Sprawozdanie 2 - Eliminacja Gaussa i LU faktoryzacja

Wojciech Smolarczyk, Wiktoria Zalińska

LU faktoryzacja

Algorytm LU faktoryzacji polega na rozłożeniu macierzy kwadratowej na iloczyn dwóch macierzy:

$$A = LU$$

gdzie:

- **L** – macierz dolnotrójkątna (z jedynkami na przekątnej),
- **U** – macierz górnotrójkątna.

Główna zaleta LU faktoryzacji to możliwość szybszego rozwiązania układów równań liniowych.
Zamiast rozwiązywać **Ax = b** bezpośrednio, dzielimy problem na dwa prostsze układy:

$$LC = b$$
$$Ux = c$$

Najpierw rozwiązujemy **Lc = b**, a następnie **Ux = c** metodą podstawiania.

Kroki LU faktoryzacji

1. **Inicjalizacja**:
    - Tworzymy macierz dolnotrójkątną **L** jako macierz jednostkową.
    - Kopiujemy **A** do **U**, ponieważ będziemy ją modyfikować.
2. **Eliminacja Gaussa**:
    - Dla każdej kolumny eliminujemy elementy pod przekątną, zapisując współczynniki eliminacji w macierzy **L**.
3. **Wynik**:
    - **U** staje się macierzą górnotrójkątną.
    - **L** zawiera współczynniki eliminacji poniżej przekątnej oraz jedynki na przekątnej.

LU faktoryzacja z pivotingiem

LU faktoryzacja z pivotingiem jest ulepszoną wersją LU, która poprawia stabilność numeryczną i pozwala uniknąć dzielenia przez małe wartości (lub zero) na przekątnej.
Wprowadza dodatkową macierz permutacji **P**, która zapisuje zamiany wierszy.

Dzięki temu zamiast rozwiązywać:

$$Ax = B$$

rozwiązujemy:

$$PAx = Pb$$

gdzie `P` rejestruje kolejność zamian wierszy.

Kroki LU faktoryzacji z pivotingiem

1. **Inicjalizacja**:
   - Tworzymy **P** jako macierz jednostkową.
   - Tworzymy **L** jako macierz jednostkową.
   - Kopiujemy **A** do **U**.
2. **Pivoting – zamiana wierszy**:
   - W każdej iteracji wybieramy największy element w kolumnie (od aktualnego wiersza w dół).
   - Zamieniamy odpowiednie wiersze macierzy U, P i L (w L zamieniamy tylko wcześniejsze kolumny).
3. **Eliminacja Gaussa** (tak jak w zwykłym LU).

Kod implementacji LU z i bez pivotingu

```
def lu_decomposition(A, pivoting=False):
    n = A.shape[0]
    # Macierz permutacyjna – początkowo jednostkowa
    P = np.eye(n)
    # Macierz dolnotrójkątna – początkowo jednostkowa
    L = np.eye(n)
    # Kopia A, bo będziemy modyfikować U
    U = A.copy()

    for i in range(n):
        if pivoting:
            # Znalezienie indeksu największego elementu w aktualnej kolumnie
poniżej przekątnej
            max_index = np.argmax(abs(U[i:, i])) + i

            # Zamiana wierszy
            if max_index != i:
                U[[i, max_index]] = U[[max_index, i]]
                P[[i, max_index]] = P[[max_index, i]]
                if i > 0:  # Zamiana tylko wcześniejszych kolumn w L
                    L[[i, max_index], :i] = L[[max_index, i], :i]

        for j in range(i+1, n):
            # Obliczenie współczynnika eliminacji
            L[j, i] = U[j, i] / U[i, i]
            # Aktualizacja macierzy U
```

```
            U[j, i:] -= L[j, i] * U[i, i:]

    if pivoting:
        return P, L, U
    else:
        return L, U
```