Program 2 – Rozkład LU, Eliminacja Gaussa

Dominik Tomalczyk
Piotr Van-Selow

Rozmiar macierzy to suma miesiąca i dnia urodzenia, stąd:
Rozmiar macierzy = 4 + 2 = 6
Język implementacji: Python

1. Pseudokod algorytmu eliminacji Gaussa generujący 1 na przekątnej :

Algorithm 1: Gauss elimination algorithm

```
for irow=1,N //loop through rows
for icol=irow+1,N // row scalling
A(irow,icol)=A(irow,icol)/A(irow,irow)
end loop
b(irow)=b(irow)/A(irow,irow)
A(irow,irow)=1.0
for irow2=irow+1,N //row subtractions
for icol=irow2+1,N
A(irow2,icol) = A(irow2,icol) - A(irow2,irow) * A(irow,icol)
end loop
b(irow2)=b(irow2)-A(irow2,irow)*b(irow2)
A(irow2,irow)=0
end loop
end loop
```

Rys. 1. Pseudokod eliminacji Gaussa. Źródło: https://pre-epodreczniki.open.agh.edu.pl/tiki-index.php?page=Gaussian+elimination+algorithm

Rozwiazanie back() i forward() również opisane w tym dokumencie:
https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_LU.html

Psuedocode for forward substitution

Steps	
1.	Set $y_1 = b_1 / l_{11}$
2.	For $i = 2, 3, \ldots n$ do Step 3
3.	$y_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right) \Big/ l_{ii}$

Pseudocode Back substitution

Steps	
1.	Set $x_n = y_n / u_{nn}$
2.	For $i = n-1, n-2, \ldots, 1$ do Step 3.
3.	$x_i = \left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \Big/ u_{ii}$

2. Kod algorytmu eliminacji Gaussa generujący 1 na przekątnej napisany dla swojego rozmiaru macierzy w swoim ulubionym języku programowania:

```python
def gauss_elimination(A: List[List[int]], b: List[int]) -> Tuple[List[List[int]], List[int]]:
    N = len(A)
    for irow in range(N):  # loop through rows
        for icol in range(irow + 1, N):  # row scaling
            if A[irow][irow] == 0:
                raise ValueError("Zero on diagonal. Pivoting needed!")
            A[irow][icol] = A[irow][icol] / A[irow][irow]
        b[irow] = b[irow] / A[irow][irow]
        A[irow][irow] = 1.0
        for irow2 in range(irow + 1, N):  # row subtractions
            m = A[irow2][irow]
            for icol in range(irow + 1, N):
                A[irow2][icol] = A[irow2][icol] - m * A[irow][icol]
            b[irow2] = b[irow2] - m * b[irow]
            A[irow2][irow] = 0

    return A, b
```

3. Losujemy macierz gęsta z wartościami losowymi oraz wektor prawej strony. Rozwiązujemy układ równań swoim programem oraz Octave (lub MATLABem) x = A\b i porównujemy wynik norm(x1-x2,2)

Wylosowana macierz A i wektor b:

```
A matrix:
[9, 4, 5, 9, 4, 0]
[3, 9, 6, 5, 6, 2]
[1, 3, 10, 2, 3, 9]
[4, 4, 3, 0, 9, 9]
[6, 0, 4, 3, 0, 0]
[7, 10, 0, 0, 10, 9]
b vector:
[5, 3, 9, 10, 5, 9]
```

Rozwiązanie układu równań AX = b dla naszego programu:

```
Solution for AX = b
[0.7096370955835308, -0.31494040178351745, 0.5207141943167615, -0.4468931169227438, 0.32287370690079464, 0.43924525330472336]
```

Rozwiązanie układu Równań AX = b, z wykorzystaniem MATLAB:

```
matlab solution
[0.7096370955835307, -0.3149404017835176, 0.5207141943167616, -0.4468931169227438, 0.3228737069007946, 0.4392452533047234]
```

Norma L2 x1 – x2 porównująca wynik w Matlabie I naszej impelemtacji:

```
Norm x1 - x2
2.4196749845665633e-16
```

4. Pseudokod algorytmu eliminacji Gaussa z Pivotingiem:

Gaussian Elimination with Partial Pivoting on A:

For $k = 1, \ldots, n-1$

Find $i_k \geq k$ such that $|a_{i_k k}| = \max_{k \leq i \leq n} |a_{ik}|$.

If $i_k > k$, interchange $a_{kj} \leftrightarrow a_{i_k j}$ for $j = k, \ldots, n$.

For $i = k+1, \ldots, n$

$a_{ik} \leftarrow -a_{ik}/a_{kk}$

For $j = k+1, \ldots, n$

$a_{ij} \leftarrow a_{ij} + a_{ik}a_{kj}$

Row Operations on b:

For $k = 1, \ldots, n-1$

If $i_k > k$, interchange $b_k \leftrightarrow b_{i_k}$.

For $i = k+1, \ldots, n$

$b_i \leftarrow b_i + a_{ik}b_k$

Źródło: https://users.wpi.edu/~walker/MA514/HANDOUTS/gaussian_elim.pdf

5. Kod algorytmu eliminacji Gaussa z pivotinigiem w Python:

```python
def gaussian_elimination_with_pivoting(A: List[List[int]], b: List[int]) -> Tuple[List[List[int]], List[int]]:
    n = len(A)

    for i in range(n):
        # Pivoting
        max_index = i
        for j in range(i + 1, n):
            if abs(A[j][i]) > abs(A[max_index][i]):
                max_index = j
        A[i], A[max_index] = A[max_index], A[i]
        b[i], b[max_index] = b[max_index], b[i]

        # elimination
        for j in range(i + 1, n):
            ratio = A[j][i] / A[i][i]
            for k in range(i, n):
                A[j][k] -= ratio * A[i][k]
            b[j] -= ratio * b[i]
    return A, b
```

6. Losujemy macierz gęsta z wartościami losowymi oraz wektor prawej strony. Rozwiazujemy układ i porównujemy wynik z Matlabem. Wylosowana macierz A i wektor b:

```
A matrix:
[0, 10, 5, 4, 6, 1]
[10, 10, 10, 1, 0, 8]
[5, 7, 10, 0, 9, 2]
[8, 5, 6, 4, 1, 5]
[4, 2, 10, 1, 4, 7]
[2, 7, 10, 1, 3, 7]
b vector:
[0, 6, 9, 1, 8, 0]
```

Rozwiązanie układu równań AX = b dla naszego programu:

```
Solution for AX = b
[2.357771787960469, 1.271967654986523, -12.300089847259665, -5.334950584007189, 9.644294699011684, 12.25480682839174]
```

Rozwiązanie układu Równań AX = b, z wykorzystaniem MATLAB:

```
matlab solution
[2.357771787960469, 1.2719676549865226, -12.300089847259665, -5.3349505840071885, 9.644294699011684, 12.25480682839174]
```

Norma L2 x1 – x2 porównująca wynik w Matlabie I naszej impelemtacji:

```
Norm x1 - x2
9.930136612989092e-16
```

7. Pseudokod faktoryzacji LU:

Steps	
1.	Initialize $\mathbf{L}$ to an identity matrix, $\mathbf{I}$ of dimension $n \times n$ and $\mathbf{U} = \mathbf{A}$.
2.	For $i = 1, \ldots, n$ do Step 3
3.	For $j = i + 1, \ldots, n$ do Steps 4-5
4.	Set $l_{ji} = u_{ji}/u_{ii}$
5.	Perform $U_j = (U_j - l_{ji}U_i)$ (where U_i, U_j represent the i and j rows of the matrix $\mathbf{U}$, respectively)

Źródło: https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_LU.html

8. Kod faktoryzacji LU w Pythonie:

```python
def LU_decomposition(A: List[int]) -> Tuple[List[List[int]], List[List[int]]]:
    """ Compute the L and U matrices from A matrix """
    n = len(A)
    L = [[0.0] * n for _ in range(n)]
    U = [row[:] for row in A]  # U = A (kopia macierzy A)

    for i in range(n):
        L[i][i] = 1.0  # Inicjalizacja elementów diagonalnych macierzy L jako 1

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

            for k in range(i, n):
                U[j][k] -= L[j][i] * U[i][k]  # Modyfikacja j-tego wiersza macierzy U

    return L, U
```

9. Losujemy macierz A, wektor b i obliczamy macierze U oraz L:
   Wylosowana macierz A oraz wektor b:

```
A matrix:
[3, 3, 9, 4, 9, 2]
[5, 2, 9, 4, 9, 10]
[0, 1, 10, 2, 7, 8]
[10, 6, 10, 1, 4, 4]
[2, 7, 2, 7, 2, 0]
[4, 9, 10, 8, 6, 0]
b vector:
[9, 10, 2, 2, 1, 0]
```

Z algorytmu otrzymujemy macierz U:

```
U matrix:
[3, 3, 9, 4, 9, 2]
[0.0, -3.0, -6.0, -2.666666666666667, -6.0, 6.666666666666666]
[0.0, 0.0, 8.0, 1.1111111111111112, 5.0, 10.222222222222221]
[0.0, 0.0, 0.0, -7.111111111111112, -10.5, 3.7777777777777786]
[0.0, 0.0, 0.0, 0.0, -7.95703125, 28.640624999999996]
[0.0, 0.0, 0.0, 1.3877787807814457e-17, 0.0, -6.285714285714278]
```

Oraz macierz L:

```
L matrix:
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.6666666666666667, 1.0, 0.0, 0.0, 0.0, 0.0]
[0.0, -0.3333333333333333, 1.0, 0.0, 0.0, 0.0]
[3.3333333333333335, 1.3333333333333333, -1.5, 1.0, 0.0, 0.0]
[0.6666666666666666, -1.6666666666666667, -1.75, -0.25781249999999994, 1.0, 0.0]
[1.3333333333333333, -1.6666666666666667, -1.5, 0.0156250000000001, 1.0476190476190474, 1.0]
```

Rozwiązując kolejno równania:

L, U = A
Y = forward(L, b)
X = back(U, y)

Źródło: https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_LU.html

Otrzymujemy rozwiązanie X:

```
Solution of AX = b
[0.44501718213058455, 0.5567010309278357, -1.5601374570446747, -0.8316151202749154, 2.5773195876288675, 0.08333333333333337]
```

Oraz rozwiązanie w Matlabie:

```
Matlab result
[0.4450171821305839, 0.5567010309278353, -1.5601374570446735, -0.8316151202749145, 2.577319587628866, 0.08333333333333333]
```

Norma L2 x1 – x2 porównująca wynik w Matlabie I naszej impelemtacji:

```
Norm x1 - x2
2.1446009652593084e-15
```

10.Pseudokod faktoryzacji LU z pivotingiem

Steps	
1.	Initialize $\mathbf{L} = \mathbf{P} = \mathbf{I}$ of dimension $n \times n$ and $\mathbf{U} = \mathbf{A}$
2.	For $i = 1, \ldots, n$ do Steps 3-4, 8
3.	Let $k = i$,
4.	While $u_{ii} = 0$, do Steps 5-7
5.	Swap row U_i with row U_{k+1}
6.	Swap row P_i with row P_{k+1}
7.	Increment k by 1.
8.	For $j = i + 1, \ldots, n$ do Steps 9-10
9.	Set $l_{ji} = u_{ji}/u_{ii}$
10.	Perform $U_j = U_j - l_{ji}U_i$ (where U_i, U_j represent the i and j rows of the matrix $\mathbf{U}$, respectively)

Źródło: https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_LU.html

11.Kod faktoryzacji LU z pivotingiem w Pythonie:

```python
def LU_decomposition_pivoting(A: List[int]) -> Tuple[List[List[int]], List[List[int]],
    """ Compute the L,U and P matrices from A matrix """
    # Get the number of rows
    n = len(A)

    # Allocate space for P, L, and U
    U = [row[:] for row in A]  # Copy of A
    L = [[0.0] * n for _ in range(n)]
    P = [[0.0] * n for _ in range(n)]
    for i in range(n):
        L[i][i] = 1.0
        P[i][i] = 1.0

    # Loop over rows
    for i in range(n):
        # Permute rows if needed
        for k in range(i, n):
            if U[i][i] != 0.0:
                break
            U[k], U[k + 1] = U[k + 1], U[k]
            P[k], P[k + 1] = P[k + 1], P[k]

        # Eliminate entries below i with row operations on U
        # and reverse the row operations to manipulate L
        for j in range(i + 1, n):
            factor = U[j][i] / U[i][i]
            L[j][i] = factor
            for k in range(i, n):
                U[j][k] -= factor * U[i][k]

    return P, L, U
```

12.Losujemy macierz gesta z warotsciami losowymi, rozwiązujemy układ i porównujemy z Matlabem:

Wylosowana macierz A i wektor b:

```
A matrix:
[0, 5, 7, 9, 0, 9]
[4, 0, 8, 7, 8, 5]
[9, 1, 5, 6, 6, 0]
[10, 1, 3, 1, 5, 9]
[5, 1, 5, 9, 4, 5]
[3, 3, 0, 9, 9, 3]
b vector:
[6, 2, 4, 8, 6, 7]
```

Macierz U:

```
U matrix:
[4, 0, 8, 7, 8, 5]
[0.0, 5.0, 7.0, 9.0, 0.0, 9.0]
[0.0, 0.0, -14.4, -11.55, -12.0, -13.05]
[0.0, 0.0, 0.0, -3.541666666666668, 0.33333333333333215, 11.375]
[0.0, 0.0, 0.0, 4.440892098500626e-16, -0.32941176470588307, 14.258823529411764]
[0.0, 0.0, 0.0, 0.0, 0.0, 548.4642857142844]
```

Macierz L:

```
L matrix:
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
[2.25, 0.2, 1.0, 0.0, 0.0, 0.0]
[2.5, 0.2, 1.2777777777777777, 1.0, 0.0, 0.0]
[1.25, 0.2, 0.4444444444444445, -1.0117647058823527, 1.0, 0.0]
[0.75, 0.6, 0.7083333333333333, -1.8441176470588232, -36.77678571428562, 1.0]
```

Macierz P:

```
P matrix:
[0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 1.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 1.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
```

Rozwiązując kolejno równania:

P, L, U = A

Y = forward(L, P*b)

X = back(U, y)

Źródło: https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_LU.html

Otrzymujemy rozwiązanie układu rownan AX = b

```
Solution of AX = b
[0.51533502637234, 0.3273425799309713, -0.5919775997916211, 0.4518460636843123, -0.11942436673829998, 0.49339063619196444]
```

Rozwiazanie układu równan AX = b w Matlabie:

```
Matlab result:
[0.5153350263723383, 0.3273425799309761, -0.5919775997916261, 0.4518460636843133, -0.1194243667382952, 0.4933906361919647]
```

Norma L2 x1 – x2 porównująca wynik w Matlabie I naszej impelemtacji:

```
L2 norm x1 - x2
8.624975342569023e-15
```