Ting Fung Lam 2924629375

MATH 501 PA4

The solution is x = [1 1 1 1] in both cases.

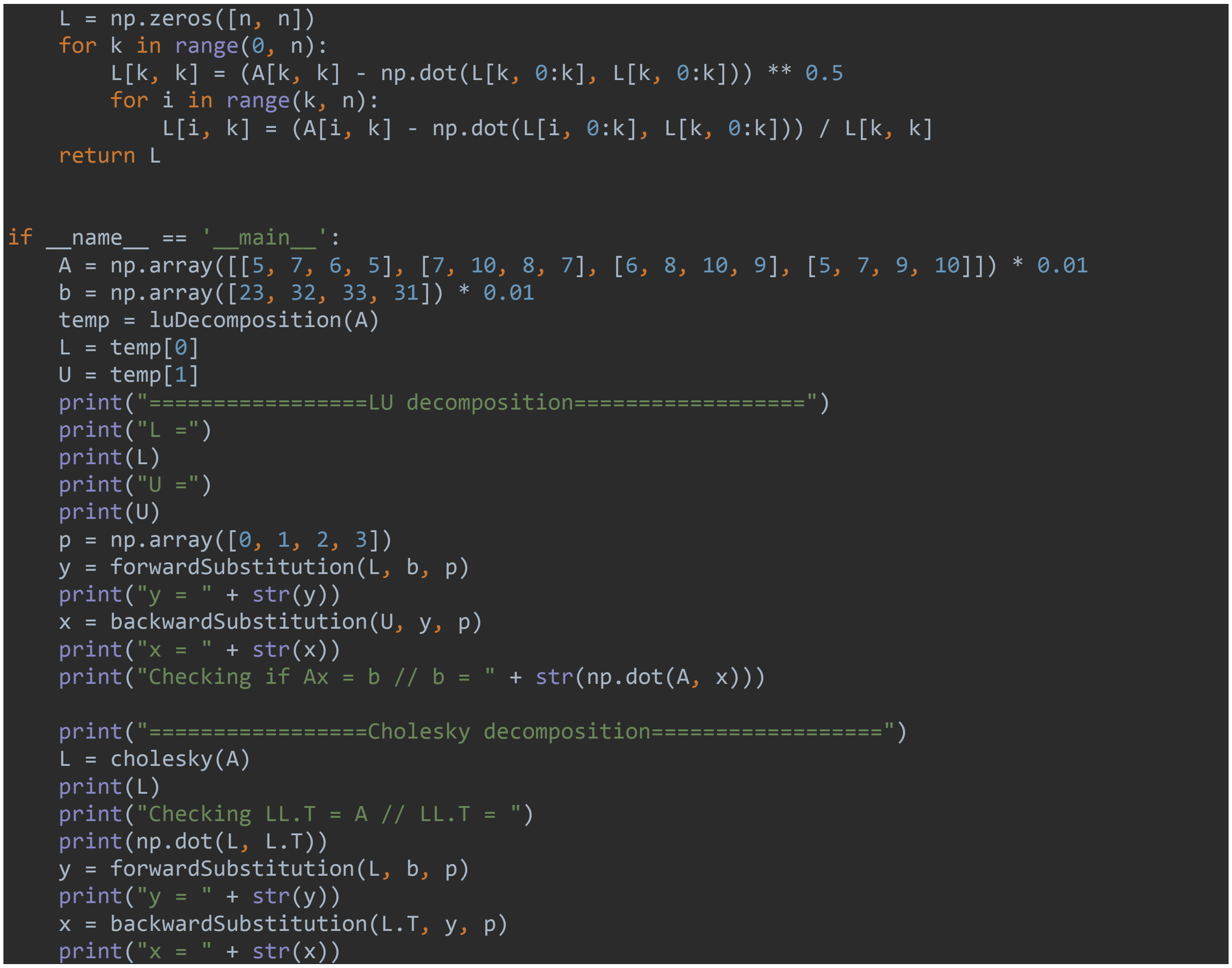
```python
import numpy as np


def forwardSubstitution(A, b, p):
    """
    Forward substitution for Ax = b
    :param A: Lower triangular matrix permuted
    :param b: Vector b
    :param p: permutation matrix with p[1] be the row with all zeros except diagonal
    :return: Vector x
    """
    n = np.size(A, 0)
    x = np.zeros(n)
    for i in range(0, n):
        x[i] = (b[p[i]] - np.dot(A[p[i], 0:i], x[0:i])) / A[p[i], i]
    return x


def backwardSubstitution(A, b, p):
    """
    Backward substitution for Ax = b
    :param A: Upper triangular matrix permuted
    :param b: Vector b
    :param p: permutation matrix with p[1] be the row with no zeros
    :return:
    """
    n = np.size(A, 0)
    x = np.zeros(n)
    for i in reversed(range(0, n)):
        x[i] = (b[p[i]] - np.dot(A[p[i], i+1:n], x[i+1:n])) / A[p[i], i]
    return x


def luDecomposition(A):
    """
    LU decomposition
    :param A: Input matrix
    :return: Matrices L and U
    """
    n = np.size(A, 0)
    L = np.zeros([n, n])
    U = np.zeros([n, n])
    for k in range(0, n):
        L[k, k] = 1
        U[k, k] = (A[k, k] - np.dot(L[k, 0:k], U[0:k, k])) / L[k, k]
        for j in range(k, n):
            U[k, j] = (A[k, j] - np.dot(L[k, 0:k], U[0:k, j])) / L[k, k]
        for i in range(k, n):
            L[i, k] = (A[i, k] - np.dot(L[i, 0:k], U[0:k, k])) / U[k, k]
    return L, U


def cholesky(A):
    n = np.size(A, 0)
```



```
    L = np.zeros([n, n])
    for k in range(0, n):
        L[k, k] = (A[k, k] - np.dot(L[k, 0:k], L[k, 0:k])) ** 0.5
        for i in range(k, n):
            L[i, k] = (A[i, k] - np.dot(L[i, 0:k], L[k, 0:k])) / L[k, k]
    return L


if __name__ == '__main__':
    A = np.array([[5, 7, 6, 5], [7, 10, 8, 7], [6, 8, 10, 9], [5, 7, 9, 10]]) * 0.01
    b = np.array([23, 32, 33, 31]) * 0.01
    temp = luDecomposition(A)
    L = temp[0]
    U = temp[1]
    print("================LU decomposition=================")
    print("L =")
    print(L)
    print("U =")
    print(U)
    p = np.array([0, 1, 2, 3])
    y = forwardSubstitution(L, b, p)
    print("y = " + str(y))
    x = backwardSubstitution(U, y, p)
    print("x = " + str(x))
    print("Checking if Ax = b // b = " + str(np.dot(A, x)))

    print("================Cholesky decomposition=================")
    L = cholesky(A)
    print(L)
    print("Checking LL.T = A // LL.T = ")
    print(np.dot(L, L.T))
    y = forwardSubstitution(L, b, p)
    print("y = " + str(y))
    x = backwardSubstitution(L.T, y, p)
    print("x = " + str(x))
```

```
=================LU decomposition==================
L =
[[ 1.   0.   0.   0. ]
 [ 1.4  1.   0.   0. ]
 [ 1.2 -2.   1.   0. ]
 [ 1.   0.   1.5  1. ]]
U =
[[ 0.05   0.07   0.06   0.05 ]
 [ 0.     0.002 -0.004  0.   ]
 [ 0.     0.     0.02   0.03 ]
 [ 0.     0.     0.     0.005]]
y = [ 0.23  -0.002  0.05   0.005]
x = [1. 1. 1. 1.]
Checking if Ax = b // b = [0.23 0.32 0.33 0.31]
=================Cholesky decomposition==================
[[ 0.2236068   0.          0.          0.        ]
 [ 0.31304952  0.04472136  0.          0.        ]
 [ 0.26832816 -0.08944272  0.14142136  0.        ]
 [ 0.2236068   0.          0.21213203  0.07071068]]
Checking LL.T = A // LL.T =
[[0.05 0.07 0.06 0.05]
 [0.07 0.1  0.08 0.07]
 [0.06 0.08 0.1  0.09]
 [0.05 0.07 0.09 0.1 ]]
y = [ 1.02859127 -0.04472136  0.35355339  0.07071068]
x = [1. 1. 1. 1.]
```