Mateusz Fydrych 193410

Układy równań liniowych

# 1.Wstęp:

Projekt ten ma na celu zbadanie i implementację trzech metod rozwiązywania układów równań liniowych: dwóch iteracyjnych - metody Jacobiego i Gaussa-Seidla, oraz jednej bezpośredniej - faktoryzacji LU. Te metody są kluczowe w kontekście rozwiązywania układów równań wynikających z dyskretyzacji równań różniczkowych, które są powszechnie stosowane w wielu dziedzinach nauki i techniki, takich jak elektronika, elektrodynamika, mechanika (w tym lotnictwo, biomechanika, motoryzacja), badania wytrzymałości materiałów i konstrukcji, symulacje odkształceń, naprężeń, przemieszczeń i drgań, akustyka, fotonika, termodynamika, dynamika płynów i wiele innych.

W praktyce, układy równań, które muszą być rozwiązane, często zawierają setki milionów niewiadomych, a ich obliczenia mogą trwać wiele godzin, a nawet dni, nawet przy użyciu najnowocześniejszych superkomputerów. Dlatego opracowanie nowych, wydajnych metod rozwiązania, które są dostosowane do współczesnych architektur komputerowych, stanowi duże wyzwanie zarówno z punktu widzenia matematyki, jak i informatyki. Jest to obszar intensywnych badań w wielu ośrodkach naukowych na całym świecie, ponieważ bez takich rozwiązań, dalszy rozwój wielu dziedzin nauki i techniki byłby niemożliwy.

Układ równań wygląda następująco:

$$Ax = b$$

gdzie A jest macierzą systemową b jest wektorem pobudzenia, natomiast x jest wektorem rozwiązań reprezentującym szukaną wielkość fizyczną.

# 2. Zadanie A

Celem tego zadania było utworzenie układu równań podanego powyżej. W moim przypadku macierze wyglądają następująco:

$$A = \begin{bmatrix} 9 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & 9 & -1 & -1 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & -1 & 9 & -1 & -1 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . & . & . & \dots & 0 \\ . & . & . & . & . & . & . & . & \dots & 0 \\ . & . & . & . & . & . & . & . & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & -1 & -1 & 9 \end{bmatrix} \text{ NxN } \text{ ,N=910}$$

$$b = \begin{bmatrix} sin(4) \\ sin(8) \\ sin(12) \\ sin(16) \\ . \\ . \\ . \\ 0 \end{bmatrix} \text{ Nx1, N=910}$$

# 3. Zadanie B

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

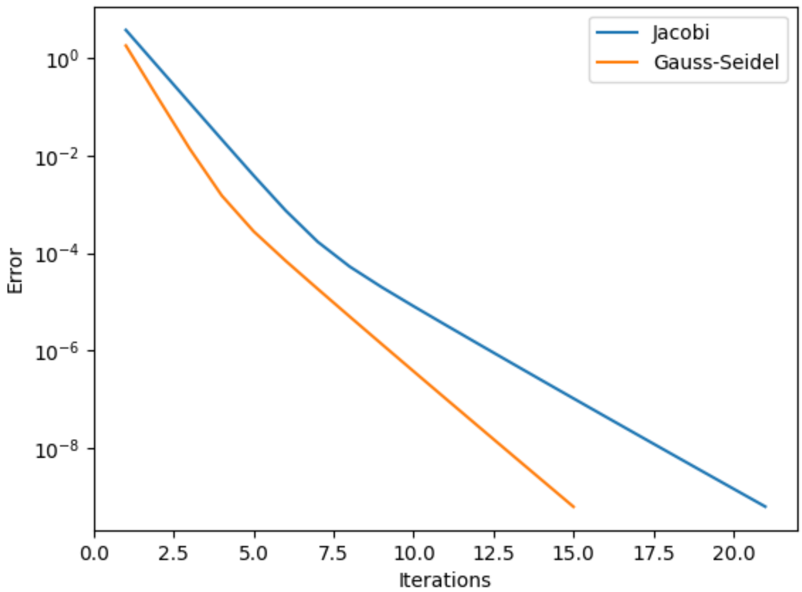
Równanie dla metody Jacobiego

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

Równanie dla metody Gaussa-Seidela

Rezultaty dla macierzy A były następujące:

| METODA | LICZBA ITERACJI | CZAS | BŁĄD |
|---|---|---|---|
| **Jacobi** | 21 | 4.68470 | 6.21470e-10 |
| **Gauss-Seidel** | 15 | 3.20845 | 6.17963e-10 |



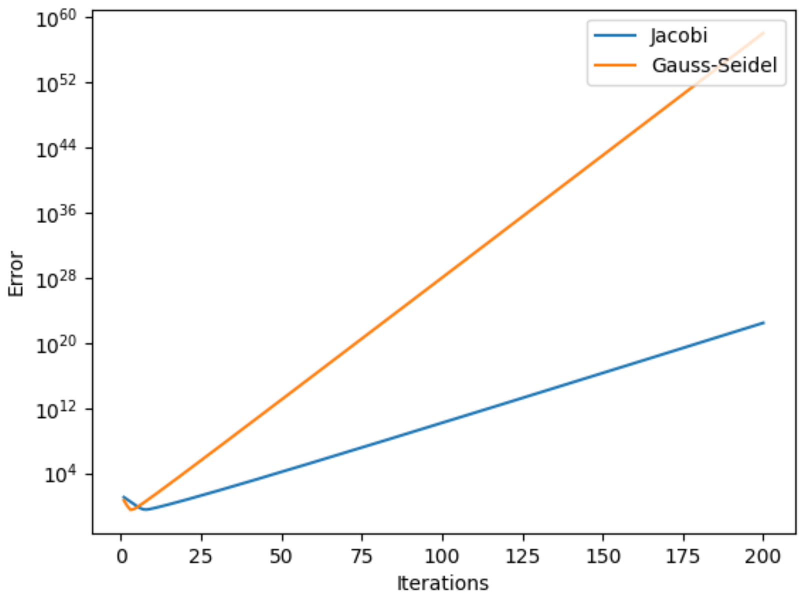

Obie metody konsekwentnie liniowo dążą ku mniejszemu błędowi. Na obu liniach widać punkt, w którym błąd zaczyna maleć wolniej. W metodzie Gaussa-Seidela błąd maleje szybciej niż w metodzie Jacobiego.W związku z tym program wyliczy wynik w mniejszej ilości iteracji i szybciej. Jest to związane z tym, że bazuje ona na <u>najnowszych</u> danych, a metoda Jacobiego na wektorze poprzedniej iteracji.

# 4.Zadanie C

$$A_c = \begin{bmatrix} 3 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & -1 & 3 & -1 & -1 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . & . & . & \dots & 0 \\ . & . & . & . & . & . & . & . & \dots & 0 \\ . & . & . & . & . & . & . & . & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & -1 & -1 & 3 \end{bmatrix}$$

Rezultaty dla macierzy $A_c$ były następujące:

| METODA | LICZBA ITERACJI | CZAS | BŁĄD |
|---|---|---|---|
| **Jacobi** | 200 | 44.47827 | 2.79241e+22 |
| **Gauss-Seidel** | 200 | 43.01261 | 1.03509e+58 |




W układzie równań z $A_c$ wyniki nie są zadowalające. Obie metody nie są w stanie znaleźć poprawnego rozwiązania dla danej macierzy.

Pomimo początkowego dobrego kursu dwóch lini błędu, już po kilku iteracjach obie metody zaczęły rosnąć. W implementacji warto zrobić górny punkt graniczny, który będzie przerywał program dla takich macierzy. Bez takiej granicy, większa liczba maksymalnej iteracji tylko przedłuży program, a nie da optymalnego rezultatu ze względu na rozbieżność tych metod w tym przypadku. W metodzie Gaussa-Seidela błąd rośnie diametralnie szybciej, co pozwoli nam na szybsze wyłapanie takiego błędu. Taki wykres udowadnia, że metody te nie są zbieżne dla tego konkretnego układu równań.

# 5. Zadanie D

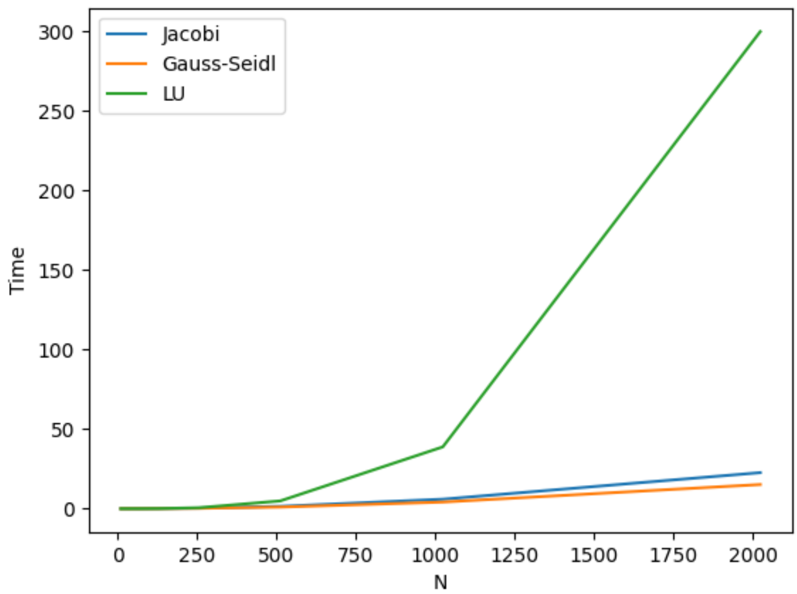
Rezultaty dla macierzy A były następujące:

| METODA | LICZBA ITERACJI | CZAS | BŁĄD |
|---|---|---|---|
| **Jacobi** | 21 | 4.68470 | 6.21470e-10 |
| **Gauss-Seidel** | 15 | 3.20845 | 6.17963e-10 |
| **Faktoryzacja LU** | nie dotyczy | 27.37054 | 2.50471e-13 |

Wynik jest dokładniejszy, jednak kosztem czasu. Czas wyliczania wydłużył się średnio 6,94 krotnie względem dwóch badanych metod iteracyjnych. Warto zauważyć, że w metodzie bezpośredniej błąd jest mniejszy. Norma residuum dla metody bezpośredniej wynosi: 2.50471e-13.

# 6. Zadanie E

Wykres pokazujący zależności czasu wyznaczenia rozwiązania dla trzech badanych metod w zależności od liczby niewiadomych N = {10,25,64,128,256,512,1024,2024}:




# 7. Zadanie F - wnioski

Testy zostały przeprowadzone metodami Jacobiego, Gaussa-Seidela i faktoryzacji LU. Obliczenia były robione bez dodatkowych bibliotek na procesorze Apple M2. Warto podkreślić, że obliczenia te da się zrobić szybciej (np. korzystając z bilblioteki Numpy). W testach metoda Gaussa-Seidela okazała się być najszybsza, następnie metoda Jacobiego, a najwolniej rezultat wyliczała metoda faktoryzacji LU. Mimo, że metoda bezpośrednia jest dokładniejsza, jest zdecydowanie wolniejsza od pewnego *n* niż metody iteracyjne. Metoda Gaussa-Seidela okazała się najszybsza, dlatego że bazuje ona na najnowszych danych, a metoda Jacobiego na wektorze poprzedniej iteracji. Metody iteracyjne nie zawsze dają dokładny wynik. Warto więc zauważając rozbieżność skorzystać z faktoryzacji LU.