Grzegorz Mikołajczyk

Zadanie numeryczne 5

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.

Wprowadzenie

Celem zadania jest napisanie programu wyliczającego przybliżenie rozwiązania równania za pomocą metod iteracyjnych Jacobiego i Gaussa-Seidela oraz graficzne przedstawienie zmian w przybliżeniach w kolejnych iteracjach.

Wynik

Problem rozwiązuje program *program.py*, wykorzystujący biblioteki *numpy* oraz *matplotlib*.

Do obliczenia wyniku wykorzystuje metody iteracyjne Jacobiego i Gaussa-Seidela w postaci następujących wzorów:

Metoda Jacobiego:

$$\overrightarrow{x_i}^{(n+1)} = \frac{\vec{b}_i - \sum_{k<i} a_{ik}\vec{x}_k^{(n)} - \sum_{k>\mathrm{i}} a_{ik}\vec{x}_k^{(n)}}{a_{ii}}$$

Metoda Gaussa:

$$\overrightarrow{x_i}^{(n+1)} = \frac{\vec{b}_i - \sum_{k<i} a_{ik}\vec{x}_k^{(n+1)} - \sum_{k>\mathrm{i}} a_{ik}\vec{x}_k^{(n)}}{a_{ii}}$$

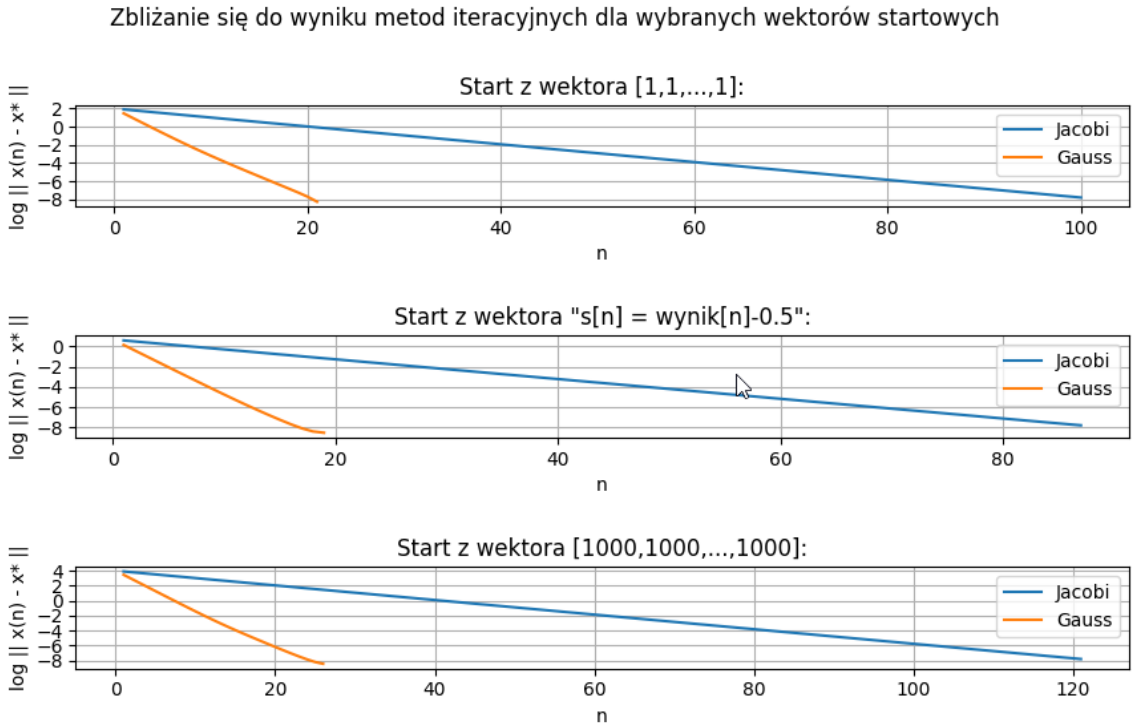
Program najpierw oblicza wynik z dokładnością 0.00000001 za pomocą metody Gaussa, gdzie warunkiem zbieżności metody jest fakt, że różnica pomiędzy normą wektora z poprzedniej i aktualnej iteracji jest mniejsza niż 0.00000001.

Przy obliczaniu wyniku, program wykorzystuje wstęgową strukturę macierzy A, której nawet nie musi trzymać w pamięci, jako że jest wartość w dowolnym punkcie można bez problemu zwrócić z funkcji.

Także we wzorach iteracyjnych, zamiast pełnych sum po wszystkich elementach nad/pod diagonalą, program wykonuje tylko te operacje, dla których wie, że elementy macierzy A będą niezerowe.

Następnie program wykonuje obie metody iteracyjne dla wektorów startowych [1,1,…,1], [1000,1000,…,1000] oraz wektora, którego każda wartość jest równa odpowiadającej jej wartości wektora rozwiązania – 0.5. W trakcie wykonywania tych metod program oblicza normę dla różnicy wektora danej iteracji od wektora wyniku i zapisuje ją w liście.

Na koniec program wyświetla wygenerowane wykresy oraz wypisuje w konsoli obliczony wektor wynikowy.



Dyskusja wyników

W celu weryfikacji wyników napisałem dodatkowy program *test.py*, który po wygenerowaniu wejściowej macierzy A, wykorzystuje funkcję biblioteczną *numpy.linalg.solve* do wyliczenia wektora $\vec{y}$.

```
$ py program.py
result converged after 40 iterations:
 [ 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.          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.
 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.         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]
```

```
$ py test.py
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.          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.
 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.         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]
```

Jak widać, wynik zgadza się z wynikiem otrzymanym metodą iteracyjną

Na otrzymanych wykresach widać również, że niezależnie od wybranego wektora startowego, po odpowiedniej ilości iteracji otrzymany wynik jest bardzo precyzyjny. Metoda Gaussa-Seidela otrzymuje precyzyjny wynik w o wiele mniejszej liczbie iteracji.