Układy równań liniowych

Julia Chomicka s188873

---

1. Wstęp

Teoria **układów równań liniowych** jest działem algebry liniowej leżącej u podstaw nowoczesnej matematyki. Algorytmami obliczeniowymi zajmuje się dział nazywany numeryczna algebra liniowa. Metody rozwiązywania układów odgrywają ważną rolę między innymi w inżynierii, informatyce, ekonomi, chemii czy fizyce. Jest ono przedmiotem badań wielu ośrodków naukowych, ponieważ bez niego rozwój wymienionych wyżej dziedzin wiedzy byłby niemożliwy.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$$\mathbf{Ax} = \mathbf{b}$$

Rozwiązanie układu równań liniowych polega na wyznaczeniu wektora $x = [x_1, x_2, ..., x_n]$ liczb rzeczywistych spełniających ten układ, czyli takich, że dla każdego i ($1 <= i <= m$) zachodzi:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i$$

Metody rozwiązania układów liniowych dzielimy na **bezpośrednie** i **iteracyjne**.

Metody bezpośrednie pozwalają na uzyskanie wyniku po określonej liczbie operacji (działań arytmetycznych). Należą do nich między innymi metoda Gaussa, czy też **faktoryzacja LU.**

Natomiast metody iteracyjne polegają na wyznaczeniu ciągu kolejnych rozwiązań przybliżonych (x0, x1, x2 . . . xn), który jest zbieżny do dokładnego rozwiązania układu równań liniowych. Do tych metod zaliczamy na przykład metodę iteracji prostych - **Jacobiego** oraz metodę **Gaussa-Seidela.**

---

2. Implementacja

W moim projekcie przykładowe metody rozwiązania układów liniowych zaimplementowałam w języku *python.* Wykorzystałam bibliotekę *matplotlib* do tworzenia wykresów.

Do przedstawienia działania metody bezpośredniej faktoryzacji LU oraz metod iteracyjnych Jacobiego oraz Gaussa-Seidela posłużę się równaniem **Ax = b**, gdzie A jest macierzą systemową, b jest wektorem pobudzenia, natomiast x jest wektorem rozwiązań reprezentującym szukaną wielkość fizyczną.

Macierz pasmowa **A** będzie miała wymiary N x N. Główna diagonala zawierać będzie wartości a1 = 13 lub a1 = 3 w zależności od prezentowanych przykładów, natomiast pozostałe cztery diagonale przyjmują wartości a2 = a3 = -1. Macierz A została zobrazowana poniżej:

$$\mathbf{A} = \begin{bmatrix} a1 & a2 & a3 & 0 & 0 & 0 & 0 & \dots & 0 \\ a2 & a1 & a2 & a3 & 0 & 0 & 0 & \dots & 0 \\ a3 & a2 & a1 & a2 & a3 & 0 & 0 & \dots & 0 \\ 0 & a3 & a2 & a1 & a2 & a3 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & a3 & a2 & a1 \end{bmatrix},$$

b jest wektorem o długości N, którego n−ty element ma wartość sin(n ·(9)).

2.1) Implementacja fakotryzacji LU

Jeden z wariantów metody eliminacji Gaussa, szczególnie stosowana, kiedy układ równań musi być rozwiązany dla wielu prawych stron (wektor b), natomiast lewa strona równania (macierz A) nie zmienia się.

-> Macierz A przedstawiamy jako iloczyn dwóch macierzy trójkątnych: **A = LU**. Jest to właśnie faktoryzacja (dekompozycja) LU. Do wyznaczenia rozkładu LU posłużyłam się Metodą Doolittle'a

-> następnie po stworzeniu macierzy trójkątnej dolnej L oraz górnej U postawiamy je do wcześniej stworzonego równania: **LUx = b**

-> kolejno tworzymy wektor pomocniczy **y = Ux** i rozwiązujemy układ równań: **Ly = b** za pomocą podstawienia wprzód. Łatwo możemy policzyć y, dzięki temu, że macierz L jest macierzą trojkątną dolną (Wyliczając niewiadomą z pierwszego wiersza, możemy policzyć niewiadomą z drugiego wiersza, aż do N-tego wiersza).

-> Mając y rozwiązujemy układ równań: **Ux = y** za pomocą podstawiania wstecz**.** Łatwo możemy policzyć x, dzięki temu, że macierz U jest macierzą trójkątną górną (Wyliczając niewiadomą z N-tego wiersza, możemy policzyć niewiadomą z N-1-tego wiersza, aż do pierwszego wiersza).

W ten sposób uzyskaliśmy rozwiązanie x.

2.2.1) Implementacja metody Jacobi’ego

Załóżmy, że $x^{(k)}$ jest przybliżeniem dokładnego rozwiązania **Ax = b**. Naturalny sposób na obliczenie kolejnego przybliżenia $x^{(k+1)}$ przedstawiono poniżej:

$$\begin{cases} x_1^{(k+1)} &= (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)})/a_{11} \\ x_2^{(k+1)} &= (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)})/a_{22} \\ x_3^{(k+1)} &= (b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)})/a_{33} \end{cases}$$

a więc w ogólności przepis na kolejne przybliżenia jest następujący:

$$x_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)})/a_{ii}$$

Wyznaczamy kolejne przybliżenia, dopóki norma wektora residuum nie osiągnie wyznaczonej przez nas bliskiej zeru wartości. Oznacza to, że różnica między przybliżonym rozwiązaniem $\tilde{x}$ a rzeczywistym x jest tak mała, że możemy przyjąć przybliżone rozwiązanie za to rzeczywiste.

Wektor residuum (r = $\tilde{x}$ − b) stosuje się w celu oszacowania błędu wnoszonego przez $\tilde{x}$. Norma wektora residuum to błąd przedstawiony w postaci skalara. Przepis na taką normę dla wektora residuum przedstawiono poniżej:

$$\|\mathbf{e}\|_2 = \sqrt{\sum_{j=1}^{n} e_j^2}$$

gdzie e_j w naszym przypadku to kolejne wartości wektora $\tilde{x}$

2.2.1) Implementacja metody Gaussa-Seidela

Metoda ta jest bardzo podobna do metody Jacob'iego, natomiast w niej do wyznaczenia kolejnych przybliżeń x korzystamy z aktualnego przybliżenia x_i. A więc przepis na kolejne przybliżenia jest następujący:

$$x_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)})/a_{ii}$$

---

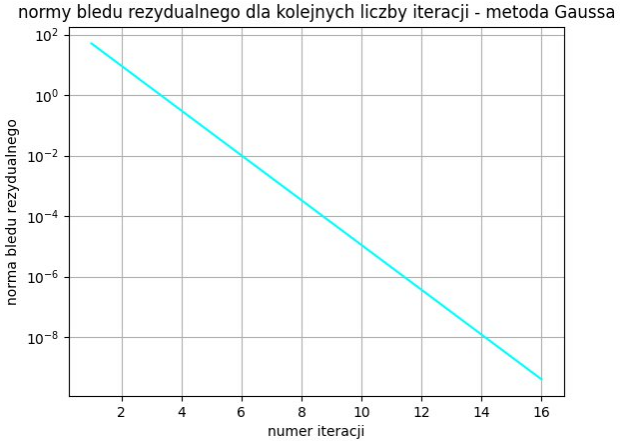
3. Analiza:

Wstępnie przeprowadziłam parę doświadczeń odnośnie metod iteracyjnych Jacob'iego oraz Gauss-Seidela. Wywołałam oba algorytmy dla wspomnianego wyżej równania, gdzie macierz pasmowa A była rozmiarów 973 x 973 oraz a1 = 13. Wektor b miał również długość 973.

```
WYNIK dla metody Jacobiego, a1 = 13
CZAS: 7.269552299985662 s
Liczba iteracji: 23
WYNIK dla metody Gaussa_Seidela, a1 = 13
CZAS: 5.761419499991462 s
Liczba iteracji: 16
```

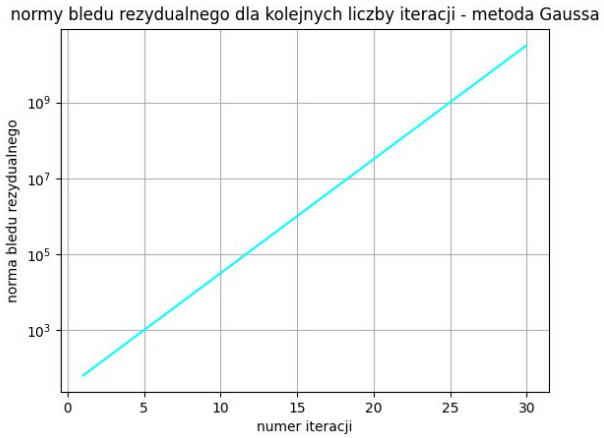
Metoda Gausa-Seidela przeprowadziła 16 iteracji, a metoda Jacob'iego 23 - możemy zauważyć, że metoda Gausa-Seidela potrzebowała mniej iteracji od metody Jacob'iego. Ponadto obliczenia zajęły jej około 1,508 mniej sekund.

Poniżej przedstawiono normę wektora residuum dla kolejnych iteracji dla obu metod:



Jak możemy zauważyć metody te są tak samo dokładne, ponieważ algorytmy te wykonują się, dopóki norma błędu residuum będzie mniejsza/równa zadanej przez nas wartości bliskiej zeru (w tym przykładzie 10^{-9}). To znaczy, że tyle będzie wynosiła różnica pomiędzy wynikiem rzeczywistym, a przybliżonym.

Następnie przeprowadziłam podobne doświadczenie, zmieniając wartość a1 na 3. Efektem tego był niekończący się program. Aby program mógł skończyć się trzeba było ograniczyć działania metod do danej liczby iteracji. Dzięki temu możemy przedstawić jak zmieniała się wartość tejże normy w zależności od numeru iteracji:



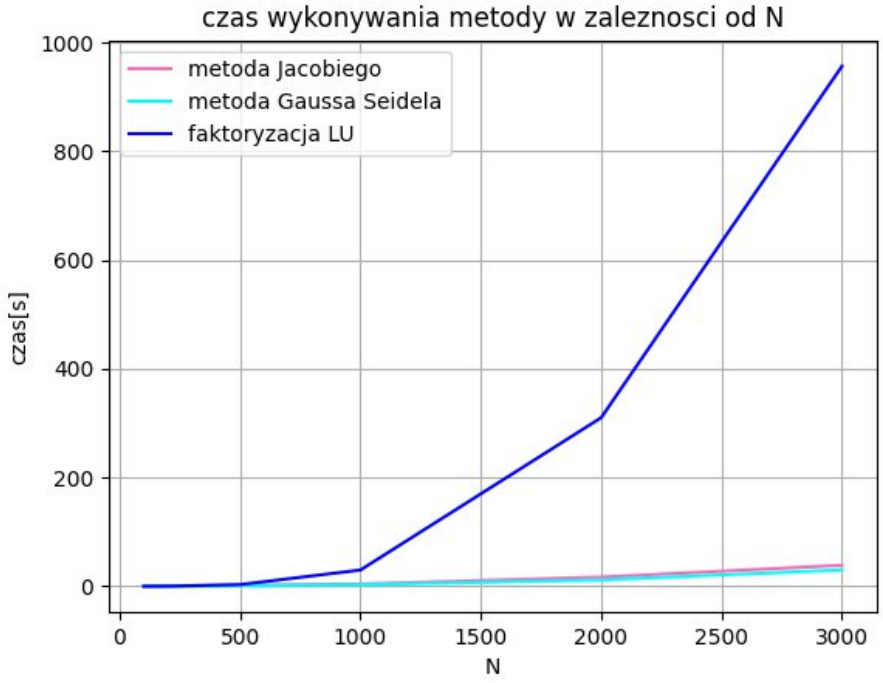
Możemy zauważyć, że norma z wektora residuum ciągle rosła dla obu metod, dlatego programy te nie kończyły się. Obie metody nie poradziły sobie z rozwiązaniem tego równania. Dla tej wartości a1 metody iteracyjne nie zbiegają się.

Kolejnym krokiem jaki zrobiłam było przetestowanie działania bezpośredniej metody czyli faktoryzacji LU. Metodę tą wywołałam dla równania poprzednio wykorzystanego czyli dla a1 równego 3. W przeciwieństwie do obu wymienionych wcześniej metod, faktoryzacja LU poradziła sobie z wyznaczeniem rozwiązania.

```
Norma bledu rezydualnego dla N=973, a1=3, faktoryzacja LU: r=7.474855258901181e-13
```

Po policzeniu normy błędu rezydualnego widzimy, że rozwiązanie zostało dobrze wyznaczone, ponieważ norma ta jest bardzo mała, bliska zeru.

Aby móc lepiej porównać czas wykonania trzech metod rozwiązywania układów równań, wywołałam je kilkukrotnie dla różnych wielkości macierzy. Wartości głównej diagonali macierzy A były równe a1 = 13. Wielkość macierzy A była równa N x N, a długość wektora b była równa N. N kolejno przybierało wartości: 100, 200, 500, 1000, 2000, 3000. Po zaimplementowaniu powyższego algorytmu stworzyłam wykresy zależności czasu dla każdej metody w zależności od N. Wykres prezentuje się następująco:

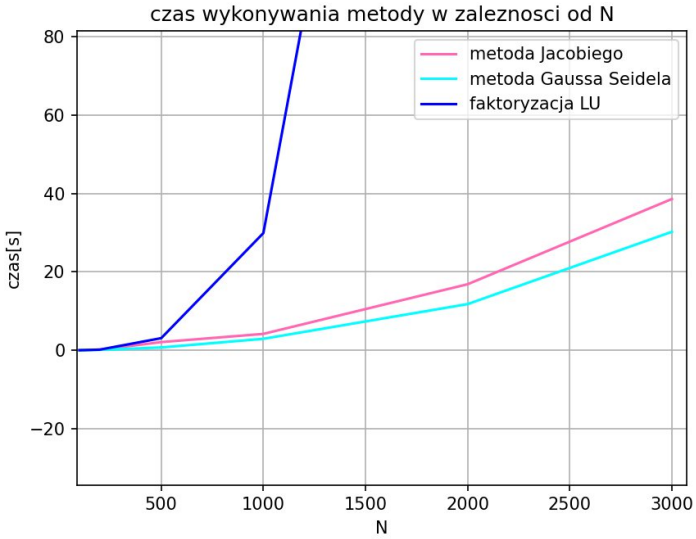


Z wykresu możemy zauważyć, że zdecydowanie metoda bezpośrednia faktoryzacji LU wymaga dużo więcej czasu niż przedstawione obie metody iteracyjne. Dla mniejszych N różnica ta nie jest bardzo zauważalna, natomiast dla większych N różnica jest zdecydowana.

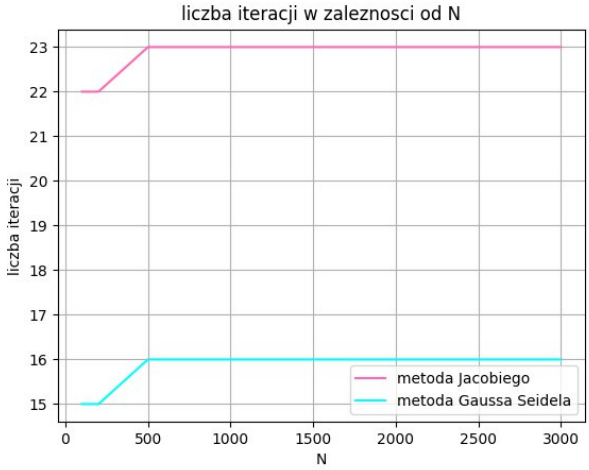
Poniżej zostały pokazane wybiórczo różnice dla N = 500, 1000, oraz 3000:

N	różnica LU/Jacobi	różnica LU/ Gauss-Seidel	różnica Jacobi / Gauss-Seidel
500	2,1727	2,4933	0,3206
1000	25,6974	26,9862	1,2888
3000	865,5991	877,3150	11,7159

Jeśli chodzi o metodę Jacob'iego i Gaussa – Seidela, różnice czasowe nie są znaczące dla mniejszych N, a są trochę większe dla większych N - szybsza okazała się być metoda Gaussa – Seidela. Przedstawia to poniższy wykres, który jest przybliżeniem wykresu przedstawionego poniżej:



Poniżej został również przedstawiony wykres pokazujący liczbę iteracji w zależności od N dla obu metod iteracyjnych. Wciąż możemy zauważyć, że metoda Gaussa – Seidela potrzebuje mniej iteracji.



4. Wnioski

Zarówno metoda bezpośrednia czyli faktoryzacji LU oraz metoda Jacob'iego i Gaussa Seidela działa tym dłużej im większa jest macierz A i b. Przy większych rozmiarach macierzy metoda faktoryzacji LU działa bardzo długo, natomiast zawsze znajdziemy tą metodą dokładny wynik. Obie metody iteracyjne plusują dużo krótszym działaniem dla większych macierzy, natomiast te metody nie zawsze znajdą rozwiązanie układu równań. Gauss-Seidel zbiega się, jeżeli macierz A jest symetryczna i dodatnio określona, oraz macierz A jest diagonalnie dominująca (tzn. $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$). Metoda Jacobiego zbiega się, jeśli macierz A jest diagonalnie dominująca. Metoda Gaussa-Seidela jest szybszą metodą iteracyjną (i potrzebuje mniej iteracji) od metody Jacobiego, ponieważ w metodzie Jacobiego wartości kolejnych przybliżeń są aktualizowane na podstawie poprzednich wartości, a w Gaussie-Seidlu wartości są aktualizowane „w locie", kiedy są obliczane. Moim zdaniem dla mniejszych macierzy warto skorzystać z faktoryzacji LU, by zagwarantować istnienie poprawnego wyniku, natomiast dla większych macierzy skorzystałabym z metody iteracyjnej Gaussa-Seidela, by doczekać się wyniku, upewniając się, czy napewno dla mojego układu równań metoda iteracyjna nie będzie problematyczna.