SPRAWOZDANIE NUM5

JAKUB KRĘCISZ

Treść zadania

Rozwiąż układ równań:

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

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.

Omówienie zadania

Na początku, uznajmy, że macierz rzadka, która jedyne niezerowe elementy ma na diagonali to nasze **A**. Jeżeli chodzi o wektor wyrazów wolnych, przyjmijmy, że jest to **b**.

Omówmy teraz nasz problem, z którym musimy się uporać. Mamy rozwiązać układ równań $Ax = b$. Nasze **A** jak w poprzednich przypadkach (mowa tu o poprzednich zadaniach numerycznych), jest macierzą rzadką i na dodatek z niezerowymi wartościami tylko na diagonalach. W zasadzie mamy tu ten sam przypadek co w poprzednich zadaniach. Jednak tym razem musimy zastosować metody Jacobiego i Gaussa-Seidela, opowiem teraz coś więcej o nich.

Jeżeli chodzi o metody o których przed chwilą wspomniałem, są to metody iteracyjne. Jest to znaczna różnica od poprzednich metod, ze względu na to, że poprzednie metody pozwalały nam na dokładne obliczenie rozwiązań (nie mówiąc o błędach zaokrągleń spowodowanych precyzją). W tym jednak przypadku, żeby metoda iteracyjna wykazała nam dokładny wynik, musiałaby się wykonywać w nieskończoność. W tym jednak przypadku, postaramy się zrobić to w skończoną ilość iteracji, aż dojdziemy do satysfakcjonującego nas wyniku. Wyniku który będzie się mieścił w naszych granicach błędu.

Zajmijmy się teraz naszą pierwszą metodą, czyli metodą Jacobiego. Aby ta metoda była zbieżna (co jest potrzebne w metodach iteracyjnych) musi być macierzą przekątniowo dominującą:

$$\forall_i |a_{i,i}| > \sum_{k \neq i} |a_{i,k}|$$

*wartości bezwzględne elementów na głównej przekątnej są większe od sumy wartości bezwzględnych pozostałych elementów w wierszach

W naszym przypadku jest to ewidentnie spełnione, nasza macierz A jest przekątniowo dominująca. Więc ta metoda na pewno będzie zbieżna.

Zabierzmy się teraz za drugą metodę, czyli metodę Gaussa-Seidela. W jej przypadku, żeby ta metoda była zbieżna, macierz musi być symetryczna i dodatnio określona. Oczywiście obie rzeczy występują w naszej macierzy, więc ta metoda również jest zbieżna.

Gdy już wiemy, że nasze obie metody na pewno będą zbieżne, możemy się zająć naszą macierzą A. Na początku rozpiszmy ją jako suma dwóch macierzy A_1 i A_2. Przekształćmy te równanie w równanie iteracyjne:

$$Ax = b \Leftrightarrow (A_1 + A_2)x = b \Leftrightarrow A_1 x = -A_2 x + b \Leftrightarrow A_1 x^{n+1} = -A_2 x^n + b$$

Przyjmujemy, że naszą macierz A możemy rozłożyć na macierze L, D, U które są odpowiednio:

- L – macierz poddiagonalna
- D – macierz diagonalna
- U – macierz nad diagonalna

Wiemy, że dla metody Jacobiego mamy A = D + (L + U), w takim razie poprzednie równanie możemy rozpisać jako:

$$A_1 x^{n+1} = -A_2 x^n + b \Leftrightarrow D x^{n+1} = -(L + U)x^n + b$$

Po rozpisaniu na składowe mamy następujący wzór:

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

Zróbmy to samo w przypadku drugiej metody, metody Gaussa-Seidela. Wiemy, że dla niej mamy: A = (L+D) + U, rozpiszmy to tak jak w poprzedniej metodzie:

$$A_1 x^{n+1} = -A_2 x^n + b \Leftrightarrow (L + D)x^{n+1} = -Ux^n + b$$

Rozpiszmy wzór na składowe:

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

No dobra, mamy już wzory, możemy jeszcze uwzględnić naszą macierz A do wzorów. Wtedy dużo obliczeń nam się uprości poprzez mnożenie przez 0. Nasze wzory ostatecznie będą miały postać:

Jacobi

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

Gauss-Seidel

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

Jak dobrze wiemy, metody iteracyjne w zasadzie idą w nieskończoność. Dlatego też, musimy ustalić jakieś limity i warunki, kiedy program musi się zakończyć. Do zbieżności naszych metod (w wyniku satysfakcjonującym, mieszczącym się w naszym błędzie do określonej pozycji) użyjemy norm euklidesowych, będziemy porównywać normę z aktualnej iteracji z tą z poprzedniej, jeśli różnica będzie mniejsza niż nasza ustalona precyzja, wtedy program zakończy działanie i wypisze rezultaty.

Uruchomienie programu

Do uruchomienia programu wykorzystamy Makefile:

Aby uruchomić nasz program, wystarczy użyć polecenia:

```
make run
```

Wyniki

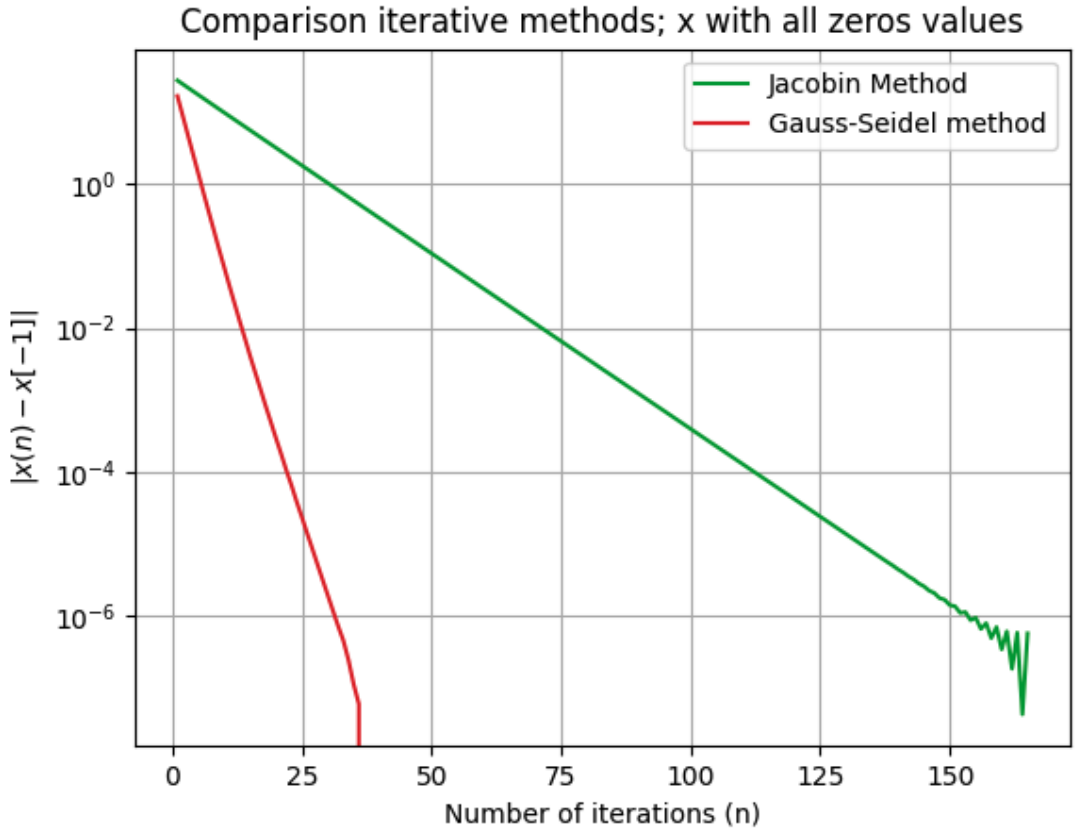
1. Rozwiązania naszego układu równań, dla naszych dwóch metod dla precyzji rzędu 10^{-16} wyglądają następująco:

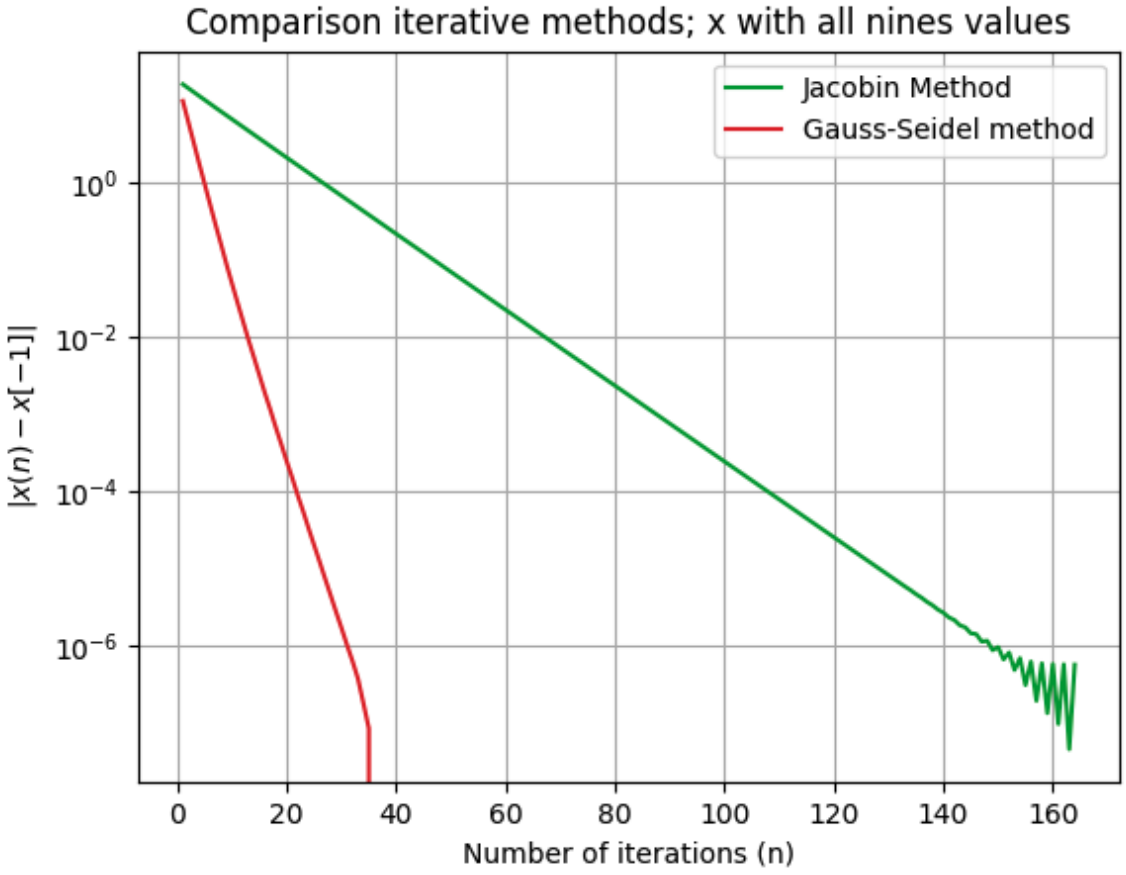
Metoda Jacobiego:

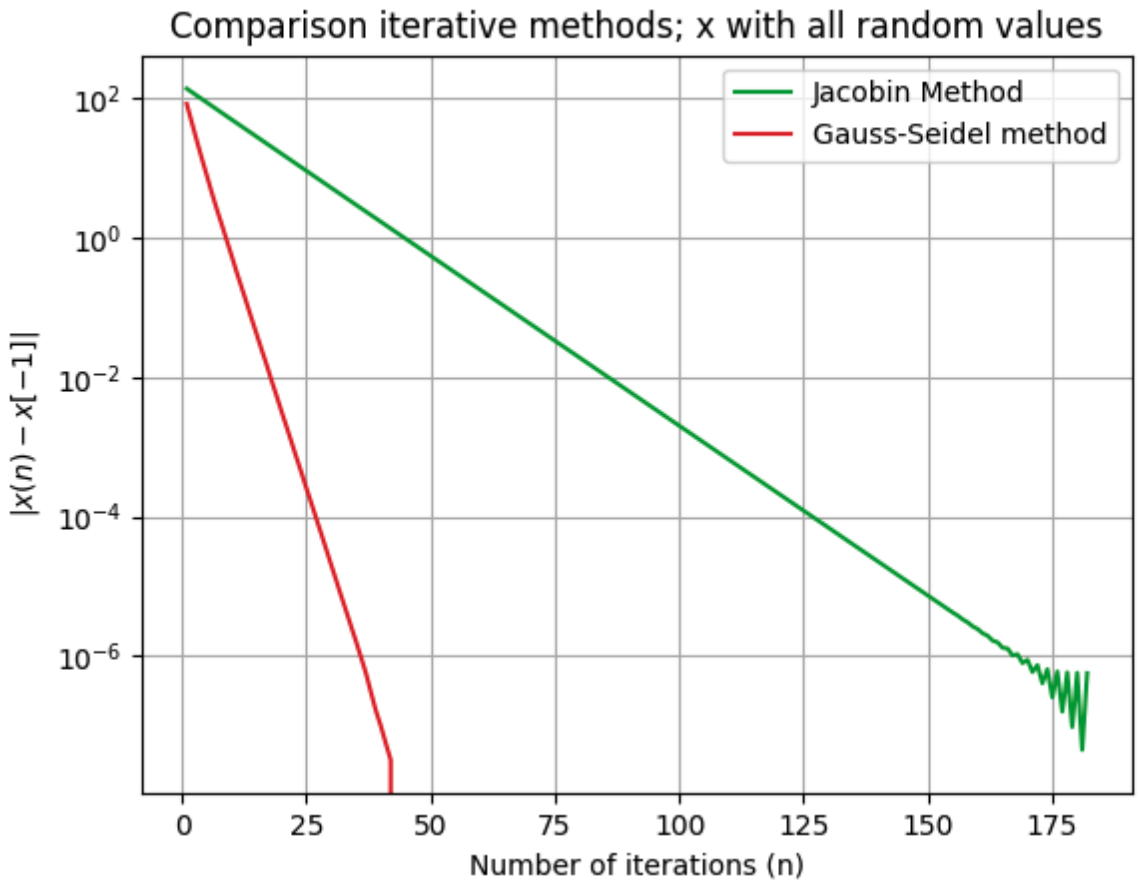
[0.1712600924915493, 0.3752397374515706, 0.5548999253689074, 0.7406038489241576,
0.9260230950961977, 1.1110874263605368, 1.2962972717646517, 1.4814829213635328,
1.6666660898136743, 1.8518519452948397, 2.0370370477383695, 2.2222222114536567,
2.407407410368299, 2.5925925923669664, 2.7777777776340287, 2.962962963030195,
3.1481481481354465, 3.3333333333327193, 3.518518518519688, 3.7037037037033413,
3.8888888888889284, 4.074074074074087, 4.2592592592592515, 4.4444444444444455,
4.629629629629629, 4.814814814814814, 4.999999999999998, 5.185185185185186,
5.370370370370369, 5.555555555555554, 5.7407407407407405, 5.925925925925925,
6.11111111111111, 6.296296296296295, 6.481481481481481, 6.666666666666664,
6.85185185185185, 7.037037037037035, 7.222222222222221, 7.407407407407406,
7.592592592592594, 7.777777777777776, 7.962962962962961, 8.148148148148147,
8.33333333333333, 8.518518518518514, 8.703703703703702, 8.888888888888888,
9.074074074074073, 9.259259259259256, 9.444444444444443, 9.629629629629626,
9.814814814814811, 10.0, 10.185185185185185, 10.370370370370368, 10.555555555555554,
10.740740740740739, 10.925925925925924, 11.111111111111109, 11.296296296296292,
11.48148148148148, 11.666666666666664, 11.85185185185185, 12.037037037037033,
12.22222222222222, 12.407407407407405, 12.592592592592588, 12.777777777777773,
12.962962962962957, 13.148148148148145, 13.33333333333333, 13.518518518518514,
13.703703703703688, 13.888888888888959, 14.074074074073906, 14.259259259259084,
14.444444444447472, 14.62962962961788, 14.81481481482921, 15.000000000089296,
15.185185184578444, 15.370370372004347, 15.555555556025894, 15.740740716823792,
15.925926030892441, 16.111110941047105, 16.296295691233045, 16.481486556842206,
16.66665111116536, 16.85185669432779, 17.037221385436528, 17.221300545231433,
17.409241909490532, 17.59631865256542, 17.736053983221186, 18.10744020272867,
18.031154065721083, 16.956038061792963, 26.479243708354264]

Metoda Gaussa-Seidela:

[0.1712600924915493, 0.3752397374515706, 0.5548999253689074, 0.7406038489241576,
0.9260230950961977, 1.1110874263605375, 1.2962972717646517, 1.481482921363533,
1.6666660898136743, 1.8518519452948397, 2.0370370477383695, 2.2222222114536567,
2.4074074103682994, 2.5925925923669664, 2.77777777763403, 2.9629629630301952,
3.1481481481354474, 3.3333333333327193, 3.518518518519688, 3.7037037037033413,
3.8888888888889284, 4.074074074074087, 4.2592592592592515, 4.444444444444446,
4.62962962962963, 4.814814814814814, 4.999999999999999, 5.185185185185186,
5.37037037037037, 5.5555555555555545, 5.740740740740743, 5.925925925925925,
6.111111111111111, 6.296296296296298, 6.481481481481481, 6.666666666666667,
6.85185185185185, 7.037037037037038, 7.222222222222222, 7.407407407407406,
7.592592592592594, 7.777777777777778, 7.962962962962962, 8.14814814814815,
8.33333333333333, 8.518518518518519, 8.703703703703704, 8.888888888888891,
9.074074074073, 9.25925925925926, 9.444444444444445, 9.629629629629628,
9.814814814814817, 10.000000000000002, 10.185185185185187, 10.370370370370368,
10.555555555555557, 10.74074074074074, 10.925925925925924, 11.111111111111112,
11.296296296296296, 11.481481481481481, 11.666666666666666, 11.851851851851853,
12.037037037037036, 12.22222222222222, 12.40740740740741, 12.59259259259259,
12.77777777777778, 12.962962962962964, 13.148148148148147, 13.333333333333334,
13.518518518518519, 13.70370370370369, 13.888888888888962, 14.074074074073906,
14.259259259259084, 14.444444444447472, 14.629629629617881, 14.814814814829212,
15.000000000089296, 15.185185184578444, 15.370370372004347, 15.555555556025894,
15.740740716823792, 15.925926030892441, 16.111110941047105, 16.296295691233045,
16.481486556842206, 16.66665111116536, 16.851856694327793, 17.037221385436535,
17.221300545231433, 17.409241909490536, 17.59631865256542, 17.736053983221197,
18.107440202728675, 18.031154065721093, 16.956038061792967, 26.47924370835427]


Comparison iterative methods; x with all zeros values
Jacobin Method
Gauss-Seidel method
10^{0}
10^{-2}
10^{-4}
10^{-6}
$|x(n) - x[-1]|$
0
25
50
75
100
125
150
Number of iterations (n)


Comparison iterative methods; x with all nines values
Jacobin Method
Gauss-Seidel method
10^{0}
10^{-2}
10^{-4}
10^{-6}
$|x(n) - x[-1]|$
0
20
40
60
80
100
120
140
160
Number of iterations (n)



Wnioski

Jeżeli chodzi o wyniki, obie metody przy określonej pozycji, dają prawie dokładnie te same wyniki. Prawdopodobnie różnica w wynikach jest spowodowana zaokrąglaniem do określonej precyzji.

Jeżeli chodzi jednak o zbieżność obu metod. Jak widać na załączonych wykresach, metoda Gaussa-Seidela zbiega znacznie szybciej niż metoda Jacobiego. Spowodowane to jest samym działaniem metody. W metodzie Jacobiego, w każdej iteracji, bazujemy na wektorze ze stanu poprzedniej iteracji. Niestety takie rozwiązanie wiąże się z większą ilością iteracji, a co za tym idzie dłuższym czasem działania programu. Natomiast w metodzie Gaussa-Seidela do obliczeń używamy najbardziej aktualnych wartości.

Jeżeli chodzi o wektor początkowy x, nie mają znaczenia jakie wartości on ma, wynik zawsze będzie taki sam.