# Rozwiązywanie układu równań liniowych metodą GEPP

Aleksandra Syska

April 28, 2025

## 1 Wstęp

Celem projektu jest rozwiązanie układu równań liniowych metodą eliminacji Gaussa z częściowym wyborem elementu głównego (**GEPP**) oraz porównanie wyników z rozwiązaniem uzyskanym za pomocą wbudowanej funkcji w programie MATLAB.

## 2 Treść zadania

Rozważamy układ równań liniowych postaci:

$$Cz = c, \quad \text{gdzie} \quad C = A + iB, \quad A, B \in \mathbb{R}^{n\times n}, \quad c = a + ib, \quad z = x + iy, \quad x, y, a, b \in \mathbb{R}^n.$$

W celu zastosowania eliminacji Gaussa z częściowym wyborem elementu głównego (**GEPP**), przekształcamy problem do postaci macierzy rozszerzonej:

$$M = \begin{pmatrix} A & -B \\ B & A \end{pmatrix} \in \mathbb{R}^{2n\times 2n}.$$

Następnie porównujemy wyniki z tymi, które uzyskamy za pomocą funkcji wbudowanej w MATLAB-ie.

## 3 Podstawowe pojęcia i wprowadzenie do metody

Eliminacja Gaussa to algorytm służący do rozwiązywania układów równań liniowych poprzez przekształcenie macierzy współczynników do postaci **górnotrójkątnej**, a następnie zastosowanie podstawiania wstecznego w celu znalezienia rozwiązania.

### 3.1 Macierz współczynników i wektor wyników

Układ równań liniowych można zapisać w postaci macierzowej:

$$Ax = b,$$

gdzie $A \in \mathbb{R}^{n\times n}$ to macierz współczynników, $x \in \mathbb{R}^n$ to wektor niewiadomych, a $b \in \mathbb{R}^n$ to wektor wyników.

Metoda eliminacji Gaussa polega na przekształceniu układu do postaci:

$$Ux = c,$$

gdzie $U$ jest macierzą górnotrójkątną. Rozwiązanie wyznaczamy stosując podstawianie wsteczne.

# 4 Eliminacja Gaussa z częściowym wyborem elementu głównego (GEPP)

Podstawowym problemem klasycznej eliminacji Gaussa jest możliwość dzielenia przez bardzo małe liczby, co prowadzi do błędów numerycznych. Aby tego uniknąć, stosuje się **częściowy wybór elementu głównego** (**pivoting**), czyli zamianę wierszy tak, aby największy (modułem) element danej kolumny znajdował się na diagonali głównej.

## 4.1 Etapy algorytmu GEPP

Algorytm GEPP przebiega w następujących krokach:

1. **Eliminacja współczynników** – dla każdej kolumny wybieramy element o największej wartości bezwzględnej spośród dostępnych wierszy i zamieniamy wiersze, aby umieścić go na przekątnej.

2. **Redukcja do macierzy górnotrójkątnej** – dla każdego wiersza poniżej przekątnej odejmujemy odpowiednią wielokrotność bieżącego wiersza.

3. **Podstawianie wsteczne** – po otrzymaniu macierzy górnotrójkątnej rozwiązujemy układ równań dla niewiadomych.

## 4.2 Matematyczne wyprowadzenie eliminacji Gaussa

Rozważmy układ równań:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n. \end{aligned}$$

Chcemy wyzerować wszystkie elementy pod $a_{11}$. Dla $i = 2, 3, \ldots, n$ obliczamy współczynnik:

$$m_{i1} = \frac{a_{i1}}{a_{11}}.$$

Następnie aktualizujemy wiersze:

$$a_{ij} = a_{ij} - m_{i1}a_{1j}, \quad \text{dla } j = 1, 2, \ldots, n.$$

Analogicznie postępujemy dla kolejnych kolumn, aż uzyskamy macierz trójkątną.

# 5 Podstawianie wsteczne

Po otrzymaniu macierzy górnotrójkątnej rozwiązujemy układ równań:

$$Ux = c,$$

gdzie $U$ ma postać:

$$\begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & u_{nn} \end{bmatrix}$$

Rozwiązujemy równania od dołu:

$$x_n = \frac{c_n}{u_{nn}},$$

$$x_{n-1} = \frac{c_{n-1} - u_{n-1,n}x_n}{u_{n-1,n-1}},$$

$$\vdots$$

$$x_1 = \frac{c_1 - \sum_{j=2}^{n} u_{1j}x_j}{u_{11}}.$$

# 6 Opis zaimplementowanych funkcji

W ramach projektu zaimplementowano dwie kluczowe funkcje do przekształcania i rozwiązywania zespolonych układów równań liniowych metodą eliminacji Gaussa z częściowym wyborem elementu głównego.

## 6.1 Funkcja `create_equations`

Funkcja ta przekształca układ równań zespolonych:

$$Cz = c, \quad C \in \mathbb{C}^{n\times n}, \quad c \in \mathbb{C}^n$$

do równoważnego układu rzeczywistego:

$$Mx = w, \quad M \in \mathbb{R}^{2n\times 2n}, \quad w \in \mathbb{R}^{2n}.$$

Działanie funkcji można podsumować w następujących krokach:

- **Rozdzielenie części rzeczywistych i urojonych:**

$$C = A + iB, \quad c = a + ib, \quad A, B, a, b \in \mathbb{R}.$$

- **Utworzenie rzeczywistego układu blokowego:**

$$M = \begin{bmatrix} A & -B \\ B & A \end{bmatrix}, \quad w = \begin{bmatrix} a \\ b \end{bmatrix}.$$

- **Zwrócenie macierzy $M$ i wektora $w$ jako nowej reprezentacji problemu.**

**Przykład użycia**

```
C = [1+2i, 3-1i; 2+0i, 4+2i];
c = [5+1i; 6-2i];
[M, w] = create_equations(C, c);
```

## 6.2 Funkcja `solve_block_system`

Funkcja ta rozwiązuje zespolony układ równań $Cz = c$ poprzez:

1. **Przekształcenie układu** na rzeczywisty system blokowy za pomocą funkcji `create_equations`.
2. **Zastosowanie eliminacji Gaussa** z częściowym wyborem elementu głównego do rozwiązania układu rzeczywistego.
3. **Zrekonstruowanie rozwiązania zespolonego** poprzez połączenie części rzeczywistej i urojonej.

**Przykład użycia**

```
C = [2+1i, 1-0.5i; 1+0.5i, 3-1i];
c = [1+0.5i; 2-1i];
z = solve_block_system(C, c);
```

## 6.3 Obsługa macierzy osobliwych

Podczas eliminacji Gaussa funkcja sprawdza, czy macierz jest osobliwa (lub bliska osobliwości) poprzez porównanie największego elementu w kolumnie do wartości `eps`, czyli maszynowej precyzji obliczeń.

W przypadku wykrycia osobliwości funkcja zwraca komunikat ostrzegawczy i może zwrócić wartości NaN w rozwiązaniu.

# 7 Opis funkcji w skrypcie testującym

W celu ułatwienia analizy wyników działania algorytmu rozwiązującego układy równań zespolonych, w skrypcie zaimplementowano dwie funkcje pomocnicze:

- `display_system(C, c)`
  Funkcja służy do czytelnego wyświetlania układu równań zespolonych w formie tekstowej. Dla każdej linii systemu konstruuje ciąg znaków opisujący równanie w postaci algebry zespolonej, uwzględniając znaki liczb zespolonych oraz odpowiednie formatowanie pierwszego i kolejnych składników. Na końcu każdego równania wypisywana jest prawa strona ($c_i$).

- `display_error(error_value, threshold)`
  Funkcja odpowiedzialna za prezentację różnicy pomiędzy rozwiązaniem uzyskanym metodą własną a rozwiązaniem MATLAB-a. Jeśli błąd bezwzględny jest mniejszy niż zadany próg (`ERROR_THRESHOLD`), funkcja informuje, że różnica jest praktycznie zerowa. W przeciwnym przypadku wyświetla wartość błędu w formacie naukowym.

# 8 Generowanie wykresów

W skrypcie służącym do oceny jakości oraz szybkości rozwiązania układów równań zespolonych zaimplementowano następujące funkcjonalności:

- **Test wydajności**
  Dla różnych rozmiarów macierzy ($n = 10, 60, 110, \ldots, 1010$) mierzony jest czas rozwiązania układu $Cz = c$ przy użyciu:
  - własnej metody użytkownika (`solve_block_system`),
  - wbudowanej funkcji MATLAB-a (`C\c`).

  Czasy wykonania obu metod są zapisywane w odpowiednich tablicach, a następnie przedstawiane na wykresie zależności czasu od rozmiaru macierzy.

- **Analiza prędkości działania**
  Po wykonaniu testów dla wszystkich rozmiarów obliczana jest względna prędkość działania obu metod jako stosunek czasów wykonania:

  $$\text{speed\_ratio} = \frac{\text{czas\_mojej\_metody}}{\text{czas\_MATLAB}}$$

  Wyniki są wypisywane w konsoli.

- **Test dokładności**
  Dla tych samych rozmiarów macierzy przeprowadzono dodatkowy test dokładności rozwiązania. Obliczana jest względna norma błędu pomiędzy rozwiązaniem własnym a rozwiązaniem MATLAB-a:

  $$\text{błąd względny} = \frac{\|z_{\text{custom}} - z_{\text{MATLAB}}\|}{\|z_{\text{MATLAB}}\|}$$

  Wyniki błędów są zapisywane i prezentowane na wykresie.

- **Wizualizacja wyników**
  W obu przypadkach (wydajność i dokładność) wyniki są przedstawione graficznie:
  - Wydajność: wykres czasu wykonania w funkcji rozmiaru macierzy.
  - Dokładność: wykres względnego błędu w funkcji rozmiaru macierzy.

# 9 Przeprowadzone testy

W celu oceny poprawności i efektywności implementacji metody GEPP przeprowadzono serię testów numerycznych. Otrzymane wyniki porównano z rozwiązaniami uzyskanymi za pomocą MATLAB-a.

### 9.1 Test 1: Duża macierz losowa $5 \times 5$ (wydajność)

Celem testu było sprawdzenie wydajności implementacji dla macierzy o większym rozmiarze. Otrzymano następujące czasy wykonania:

- **Czas mojej metody:** 15.4286 s
- **Czas MATLAB:** 0.0714 s
- **Różnica w rozwiązaniach:** $2.457546 \times 10^{-10}$

Pomimo poprawności wyniku, wydajność mojej implementacji była znacznie gorsza niż MATLAB-a.

### 9.2 Test 2: Macierz dobrze uwarunkowana $3 \times 3$

W teście rozwiązano dobrze uwarunkowany układ równań. Otrzymane wyniki:

- **Moje rozwiązanie:** $\begin{bmatrix} 0.4339 + 0.0356i & 0.5702 - 0.3058i & 0.2668 + 0.3069i \end{bmatrix}$
- **Rozwiązanie MATLAB:** $\begin{bmatrix} 0.4339 + 0.0356i & 0.5702 - 0.3058i & 0.2668 + 0.3069i \end{bmatrix}$
- **Różnica w rozwiązaniach:** 2.077037e-16 $10^{-16}$

Wyniki obu metod są identyczne.

### 9.3 Test 3: Macierz Hermitowska $4 \times 4$

Test przeprowadzono dla macierzy Hermitowskiej. Otrzymano następujące wyniki:

- **Moje rozwiązanie:** zgodne z MATLAB-em
- **Różnica w rozwiązaniach:** 4.200181 $10^{-16}$

Metoda GEPP dała poprawne wyniki.

### 9.4 Test 4: Macierz źle uwarunkowana $3 \times 3$

Przetestowano układ z macierzą źle uwarunkowaną, czyli taką, dla której niewielkie zmiany danych wejściowych mogą powodować duże zmiany w rozwiązaniu. Otrzymano:

- **Moje rozwiązanie:** $\begin{bmatrix} 2 & 0 & 0 \end{bmatrix}$
- **Rozwiązanie MATLAB:** $\begin{bmatrix} 2 & 0 & 0 \end{bmatrix}$
- **Różnica w rozwiązaniach:** 0 (mniej niż $10^{-16}$)

Obie metody dały zgodne wyniki.

### 9.5 Test 5: Macierz prawie osobliwa $4 \times 4$

W przypadku macierzy bliskiej osobliwości MATLAB wygenerował ostrzeżenie:

- **Moje rozwiązanie:** NaN (nieokreślone wartości)
- **Rozwiązanie MATLAB:** NaN (nieokreślone wartości)
- **Różnica w rozwiązaniach:** NaN

Oba algorytmy prawidłowo wykryły problem i wskazały brak rozwiązania.

### 9.6 Test 6: Macierz osobliwa $3 \times 3$

W przypadku macierzy osobliwej MATLAB również zgłosił ostrzeżenie o osobliwości:

- **Moje rozwiązanie:** NaN
- **Rozwiązanie MATLAB:** NaN (z wyjątkiem jednej wartości $\infty$)
- **Różnica w rozwiązaniach:** NaN

Obie metody wykryły brak jednoznacznego rozwiązania.

### 9.7 Test 7: Macierz rzadka $5 \times 5$

Test przeprowadzono na rzadkiej macierzy, czyli takiej, w której większość elementów to zera. Otrzymano:

- **Czas mojej metody:** 0.0003 s
- **Czas MATLAB:** 0.0001 s
- **Różnica w rozwiązaniach:** $1.67685910^{-12}$

## 10 Podsumowanie analizy wyników

### 10.1 Porównanie różnicy w rozwiązaniach

| Test | Różnica w rozwiązaniach |
|---|---|
| Duża macierz losowa 5x5 | 2.457546e-10 |
| Macierz dobrze uwarunkowana 3x3 | 2.077037e-16 |
| Macierz Hermitowska 4x4 | 4.200181e-16 |
| Macierz źle uwarunkowana 3x3 | 0 (mniej niż 1e-16) |
| Macierz rzadka 5x5 | 1.676859e-16 |
| Macierz prawie osobliwa 4x4 | NaN |
| Macierz osobliwa 3x3 | NaN |

Table 1: Porównanie różnicy w rozwiązaniach

Przeprowadzone testy wykazały, że uzyskane rozwiązania są w większości przypadków zgodne z wynikami MATLAB-a. Różnice w wynikach były minimalne i nie przekraczały $10^{-10}$, co potwierdza poprawność implementacji. W przypadkach macierzy osobliwych oraz prawie osobliwych obie metody zwróciły wartości NaN, co jest zgodne z oczekiwanym zachowaniem numerycznym.

## 10.2 Porównanie czasu wykonania metod

| Test | Czas mojej metody (s) | Czas MATLAB (s) |
|---|---|---|
| Duża macierz losowa 5x5 | 15.4286 | 0.0714 |
| Macierz rzadka 5x5 | 0.0003 | 0.0001 |

Table 2: Porównanie czasu wykonania metod

Największa różnica pomiędzy obiema metodami dotyczyła jednak czasu wykonania. W przypadku dużej macierzy losowej $5 \times 5$ nasza implementacja była znacząco wolniejsza niż rozwiązanie MATLAB-a, co sugeruje możliwość dalszej optymalizacji.

# 11 Test działania metody w zależności od rozmiaru macierzy

W ramach testów sprawdzono czas wykonania oraz dokładność własnej metody rozwiązującej układy równań zespolonych w porównaniu z funkcją `MATLAB` w zależności od rozmiaru macierzy.

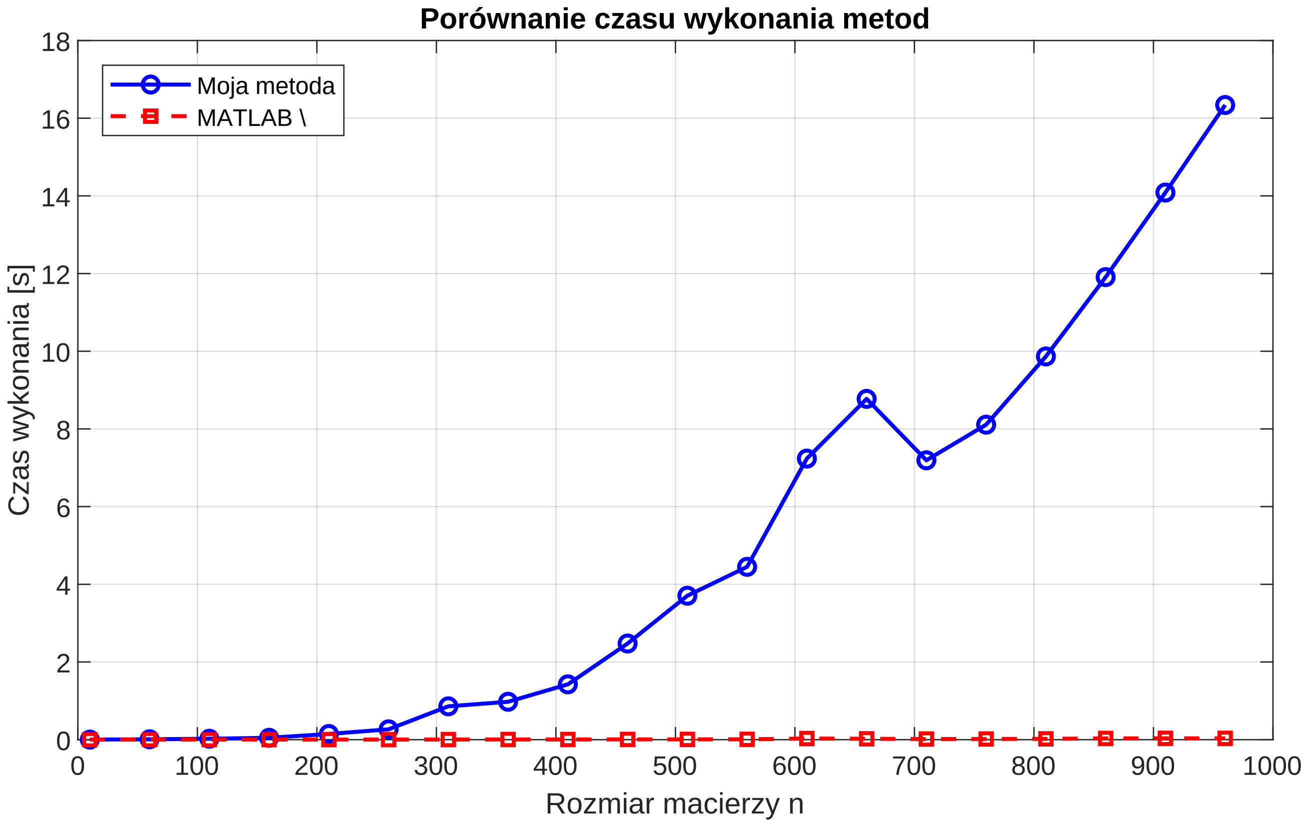
## 11.1 Wydajność

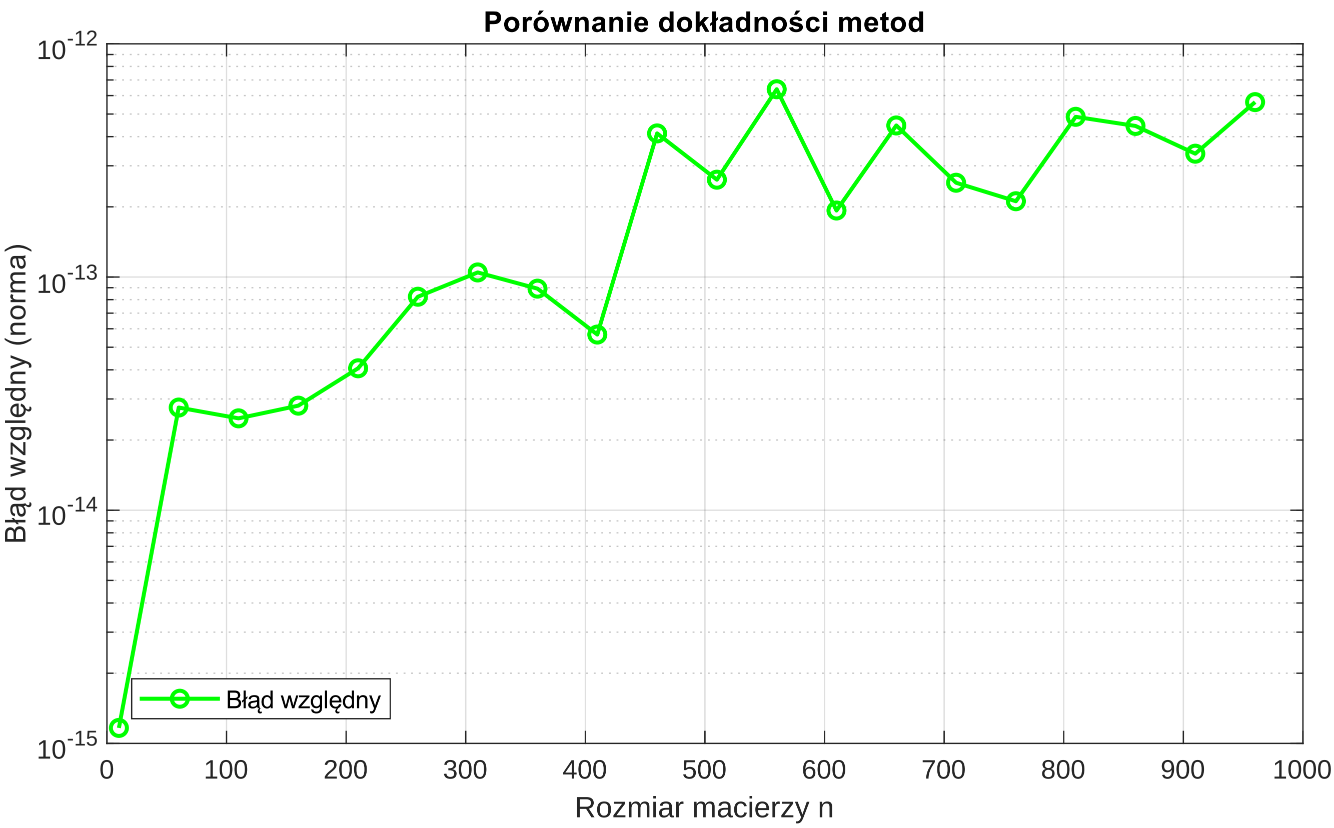
Na górnym wykresie przedstawiono czas wykonania obu metod w funkcji rozmiaru macierzy. Widać wyraźnie, że funkcja wbudowana MATLAB-a (\) działa szybciej od własnej implementacji, zwłaszcza dla większych rozmiarów $n$.

Dla małych macierzy ($n < 100$) różnice w czasie są stosunkowo niewielkie, jednak wraz ze wzrostem rozmiaru macierzy różnice te stają się coraz bardziej wyraźne.

## 11.2 Dokładność

Można zauważyć, że dla wszystkich rozmiarów macierzy względny błąd pozostaje bardzo niski, zwykle na poziomie $10^{-14}$ do $10^{-12}$. Oznacza to, że własna metoda zachowuje wysoką dokładność rozwiązania, praktycznie równą dokładności funkcji MATLAB-a, niezależnie od rozmiaru układu.


Porównanie czasu wykonania metod
Moja metoda
MATLAB \
Czas wykonania [s]
Rozmiar macierzy n
0
2
4
6
8
10
12
14
16
18
0
100
200
300
400
500
600
700
800
900
1000


Porównanie dokładności metod
Błąd względny
Błąd względny (norma)
Rozmiar macierzy n
10^-15
10^-14
10^-13
10^-12
0
100
200
300
400
500
600
700
800
900
1000

## 12 Wnioski

Podsumowując, opracowana metoda charakteryzuje się wysoką dokładnością rozwiązań przy akceptowalnym wzroście czasu wykonania w porównaniu do funkcji wbudowanej w MATLAB-a. Metoda może być stosowana w praktyce tam, gdzie priorytetem jest dokładność rozwiązania, a czas wykonania nie jest krytycznym czynnikiem.