51851851851993, 8.703703703701, 8.888888888891266, 9.074074074074533, 9.25925925925594, 9.444444444447756, 9.629629629628704, 9.814814814813959, 10.000000000000073, 10.18518518518791, 10.370370370364453, 10.55555555556316, 10.740740740734688, 10.925925925927343, 11.111111111114178, 11.296296296292672, 11.481481481481723, 11.666666666670283, 11.851851851846474, 12.03703703704201, 12.222222222219118, 12.407407407408158, 12.592592592593142, 12.77777777777768, 12.962962962962061, 13.148148148149113, 13.333333333333146, 13.518518518518066, 13.703703703704484, 13.888888888887585, 14.074074074075961, 14.259259259257298, 14.444444444447386, 14.629629629620716, 14.814814814824318, 15.000000000094476, 15.185185184574609, 15.370370372006219, 15.55555555602561, 15.740740716823282, 15.925926030893068, 16.111110941046686, 16.296295691233222, 16.481486556842167, 16.66665111116534, 16.851856694327818, 17.037221385436514, 17.221300545231436, 17.409241909490536, 17.59631865256542, 17.736053983221197, 18.107440202728675, 18.031154065721093, 16.956038061792967, 26.47924370835427]

„x” z losowymi wartościami:



„x” z zerami:



6. Wnioski

Obydwie metody dają poprawne wyniki. Jednak metoda Gaussa-Seidela wyniki podaje w dużo szybszym czasie. Na wykresie widać, że zbiega ona do pożądanej wartości dużo szybciej. Wynika to z tego, że do obliczeń wykorzystuje najbardziej aktualne wartościm w przeciwieństwie do metody Jacobiego, która bazuje na wektorze z poprzedniej iteracji.