Lab 2

```
from scipy.linalg import lu, qr, eig, svd, lu_factor, lu_solve
import numpy as np
```

```
A1 = np.array([[2, 1],
               [6, 7]])

A2 = np.array([[1, 1, 1],
               [1, 1, 1],
               [1, 1, 1]])

A3 = np.array([[2, -1, 0],
               [-1, 2, -1],
               [0, -1, 2]])

A1, A2, A3
```

```
(array([[2, 1],
        [6, 7]]),
 array([[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]]),
 array([[ 2, -1,  0],
        [-1,  2, -1],
        [ 0, -1,  2]]))
```

6. Write a Python function `Ludecompose(A)` that takes as input a two-dimensional numpy array `A` and return either the LU decomposition of `A` or NaN if `A` does not have a LU decomposition. Use the function to verify the results from questions 1–3.

```
def Ludecompose(A):
  try:
    P, L, U = lu(A)
    return P, L, U
  except:
    return np.nan, np.nan, np.nan

P, L, U = Ludecompose(A1)
print(L, "\n\n", U, "\n\n", "L*U = ", np.dot(P, np.dot(L, U)), "\n\n")

P, L, U = Ludecompose(A2)
print(L, "\n\n", U, "\n\n", "L*U = ", np.dot(P, np.dot(L, U)), "\n\n")

P, L, U = Ludecompose(A3)
print(L, "\n\n", U, "\n\n", "L*U = ", np.dot(P, np.dot(L, U)))
```

```
[[1.         0.        ]
 [0.33333333 1.        ]]

 [[ 6.          7.        ]
 [ 0.         -1.33333333]]

 L*U =  [[2. 1.]
 [6. 7.]]


[[1. 0. 0.]
 [1. 1. 0.]
 [1. 0. 1.]]

 [[1. 1. 1.]
 [0. 0. 0.]
 [0. 0. 0.]]

 L*U =  [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

```
[[ 1.          0.          0.        ]
 [-0.5         1.          0.        ]
 [ 0.         -0.66666667  1.        ]]

 [[ 2.         -1.          0.        ]
 [ 0.          1.5        -1.        ]
 [ 0.          0.          1.33333333]]

 L*U =  [[ 2. -1.  0.]
 [-1.  2. -1.]
 [ 0. -1.  2.]]
```

7. Use `numpy.linalg.solve` to verfiy the result from question 5.

```
A = np.array([[2, 1],
              [6, 7]])

b = np.array([1, 1])

A, b
```

```
(array([[2, 1],
        [6, 7]]),
 array([1, 1]))
```

```
np.linalg.solve(A, b)
```

```
array([ 0.75, -0.5 ])
```

8. Use `numpy.linalg.qr` to calculate the QR decompositions of the matrices A_1, A_2, A_3

```
Q1, R1 = qr(A1)
Q2, R2 = qr(A2)
Q3, R3 = qr(A3)

print(f"Q1 = \n{Q1} \n\nR1 = \n{R1} \n\nQ*R = \n{np.dot(Q1, R1)}")
print(f"Q2 = \n{Q2} \n\nR2 = \n{R2} \n\nQ*R = \n{np.dot(Q2, R2)}")
print(f"Q3 = \n{Q3} \n\nR3 = \n{R3} \n\nQ*R = \n{np.dot(Q3, R3)}")
```

```
Q1 =
[[-0.31622777 -0.9486833 ]
 [-0.9486833   0.31622777]]

R1 =
[[-6.32455532 -6.95701085]
 [ 0.          1.26491106]]

Q*R =
[[2. 1.]
 [6. 7.]]
Q2 =
[[-0.57735027 -0.57735027 -0.57735027]
 [-0.57735027  0.78867513 -0.21132487]
 [-0.57735027 -0.21132487  0.78867513]]

R2 =
[[-1.73205081 -1.73205081 -1.73205081]
 [ 0.          0.          0.        ]
 [ 0.          0.          0.        ]]

Q*R =
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
Q3 =
[[-0.89442719 -0.35856858  0.26726124]
 [ 0.4472136  -0.71713717  0.53452248]
 [-0.          0.5976143   0.80178373]]

R3 =
[[-2.23606798  1.78885438 -0.4472136 ]
 [ 0.         -1.67332005  1.91236577]
 [ 0.          0.          1.06904497]]

Q*R =
```

```
[[ 2.00000000e+00 -1.00000000e+00 -6.98172051e-17]
 [-1.00000000e+00  2.00000000e+00 -1.00000000e+00]
 [ 0.00000000e+00 -1.00000000e+00  2.00000000e+00]]
```

9. Use `numpy.linalg.eig` to calculate the eigendecompositions of the matrices A_1, A_2, A_3

```
eigvals_A1, eigvecs_A1 = eig(A1)
eigvals_A2, eigvecs_A2 = eig(A2)
eigvals_A3, eigvecs_A3 = eig(A3)

print(f"eigvals_A1 = \n{eigvals_A1} \n\neigvecs_A1 = \n{eigvecs_A1}\n\n")
print(f"eigvals_A2 = \n{eigvals_A2} \n\neigvecs_A2 = \n{eigvecs_A2}\n\n")
print(f"eigvals_A3 = \n{eigvals_A3} \n\neigvecs_A3 = \n{eigvecs_A3}\n\n")
```

```
eigvals_A1 =
[1.+0.j 8.+0.j]

eigvecs_A1 =
[[-0.70710678 -0.16439899]
 [ 0.70710678 -0.98639392]]


eigvals_A2 =
[-2.22044605e-16+0.j  3.00000000e+00+0.j  0.00000000e+00+0.j]

eigvecs_A2 =
[[-0.81649658  0.57735027  0.        ]
 [ 0.40824829  0.57735027 -0.70710678]
 [ 0.40824829  0.57735027  0.70710678]]


eigvals_A3 =
[3.41421356+0.j 2.        +0.j 0.58578644+0.j]

eigvecs_A3 =
[[-5.00000000e-01 -7.07106781e-01  5.00000000e-01]
 [ 7.07106781e-01  4.05405432e-16  7.07106781e-01]
 [-5.00000000e-01  7.07106781e-01  5.00000000e-01]]
```

10. Use `numpy.linalg.svd` to calculate the singular value decompositions of A_1, A_2, A_3

```
U1, S1, Vt1 = svd(A1)
U2, S2, Vt2 = svd(A2)
U3, S3, Vt3 = svd(A3)

print(f"U1 = \n{U1} \n\nS1 = \n{S1} \n\nVt1 = \n{Vt1}\n\n")
print(f"U2 = \n{U2} \n\nS2 = \n{S2} \n\nVt2 = \n{Vt2}\n\n")
print(f"U3 = \n{U3} \n\nS3 = \n{S3} \n\nVt3 = \n{Vt3}\n\n")
```

```
U1 =
[[-0.21991191 -0.97551973]
 [-0.97551973  0.21991191]]

S1 =
[9.4489777  0.84665244]

Vt1 =
[[-0.66599186 -0.74595901]
 [-0.74595901  0.66599186]]


U2 =
[[-0.57735027 -0.57735027 -0.57735027]
 [-0.57735027 -0.21132487  0.78867513]
 [-0.57735027  0.78867513 -0.21132487]]

S2 =
[3. 0. 0.]

Vt2 =
[[-0.57735027 -0.57735027 -0.57735027]
 [ 0.         -0.70710678  0.70710678]
 [ 0.81649658 -0.40824829 -0.40824829]]
```

```
U3 =
[[-5.00000000e-01 -7.07106781e-01  5.00000000e-01]
 [ 7.07106781e-01 -3.88578059e-16  7.07106781e-01]
 [-5.00000000e-01  7.07106781e-01  5.00000000e-01]]

S3 =
[3.41421356 2.         0.58578644]

Vt3 =
[[-5.00000000e-01  7.07106781e-01 -5.00000000e-01]
 [-7.07106781e-01  1.11022302e-16  7.07106781e-01]
 [ 5.00000000e-01  7.07106781e-01  5.00000000e-01]]
```