In [150…

```
%run my_functions_library.ipynb                                    # Running the entire library here because
                                                                   # importing the library is not working here
```

Question 1

LU decomposition using Doolittle's condition i.e. L[i][i]=1

In [151…

```
print("The original matrix is: ")
A1,row,col = read_matrix('matrix_Q1.txt')                          # reading and printing the matrix given in the question
print_matrix(A1,row,col)

b_vector=[6,-3,-2,0]                                               # defining the vector b

A1, b_vector = partial_pivot_LU(A1, b_vector, row)                 # partial pivoting to avoid division by zero at pivot place
A1 = LU_doolittle(A1,row)                                          # calling LU decomposition function

print("The transformed LU matrix is ")
print_matrix(A1,row,row)

x = [0 for i in range(row)]
x = backward_subs_doolittle(A1,row,b_vector)                       # calling function for backward substitution

print("The solutions are : ")
print()
for i in range(row):
    print("x["+str(i+1)+"] = "+str(x[i]))                          # printing the solutions
```

```
The original matrix is:

1.0    0.0    1.0    2.0

0.0    1.0    -2.0    0.0

1.0    2.0    -1.0    0.0

2.0    1.0    3.0    -2.0


The transformed LU matrix is

1.0    0.0    1.0    2.0

0.0    1.0    -2.0    0.0

1.0    2.0    2.0    -2.0

2.0    1.0    1.5    -3.0


The solutions are :

x[1] = 1.0
x[2] = -1.0
x[3] = 1.0
x[4] = 2.0
```

LU decomposition using Crout's condition i.e. U[i][i]=1

In [152…

```
print("The original matrix is: ")
A2,row,col = read_matrix('matrix_Q1.txt')                          # reading and printing the matrix given in the question
print_matrix(A2,row,col)

b_vector=[6,-3,-2,0]                                               # defining the vector b

A2, b_vector = partial_pivot_LU(A2, b_vector, row)                 # partial pivoting to avoid division by zero at pivot place
A2=LU_crout(A2,ro)                                                 # calling LU decomposition function

print("The transformed LU matrix is ")
print_matrix(A2,row,row)

x = [0 for i in range(row)]
x= backward_subs_crout(A2,row,b_vector)                            # calling function for backward substitution

print("The solutions are : ")
print()
for i in range(row):
    print("x["+str(i+1)+"] = "+str(x[i]))                          # printing the solutions
```

```
The original matrix is:

1.0    0.0    1.0    2.0

0.0    1.0    -2.0    0.0

1.0    2.0    -1.0    0.0

2.0    1.0    3.0    -2.0


The transformed LU matrix is

1.0    0.0    1.0    2.0

0.0    1.0    -2.0    0.0

1.0    2.0    2.0    -1.0

2.0    1.0    3.0    -3.0


The solutions are :

x[1] = 1.0
x[2] = -1.0
x[3] = 1.0
x[4] = 2.0
```

Question 2

In [153…

```
print("The original matrix is : ")
B,row,col=read_matrix('matrix_Q2.txt')                             # reading and printing the matrix given in the question
print_matrix(B,row,row)

C=copy.deepcopy(B)                                                 # deepcopy for unchanged matrix required for inverse

I_matrix=return_identity(row)
B, I_matrix = partial_pivot_LU(B, I_matrix, row)                   # Then partial pivoting is done for both matrix and vector.
B=LU_doolittle(B,row)                                              # calling LU decomposition function

print("The transformed LU matrix is ")
print_matrix(B,row,row)


det=determinant(B,row)                                             # storing the determinant
if det == 0:                                                       # Checking if inverse exists
    print("Determinant = zero.\nInverse doesn't exist.")
else:
    print("The inverse is:")


    inverse= inverse_by_LU(C, row)                                 # Then the decomposition algorithm is applied.
    print_matrix(inverse,row,row)                                   # Calculating and printing inverse


    print("Multiplying original matrix with the inverse for Verification: ")                                  # Verification:
    mm,r,c=matrix_multiply(C,row,row,inverse,row,row)                  # to see if it gives indentity matrix on multiplication
    print_matrix(round_matrix(mm),r,c)                              # of our obtained inverse matrix with original matrix
```

```
The original matrix is :

0.0    2.0    8.0    6.0

0.0    0.0    1.0    2.0

0.0    1.0    0.0    1.0

3.0    7.0    1.0    0.0


The transformed LU matrix is

3.0    7.0    1.0    0.0

0.0    2.0    8.0    6.0

0.0    0.0    1.0    2.0

0.0    0.5    -4.0    6.0


The inverse is:

-0.25000000000000006    1.6666666666666672    -1.8333333333333333    0.3333333333333333

0.08333333333333337    -0.6666666666666667    0.8333333333333333    0.0

0.16666666666666666    -0.3333333333333326    -0.3333333333333333    0.0

-0.08333333333333333    0.6666666666666666    0.16666666666666666    0.0


Multiplying original matrix with the inverse for Verification:

1.0    0.0    0.0    0.0

0.0    1.0    0.0    0.0

0.0    -0.0    1.0    0.0

0.0    -0.0    -0.0    1.0
```

Question 3

Solving using Cholesky decomposition

In [154…

```
print("The original matrix is: ")
C,row,col=read_matrix('matrix_Q3.txt')                             # reading and printing the matrix given in the question
print_matrix(C,row,col)

b_vector=[2.20, 2.85, 2.79, 2.87]                                  # defining the vector b


C, b_vector = partial_pivot_LU(C, b_vector, row)                   # partial pivoting to avoid division by zero at pivot place
C=LU_Cho(C,row)                                                    # calling Cholesky decomposition function
print("The transformed Cholesky matrix is: ")
round_matrix(C)
print_matrix(C,row,row)

a=backward_subs_Cholesky(C,row,b_vector)                           # calling backward substitution Cholesky function

print("The solutions are : ")
print()
for i in range(row):
    print('%.2f'%a[i])                                             # printing the solutions
```

```
The original matrix is:

10.0    1.0    0.0    2.5    2.2

1.0    12.0    -0.3    1.1    2.85

0.0    -0.3    9.5    0.0    2.79

2.5    1.1    0.0    6.0    2.87


The transformed Cholesky matrix is:

3.16    0.32    0.0    0.79

0.32    3.45    -0.09    0.25

0.0    -0.09    3.08    0.01

0.79    0.25    0.01    2.31


The solutions are :

0.10
0.20
0.30
0.40
```

In [ ]: