Sprawozdanie - Metody numeryczne i optymailzacja

Jakub Andryszczak 259519,
Jakub Żak 244255,
Maciej Cierpisz 249163

Spis treści

1 Zadanie nr. 1

Rozwiązać ręcznie i komputerowo metodą eliminacji Gaussa poniższy układ równań liniowych. Znaleźć elementy podstawowe (pivots).

$$\begin{cases} 2u - v = 0 \\ -u + 2v - w = 0 \\ -v + 2w - z = 0 \\ -w + 2z = 5 \end{cases} . \tag{1}$$

Do wykonania tego zadania rozpisano lewą stronę jako macierz 4x4 oraz wektor wynikowy 4x1. Zostały one połączone dzięki czemu otrzymano macierz rozszerzoną 4x5. Poniżej przedstawiono kolejno sposób wyliczania macierzy stosując do tego metodę eliminacji Gaussa.

$$\left[\begin{array}{cccc|c} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & 5 \end{array}\right] \begin{array}{l} \\ \cdot\frac{1}{2} \\ + \\ \\ \end{array}$$

Elementem podstawowym dla pierwszej kolumny znajduje się w pierwszej kolumnie i ma wartość 2

$$\left[\begin{array}{cccc|c} 2 & -1 & 0 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & 5 \end{array}\right] \begin{array}{l} \\ \cdot\frac{2}{3} \\ + \\ \\ \end{array}$$

$$\left[\begin{array}{cccc|c} 2 & -1 & 0 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & 0 & 0 \\ 0 & 0 & \frac{4}{3} & -1 & 0 \\ 0 & 0 & -1 & 2 & 5 \end{array}\right] \begin{array}{l} \\ \\ \cdot\frac{3}{4} \\ + \end{array}$$

$$\left[\begin{array}{cccc|c} 2 & -1 & 0 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & 0 & 0 \\ 0 & 0 & \frac{4}{3} & -1 & 0 \\ 0 & 0 & 0 & \frac{5}{4} & 5 \end{array}\right] \begin{array}{l} \\ \\ \\ \cdot\left(-\frac{4}{5}\right) \end{array} \begin{array}{l} \\ \\ \\ \longleftarrow + \end{array}$$

Kolejnymi elementami podstawowymi są liczby 1.5 , 1.(3) i 1.25

$$\left[\begin{array}{cccc|c} 2 & -1 & 0 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & 0 & 0 \\ 0 & 0 & \frac{4}{3} & 0 & 4 \\ 0 & 0 & 0 & 1 & 4 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 2 & -1 & 0 & 0 & 0 \\ 0 & \frac{3}{2} & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 4 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 2 & 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 4 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 4 \end{array}\right]$$

Napisano kod w języku Python, który wykonuje eliminację Gaussa bez wyboru elementu podstawowego. Kod znajduje się poniżej.

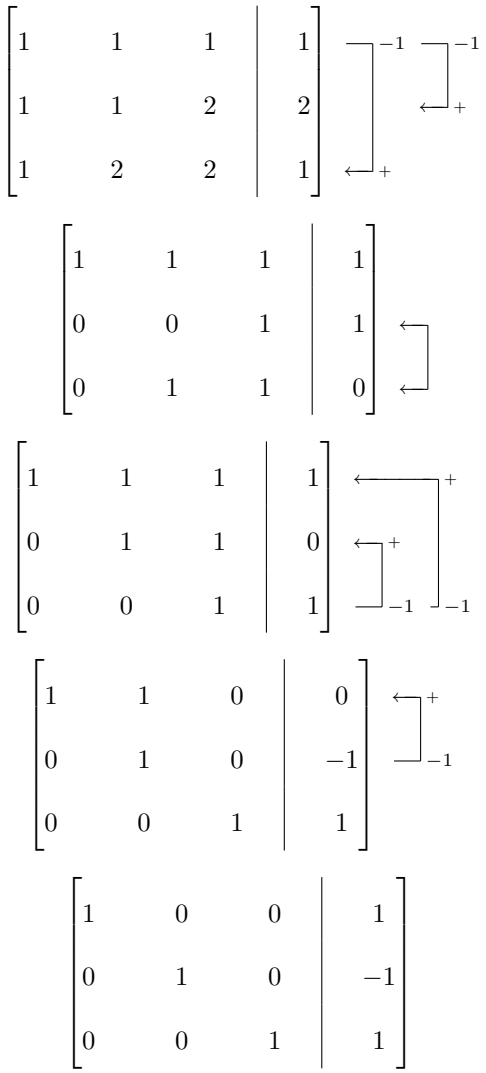
```python
import numpy as np

def gaussian_elimination(matrix, vector):
    n = len(vector)

    # Tworzenie rozszerzonej macierzy [matrix | vector]
    augmented_matrix = np.hstack((matrix, vector))

    # Eliminacja Gaussa
    for i in range(n):
        # Sprawdzanie, czy na przekątnej jest 0
        if augmented_matrix[i, i] == 0:
            raise ValueError("Zero na przekątnej. Metoda eliminacji
                Gaussa nie może ćby kontynuowana.")

        # Eliminacja współczynników pod przekątną
        for j in range(i + 1, n):
            ratio = augmented_matrix[j, i] / augmented_matrix[i, i]
            augmented_matrix[j, i:] = augmented_matrix[j, i:] - ratio *
                augmented_matrix[i, i:]

    # Rozwiązanie łukadu równań z wstecznym podstawianiem
    for i in range(n - 1, -1, -1):
        augmented_matrix[i, :] = augmented_matrix[i, :] /
            augmented_matrix[i, i]
        for j in range(i - 1, -1, -1):
            augmented_matrix[j, :] = augmented_matrix[j, :] -
                augmented_matrix[j, i] * augmented_matrix[i, :]

    return augmented_matrix
```

2 Zadanie nr. 2

Rozwiązać poniższy układ równań liniowych metodą eliminacji Gaussa z częściowym wyborem element podstawowego:

$$\begin{cases} x_1 + x_2 + x_3 = 1 \\ x_1 + x_2 + 2x_3 = 2 \\ x_1 + 2x_2 + 2x_3 = 1 \end{cases} . \tag{2}$$

Wyjaśnij dlaczego elimin. Gaussa bez wyboru elementu podstawowego nie działa poprawnie.



Wyjaśnienie dlaczego eliminacja Gaussa bez wyboru elementu podstawowego przedstawiona jest na przykładzie następnego zadania. Kod różni się w małym stopniu od tego zaimplementowanego w poprzednim zadaniu. Zmiana polega na dodaniu linijek odpowiadających za wybór elementu. Przedstawiona została poniżej.

(LINIJKI WYBIERAJĄCE ELEMENT)

---

3 Zadanie nr. 3

Rozwiązać ręcznie układ równań liniowych:

$$\begin{cases} 0,0001x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases} . \tag{3}$$

przy pomocy eliminacji Gaussa z i bez wyboru elementu podstawowego, zaokrąglając wyniki obliczeń do trzech znaczących cyfr.

Początkowo rozwiązano układ nie wybierając elementu podstawowego.

$$\left[\begin{array}{cc|c} -10^{-4} & 1 & 1 \\ 1 & 1 & 2 \end{array}\right] \begin{array}{l} \\ \longleftarrow + \end{array} \quad \begin{array}{l} \longleftarrow 10^4 \\ \\ \end{array}$$

$$\left[\begin{array}{cc|c} -10^{-4} & 1 & 1 \\ 0 & 10^4 & 10^4 \end{array}\right] \begin{array}{l} \mid \cdot \left(-10^4\right) \\ \mid \cdot \left(10^{-4}\right) \end{array} \quad \begin{array}{l} \longleftarrow + \\ \longleftarrow -1 \end{array}$$

Podczas sumowania wiersza drugiego zaokrąglono wyniki do trzech liczb znaczących.

$$\left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 1 \end{array}\right]$$

Po przeliczeniu okazało się że wynik jest niepoprawny. Błąd powstaje w drugim równaniu. Podstawiając wartości, które wyszły powstaje równanie:

$$1 = 2 \tag{4}$$

Co jest sprzeczne z prawdą.

W drugim przypadku wybierano element podstawowy co pozwoliło na otrzymanie innego wyniku.

$$\left[\begin{array}{cc|c} -10^{-4} & 1 & 1 \\ 1 & 1 & 2 \end{array}\right] \begin{array}{l} \longleftarrow \\ \longleftarrow \end{array}$$

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ -10^{-4} & 1 & 1 \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 1 & 1 \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right]$$

Po podstawieniu wartości, które wyszły okazały się one poprawne dla tego układu równań.

4 Zadanie nr. 4

Rozwiązać komputerowo układ równań liniowych:

$$\begin{cases} 0,835x_1 + 0,667x_2 = 0,168 \\ 0,333x_1 + 0,266x_2 = 0,067 \end{cases} . \tag{5}$$

Następnie, proszę zmienić (zaburzyć) wartość elementu 2 b z 0.067 na 0.066 i porównać rozwiązanie z uzyskanym dla układu niezaburzonego. Wyjaśnij dlaczego nastąpiła tak znacząca zmiana rozwiązania posługując się wskaźnikiem uwarunkowania macierzy systemowej.

Do wykonania tego zadania wykorzystano algorytm eliminacji Gaussa napisany dla zadania drugiego, podstawiając odpowiednie wartości. Dla pierwszego przypadku wektor wyjściowy miał wartości:

$$x = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{6}$$

W momencie wymiany wartości elementu b wektor wyjściowy prezentuje się następująco:

$$x = \begin{bmatrix} -665,(9) \\ 833,(9) \end{bmatrix} \tag{7}$$

Ta dość spora zmiana jest związana ze wskaźnikiem uwarunkowania macierzy, który dla obu przypadków jest dosyć wysoki. Oznacza to, że problem jest źle uwarunkowany. Tego typu zagadnienia nie nadają się do numerycznego rozwiązania, ponieważ już sam błąd wynikający z numerycznej reprezentacji liczb wprowadza nieproporcjonalnie duży błąd w odpowiedzi.

5 Zadanie nr. 5

Zaimplementuj dowolny algorytm do faktoryzacji LU i zastosuj do macierzy:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{8}$$

Wyznacz det(A) na podstawie faktorów. Znajdź rozwiązanie dla układu równań Ax b dla b = $[1...1]^T$, posługując się otrzymanymi faktorami.

Zdecydowano się na wykorzystanie algorytm Crout'a. Poniższa implementacja algorytmu na zadanej macierzy polegająca na zastąpieniu macierzy A mnożeniem macierzy dolno i górno trójkątnej o przekątnej składającej się z jedynek. Następnie obliczane są elementy macierzy L i U za pomocą równań bazujących na mnożeniu macierzy L i U oraz przyrownaniu wyników mnożeń do odpowiednich elementów macierzy A.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [L] * [U]$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} 1 & U_{12} & U_{13} & U_{14} \\ 0 & 1 & U_{23} & U_{24} \\ 0 & 0 & 1 & U_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$1 = [L_{11} \; 0 \; 0 \; 0] * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad L_{11} = 1$$

$$2 = [L_{11} \; 0 \; 0 \; 0] * \begin{bmatrix} U_{12} \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad U_{12} = 2$$

$$3 = [L_{11} \; 0 \; 0 \; 0] * \begin{bmatrix} U_{13} \\ U_{23} \\ 1 \\ 0 \end{bmatrix} \quad U_{13} = 3$$

$$3 = U_{13} = 0 + 0 + 0$$

$$4 = [L_{11} \; 0 \; 0 \; 0] * \begin{bmatrix} U_{14} \\ U_{24} \\ U_{34} \\ 1 \end{bmatrix}$$

$$U_{14} = 4$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & U_{23} & U_{24} \\ 0 & 0 & 1 & U_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$-1 = \begin{bmatrix} L_{21} & L_{22} & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad L_{21} = -1$$

$$1 = \begin{bmatrix} -1 & L_{22} & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad 1 = -2 + L_{22} + 0 + 0$$

$$L_{22} = 3$$

$$2 = \begin{bmatrix} -1 & 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ u_{23} \\ 1 \\ 0 \end{bmatrix} \quad 2 = -3 + 3u_{23} + 0 + 0$$

$$5 = 3u_{23}$$

$$u_{23} = \frac{5}{3}$$

$$1 = \begin{bmatrix} -1 & 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ u_{24} \\ u_{34} \\ 1 \end{bmatrix} \quad 1 = -4 + 3u_{24} + 0 + 0$$

$$5 = 3u_{24}$$

$$u_{24} = \frac{5}{3}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 3 & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 5/3 & 5/3 \\ 0 & 0 & 1 & U_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$0 = \begin{bmatrix} L_{31} & L_{32} & L_{33} & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad L_{31} = 0$$

$$2 = \begin{bmatrix} 0 & L_{32} & L_{33} & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad L_{32} = 2$$

$$1 = \begin{bmatrix} 0 & 2 & L_{33} & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 5/3 \\ 1 \\ 0 \end{bmatrix} \quad 1 = \frac{10}{3} = L_{33}$$

$$L_{33} = -\frac{7}{3}$$

$$3 = \begin{bmatrix} 0 & 2 & -\frac{7}{3} & 0 \end{bmatrix} \begin{bmatrix} 4 \\ \frac{5}{3} \\ u_{34} \\ 1 \end{bmatrix}.$$

$$\frac{7}{3} u_{34} = -3 + \frac{10}{3} \qquad \frac{7}{3} u_{34} = \frac{1}{3}$$

$$u_{34} = \frac{1}{7}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 3 & 0 & 0 \\ 0 & 2 & -\frac{7}{3} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & \frac{5}{3} & \frac{5}{3} \\ 0 & 0 & 1 & \frac{1}{7} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$0 = \begin{bmatrix} L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad L_{41} = 0$$

$$0 = \begin{bmatrix} 0 & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad L_{42} = 0$$

$$1 = \begin{bmatrix} 0 & 0 & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} 3 \\ \frac{5}{3} \\ 1 \\ 0 \end{bmatrix} \qquad L_{43} = 1$$

$$1 = \begin{bmatrix} 0 & 0 & 1 & L_{44} \end{bmatrix} \begin{bmatrix} 4 \\ \frac{5}{3} \\ \frac{1}{7} \\ 1 \end{bmatrix}$$

$$1 = \frac{1}{7} + L_{44} \qquad L_{44} = \frac{6}{7}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 3 & 0 & 0 \\ 0 & 2 & -\frac{7}{3} & 0 \\ 0 & 0 & 1 & \frac{6}{7} \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & \frac{5}{3} & \frac{5}{3} \\ 0 & 0 & 1 & \frac{1}{7} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Poniżej implementacja programowa w języku Python:

```
  import numpy as np


def LU(A):
    n = len(A)
    L = np.zeros((n, n))
    U = np.zeros((n, n))

    for i in range(n):
        # Przekątna macierzy L wynosi 1
        L[i][i] = 1

        for j in range(i, n):
            #  Obliczanie elementów macierzy górnej trójkątnej U
            sum = 0
            for k in range(i):
                sum += L[i][k] * U[k][j]
            U[i][j] = A[i][j] - sum

        for j in range(i+1, n):
            #  Oblicznie elementów macierzy dolnej trójkątnej L
            sum = 0
            for k in range(i):
                sum += L[j][k] * U[k][i]
            L[j][i] = (A[j][i] - sum) / U[i][i]

    return L, U
```

6 Zadanie nr. 6

Przekształć ręcznie poniższą macierz do postaci RREF, wyznacz rank (A) oraz wskaż kolumny odpowiadające zmiennym podstawowym i wolnym.

$$A = \begin{bmatrix} 1 & 2 & 2 & 3 & 1 \\ 2 & 4 & 4 & 6 & 2 \\ 3 & 6 & 6 & 9 & 6 \\ 1 & 2 & 4 & 5 & 3 \end{bmatrix} \tag{9}$$

Etapy rozwiązania macierzy przedstawiono poniżej:

$$\begin{bmatrix} 1 & 2 & 2 & 3 & 1 \\ 2 & 4 & 4 & 6 & 2 \\ 3 & 6 & 6 & 9 & 6 \\ 1 & 2 & 4 & 5 & 3 \end{bmatrix} \begin{matrix} \\ \leftarrow +(-2) \\ \leftarrow +(-3) \\ \leftarrow +(-1) \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 2 & 2 & 2 \end{bmatrix} \begin{matrix} \\ \leftarrow \\ \\ \leftarrow \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 2 & 3 & 1 \\ 0 & 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \\ \mid \cdot \left(\frac{1}{2}\right) \\ \mid \cdot \left(\frac{1}{3}\right) \\ \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 2 & 3 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow + & \leftarrow + & \\ -2 & & \leftarrow + \\ & & -1 \\ & & \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Wartość rank(A) = 3, zmienne podstawowe (A_{*1}, A_{*3}, A_{*5}), zmienne wolne (A_{*2}, A_{*4})

7 Zadanie nr. 7

Zaimplementuj dowolną metodę ortogonalizacji macierzy i wyznacz faktoryzację QR macierzy:

$$A = \begin{bmatrix} -2 & 1 & 2 & 1 \\ 2 & -1 & 2 & 1 \\ 2 & 3 & -4 & 5 \\ 2 & 3 & 0 & -1 \end{bmatrix} \tag{10}$$

Zaimplementuj dowolną metodę ortogonalizacji macierzy i wyznacz faktoryzację QR macierzy:

$$A = \begin{bmatrix} -2 & 1 & 2 & 1 \\ 2 & -1 & 2 & 1 \\ 2 & 3 & -4 & 5 \\ 2 & 3 & 0 & -1 \end{bmatrix} \tag{11}$$

Do ortogonalizacji macierzy i stworzenia macierzy Q wykorzytano metodę Grama-Schmidt'a. Początkowo macierz podzielono na 4 osobne wektory wzdłuż kolumn. Następnie zaczęto wyliczać wartość u_1. Dla tego przypadku.

$$u_1 = a_1 \tag{12}$$

Następnie wyliczono normę dla tego wektora.

$$||u_1|| = \sqrt{{a_{11}}^2 + {a_{12}}^2 + {a_{13}}^2 + {a_{14}}^2} = 4 \tag{13}$$

Po wyliczeniu wartości $u_1 oraz ||u_1||$ wyliczono pierwszą kolumnę macierzy Q ze wzoru.

$$q_1 = \frac{u_1}{||u_1||} = \begin{bmatrix} -0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \tag{14}$$

Kolejne kolumny zostały przeliczane stosując wzoru jak w poprzednim przykładzie. Następnie zsumowano je tworząc macierz Q 4x4.

$$Q = \begin{bmatrix} -0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & -0.5 \end{bmatrix} \tag{15}$$

W przypadku wyliczenia macierzy R możemy ją wyliczyć przekształcając wzór

$$A = QR / * Q^T \tag{16}$$

$$AQ^T = QRQ^T \tag{17}$$

Z własności macierzy ortogonalnych wynika następująca zależność

$$Q^T Q = I \tag{18}$$

I jest macierzą jednostkową, która jest neutralna dla mnożenia macierzy co za tym idzie nasz wzór wygląda następująco

$$AQ^T = R \tag{19}$$

Podstawiając do tego wzoru wartości otrzymujemy że

$$R = \begin{bmatrix} -0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & -0.5 \end{bmatrix} * \begin{bmatrix} -2 & 1 & 2 & 1 \\ 2 & -1 & 2 & 1 \\ 2 & 3 & -4 & 5 \\ 2 & 3 & 0 & -1 \end{bmatrix} \tag{20}$$

$$R = \begin{bmatrix} 4 & 2 & -2 & 2 \\ 0 & 4 & -2 & 2 \\ 0 & 0 & 4 & -2 \\ 0 & 0 & 0 & 4 \end{bmatrix} \tag{21}$$

Stworzono kod w Python'ie, który wykonuje wyżej wytłumaczoną Faktoryzację QR.

```
  import numpy as np

def gram_schmidt(A):
    m, n = A.shape
    Q = np.zeros((m, n))
    R = np.zeros((n, n))

    for i in range(n):
        #  Obliczenie ortogonalnego wektora q_i
        q = A[:, i]
        for j in range(i):
            # Odejmowanie rzutu q na każde q_j
            R[j, i] = np.dot(Q[:, j], A[:, i])
            q = q - R[j, i] * Q[:, j]
        # Obliczenie świelkoci q (która jest przekątną R)
        R[i, i] = np.linalg.norm(q)
        # Unikanie dzielenia przez zero
        if R[i, i] == 0:
            raise ValueError("Matrix A has linearly dependent vectors")
        # Step 3: Normalize q to get the orthonormal vector q_i
        Q[:, i] = q / R[i, i]
    return Q, R
```

8 Zadanie nr. 8

Niech Ax = b , gdzie , symbol oznacza iloczyn Kroneckera, jest macierzą jednostkową, C jest macierzą losową wygenerowaną z rozkładu równomiernego (rand), M = 200, N = 30, oraz (rozkład normalny - randn). Rozwiąż układ równań różnymi metodami bezpośrednimi (eliminacja Gaussa, faktoryzacja LU, faktoryzacja QR) i porównaj metody ze względu na czas wykonywania się algorytmów i błąd aproksymacji rozwiązania.

Badaniu poddano własne implementacje algorytmów oraz metody dostępne w bibliotece scipy. Poniej zestawiono wyniki badań w postaci zmierzonych czasów wykonania poszczegolnych algorytmów oraz obliczone błędy aproksymacji.

Rozmiar macierzy 6000x6000	
Algorytm	Czas [s]
Gauss algorytm własny	205,1
LU algorytm własny	157,51
LU biblioteka scipy	1,4
QR algorytm własny	101,8
QR biblioteka scipy	9,1

Tabela. 1 Wykaz czasów wykonania algorytmów

Błędy aproksymacji	
Gauss algorytm własny	1,83
LU algorytm własny	3,38
QR algorytm własny	4697724

Tabela. 2 Wykaz błędów aproksymacji algorytmów

9 Zadanie nr. 9

Znajdź prądy gałęziowe zakładając, że wszystkie rezystancje wynoszą 20 Ohm.

Równania napięciowe

$$\begin{cases} 2I_3 - 2I_4 = 1 \\ -I_3 - I_5 = 1 \\ 2I_6 - 2I_5 = 1 \end{cases} . \tag{22}$$

Równania prądowe

$$\begin{cases} I_2 + I_3 - I_5 = 0 \\ I_5 + I_6 - I_1 = 0 \\ I_4 - I_2 - I_6 = 0 \\ I_1 - I_3 - I_4 = 0 \end{cases} . \tag{23}$$

Równania prądowe

$$\begin{cases} I_1 - I_3 - I_4 = 0 \\ I_2 + I_3 - I_5 = 0 \\ -I_3 - I_5 = 1 \\ 2I_3 - 2I_4 = 1 \\ -2I_5 + 2I_6 = 1 \\ -I_2 + I_4 - I_6 = 0 \end{cases} . \tag{24}$$

Kod obliczający wartości szukanych prądów

```
import numpy as np
from numpy.linalg import solve

#Inicjalizacja macierzy A i wektora b
A = np.array([ [1, 0, -1, -1, 0, 0],
               [0, 1, 1, 0, -1, 0],
               [0, 0, -1, 0, -1, 0],
               [0, 0, 2, -2, 0, 0],
               [0, 0, 0, 0, -2, 2],
               [0, -1, 0, 1, 0, -1],])

b = np.array([0, 0, -1, -1, -1, 0],dtype=float)

#Obliczenie wyników metodą eliminacji Gaussa z pivotingiem
x = solve(A,b)
```

Wyniki obliczeń

$$\begin{cases} I_1 = 1 \\ I_2 = 0.5 \\ I_3 = 0.25 \\ I_4 = 0.75 \\ I_5 = 0.75 \\ I_6 = 0.25 \end{cases} . \tag{25}$$

10 Wnioski

Laboratoria miały za zadanie przedstawienie metod rozwiązywania macierzy takich jak: Eliminacja Gaussa, Faktoryzacja LU i QR. Po przeprowadzeniu wcześniej określonych zadań Eliminacja Gaussa jest prostą metodą przekształcania macierzy i najdokładniejsza w porównaniu do pozostałych dwóch, natomiast porównując czas wykonania szczególnie dla macierzy o dużych rozmiarach pracuje najwolniej. Faktoryzacja LU polega na przedstawieniu macierzy w postaci wyniku mnożenia macierzy dolnotrójkątnej L z macierzą górnotrójkątną U. Jest ona szybsza w porównaniu do eliminacji Gaussa, ale posiada większy błąd aproksymacji. W przypadku faktoryzacji QR polega ona na wyznaczeniu macierzy ortogonalnej do niej Q, a następnie przekształcając wzór $A = QR$ Wyliczamy macierz R. Jest to najszybszy sposób przekształcania, aczkolwiek posiada on największy błąd.