Metody numeryczne - projekt 2 - Układy równań liniowych

Jerzy Szyjut, numer albumu: 193064

20.04.2024r.

1 Wstęp

1.1 Układy równań liniowych

Układ równań liniowych to zbiór równań postaci:

$$Ax = b \tag{1}$$

Gdzie dane są macierz kwadratowa A i wektor b, a szukany jest wektor x. Powyższe równanie macierzowe można zapisać w postaci równań:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

1.2 Metody bezpośrednie rozwiązywania układów równań liniowych

Najprostszym sposobem rozwiązania układu równań liniowych jest metoda eliminacji Gaussa. Metoda ta polega na przekształceniu macierzy A do postaci macierzy trójkątnej górnej, a następnie rozwiązaniu układu równań za pomocą algorytmu podstawiania wstecz. Metoda ta ma złożoność obliczeniową $O(n^3) + O(mn^2)$ [1].

Kolejną metodą rozwiązywania układów równań liniowych jest faktoryzacja LU. Metoda ta polega na rozkładzie macierzy A na iloczyn dwóch macierzy L i U, gdzie L to macierz trójkątna dolna, a U to macierz trójkątna górna, tak że $LUx = b$. Tworzymy wektor pomocniczy $y = Ux$ Następnie rozwiązujemy dwa układy równań liniowych:

$$Ly = b$$
$$Ux = y$$

W metodzie faktoryzacji LU zastosowano eliminację Gaussa, ale przez rozłożenie macierzy A na dwie macierze trójkątne L i U, algorytm podstawiania wstecz ma złożoność obliczeniową $O(n^2)$[1], a nie $O(n^3)$. W całości metoda faktoryzacji LU ma złożoność obliczeniową $O(n^3)$ [1].

Dodatkowo w metodzie faktoryzacji LU czasochłonne rozkładanie macierzy A na dwie macierze L i U wykonuje się tylko raz i można wielokrotnie rozwiązywać układ równań liniowych dla różnych wektorów b.

1.3 Metody iteracyjne rozwiązywania układów równań liniowych

Metody iteracyjne polegają na rozwiązaniu układu równań liniowych poprzez iteracyjne przybliżanie rozwiązania. W każdej iteracji obliczane jest nowe przybliżenie rozwiązania, które jest coraz bliższe rzeczywistemu rozwiązaniu. Metody iteracyjne są zazwyczaj mniej dokładne niż metody bezpośrednie, ale pozwalają na rozwiązanie dużych układów równań liniowych w krótszym czasie przy większym marginesie błędu.

Przykładem metody iteracyjnej jest metoda Jacobiego. Metoda ta polega na podziale macierzy A na macierz diagonalną D, macierz trójkątną dolną L i macierz trójkątną górną U, tak że $A = D - L - U$. Następnie iteracyjnie obliczane jest nowe przybliżenie rozwiązania $x^{(k+1)}$ na podstawie poprzedniego przybliżenia $x^{(k)}$:

$$x^{(k+1)} = D^{-1}(L+U)x^{(k)} + D^{-1}b \tag{2}$$

[2]

Kolejną metodą iteracyjną jest metoda Gaussa-Seidla, która działa w bardzo zbliżony sposób do metody Jacobiego, ale w każdej iteracji korzysta z najnowszych wartości $x^{(k+1)}$. Metodę Gaussa-Seidla można opisać równaniem:

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b \tag{3}$$

[2] Dzięki temu metoda Gaussa-Seidla zazwyczaj szybciej zbiega do rozwiązania niż metoda Jacobiego.

1.4 Wektor residuum

W przypadku metod iteracyjnych, wskaźnikiem jakości rozwiązania układu równań liniowych w obecnej iteracji jest wektor residuum. Wektor residuum to wektor, który można uzyskać wzorem:

$$res = Ax - b \tag{4}$$

Wartość wektora residuum mówi o tym, jak bardzo rozwiązanie uzyskane w danej iteracji różni się od rzeczywistego rozwiązania. Im mniejsza wartość wektora residuum, tym lepsze przybliżenie rozwiązania. Gdy norma wektora residuum jest mniejsza od zadanego progu błędu, można uznać, że rozwiązanie jest wystarczająco dokładne. Normę wektora oblicza się wzorem:

$$||res|| = \sqrt{\sum_{i=1}^{n} res_i^2} \tag{5}$$

2 Metody Jacobiego i Gaussa-Seidla dla macierzy z silną dominacją w wierszach

2.1 Macierz A i wektor b

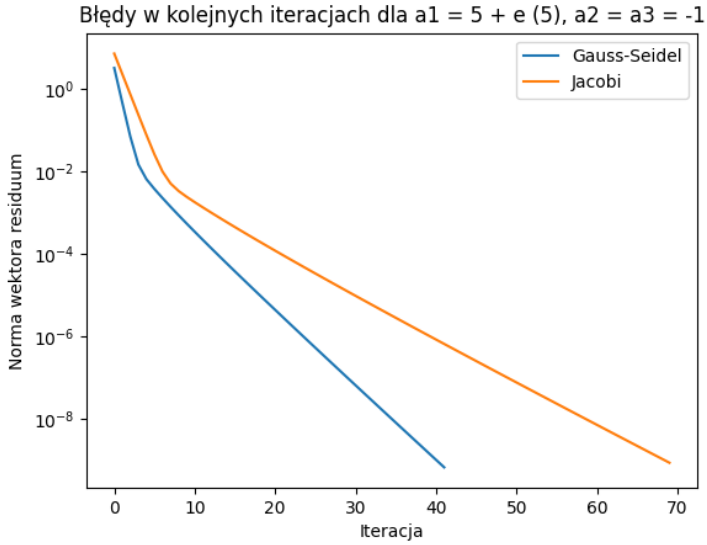
Macierze A i wektor b użyte dla algorytmów w tej sekcji to:

$$A = \begin{bmatrix} 5 & -1 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ -1 & 5 & -1 & -1 & \cdots & 0 & 0 & 0 & 0 \\ -1 & -1 & 5 & -1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 5 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 5 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 5 & -1 & -1 \\ 0 & 0 & 0 & 0 & \cdots & -1 & -1 & 5 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1 & -1 & 5 \end{bmatrix} \tag{6}$$

$$b = \begin{bmatrix} sin(3 \cdot 1) \\ sin(3 \cdot 2) \\ sin(3 \cdot 3) \\ \vdots \\ sin(3 \cdot n) \end{bmatrix} \tag{7}$$

Macierz A ma rozmiar $n \times n$, gdzie $n = 964$, a wektor b ma długość n.

2.2 Porównanie metod Jacobiego i Gaussa-Seidla



Na powyższym wykresie przedstawiono zmiany wartości normy wektora residuum w zależności od liczby iteracji dla metod Jacobiego i Gaussa-Seidla. Jak widać, metoda Gaussa-Seidla zbiega do rozwiązania znacznie szybciej niż metoda Jacobiego. Dla zadanego progu błędu $\epsilon = 10^{-9}$ metoda Gaussa-Seidla osiąga zbieżność w 42 iteracjach, podczas gdy metoda Jacobiego osiąga zbieżność w 70 iteracjach.

Metoda	Liczba iteracji	Czas obliczeń [s]	Norma wektora residuum
Jacobiego	70	11.79	$8.58 \cdot 10^{-10}$
Gaussa-Seidla	42	8.46	$6.66 \cdot 10^{-10}$

Table 1: Porównanie liczby iteracji, czasu obliczeń i normy wektora residuum dla metod Jacobiego i Gaussa-Seidla

Jako, że metoda Gaussa-Seidla wykonuje zbliżone obliczenia do metody Jacobiego, to mniejsza liczba iteracji będzie oznaczała krótszy czas obliczeń.

3 Metody Jacobiego i Gaussa-Seidla dla macierzy rozbieżnej

3.1 Macierz A i wektor b

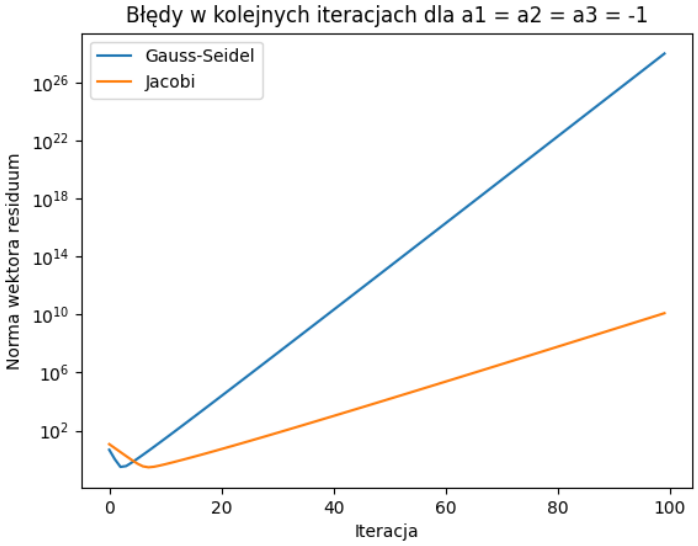
Macierze A i wektor b użyte dla algorytmów w tej sekcji to:

$$A = \begin{bmatrix} 3 & -1 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & \cdots & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 3 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & \cdots & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1 & -1 & 3 \end{bmatrix} \tag{8}$$

$$b = \begin{bmatrix} sin(3 \cdot 1) \\ sin(3 \cdot 2) \\ sin(3 \cdot 3) \\ \vdots \\ sin(3 \cdot n) \end{bmatrix} \tag{9}$$

Macierz A ma rozmiar $n \times n$, gdzie $n = 964$, a wektor b ma długość n.

3.2 Porównanie metod Jacobiego i Gaussa-Seidla



Na powyższym wykresie przedstawiono zmiany wartości normy wektora residuum w zależności od liczby iteracji dla metod Jacobiego i Gaussa-Seidla. Jak widać, norma wektora residuum dla obu metod zbiega do nieskończoności. Wynika to z faktu, że macierz macierz $T = D^{-1}(L+U)$ ma promień spektralny (największa wartość własna - eigenvalue) większy od 1[3]. W takim przypadku metody iteracyjne nie zbiegają do rozwiązania.

Metoda	Liczba iteracji	Czas obliczeń [s]	Norma wektora residuum
Jacobiego	100	17.47	$1.25 \cdot 10^{10}$
Gaussa-Seidla	100	20.08	$9.74 \cdot 10^{27}$

Table 2: Porównanie liczby iteracji, czasu obliczeń i normy wektora residuum dla metod Jacobiego i Gaussa-Seidla

Jak widać, obie metody zakończyły swoje działanie po 100 iteracjach, co jest maksymalną liczbą iteracji w moim programie.

4 Faktoryzacja LU

Macierzy przedstawioną w poprzedniej sekcji można poddać faktoryzacji LU, tam gdzie metody Jacobiego i Gaussa-Seidla zawiodły.

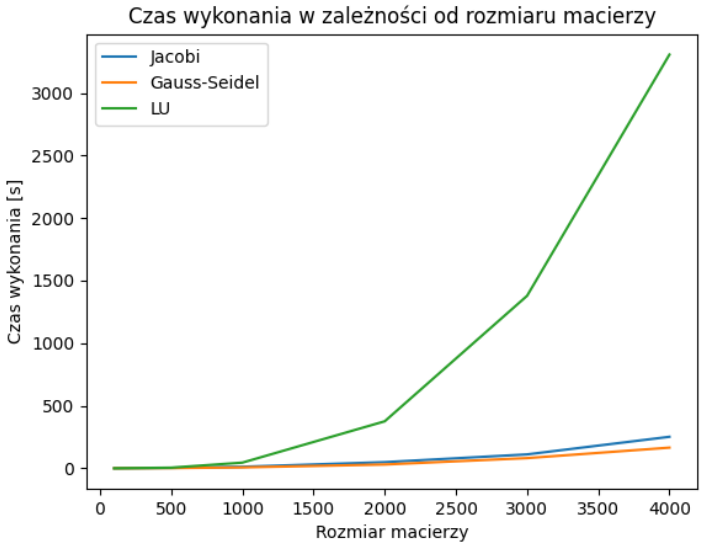
Metoda	Czas obliczeń [s]	Norma wektora residuum
Faktoryzacja LU	39.13	$2.36 \cdot 10^{-13}$

Table 3: Wyniki faktoryzacji LU dla macierzy rozbieżnej

Jak widać, faktoryzacja LU pozwoliła na uzyskanie dokładnego rozwiązania układu równań liniowych, jednak czas obliczeń był znacznie dłuższy niż dla metod iteracyjnych.

5 Porównanie metod Jacobiego, Gaussa-Seidla i faktoryzacji LU

Metody Jacobiego i Gaussa-Seidla i faktoryzacji LU zostały uruchomione dla macierzy z sekcji Metody Jacobiego i Gaussa-Seidla dla macierzy z silną dominacją w wierszach. Rozmiary macierzy i wektora n należą do zbioru $\{500, 1000, 2000, 3000, 4000\}$.



Z powyższych tabel wynika, że liczba iteracji dla metod Jacobiego i Gaussa-Seidla jest zbliżona dla macierzy o dowolnym rozmiarze n, natomiast iteracje były dłuższe dla większych macierzy.

Metoda	Liczba iteracji	Czas obliczeń [s]	Norma wektora residuum
Jacobiego	69	0.15	$9.11 \cdot 10^{-10}$
Gaussa-Seidla	41	0.09	$9.11 \cdot 10^{-10}$
Faktoryzacja LU	1	0.06	$7.93 \cdot 10^{-16}$

Table 4: Porównanie dla $n = 100$

Metoda	Liczba iteracji	Czas obliczeń [s]	Norma wektora residuum
Jacobiego	71	3.30	$8.24 \cdot 10^{-10}$
Gaussa-Seidla	42	1.90	$8.24 \cdot 10^{-10}$
Faktoryzacja LU	1	5.13	$1.78 \cdot 10^{-15}$

Table 5: Porównanie dla $n = 500$

Metoda	Liczba iteracji	Czas obliczeń [s]	Norma wektora residuum
Jacobiego	69	12.34	$9.03 \cdot 10^{-10}$
Gaussa-Seidla	40	7.25	$7.25 \cdot 10^{-10}$
Faktoryzacja LU	1	45.42	$2.50 \cdot 10^{-15}$

Table 6: Porównanie dla $n = 1000$

Metoda	Liczba iteracji	Czas obliczeń [s]	Norma wektora residuum
Jacobiego	70	49.15	$9.62 \cdot 10^{-10}$
Gaussa-Seidla	42	30.34	$9.62 \cdot 10^{-10}$
Faktoryzacja LU	1	375.68	$3.53 \cdot 10^{-15}$

Table 7: Porównanie dla $n = 2000$

Metoda	Liczba iteracji	Czas obliczeń [s]	Norma wektora residuum
Jacobiego	70	111.21	$9.99 \cdot 10^{-10}$
Gaussa-Seidla	42	81.47	$9.99 \cdot 10^{-10}$
Faktoryzacja LU	1	1378.66	$4.31 \cdot 10^{-15}$

Table 8: Porównanie dla $n = 3000$

Metoda	Liczba iteracji	Czas obliczeń [s]	Norma wektora residuum
Jacobiego	69	251.88	$8.00 \cdot 10^{-10}$
Gaussa-Seidla	40	165.24	$8.00 \cdot 10^{-10}$
Faktoryzacja LU	1	3306.43	$4.93 \cdot 10^{-15}$

Table 9: Porównanie dla $n = 4000$

Faktoryzacja LU nie skaluje się tak dobrze jak metody iteracyjne, co wynika z faktu, że faktoryzacja LU ma złożoność obliczeniową $O(n^3)$.

6 Wnioski

Chociaż metoda faktoryzacji LU daje dokładne wyniki, wymaga znacznie więcej czasu obliczeń niż metody iteracyjne, w których można kontrolować dokładność rozwiązania przez zdefiniowanie progu błędu. Dla większych macierzy faktoryzacja LU jest niepraktyczna przez słabą skalowalność.

Jednak metody iteracyjne czasem zawodzą i nie zbiegają do wyniku, więc w przypadku gdy wykryje się rozbieżność, pozostaje nam faktoryzacja LU, która zawsze zwróci rozwiązanie, ale kosztem czasu obliczeń.

References

[1] Grzegorz Fotyga. *Układy Równań Liniowych*. Wykład. 2024.

[2] Grzegorz Fotyga. *Układy Równań Liniowych - algorytmy iteracyjne*. Wykład. 2023.

[3] Zhiliang Xu. *7.3 The Jacobi and Gauss-Seidel Iterative Methods*. Wykład.