第三周上机

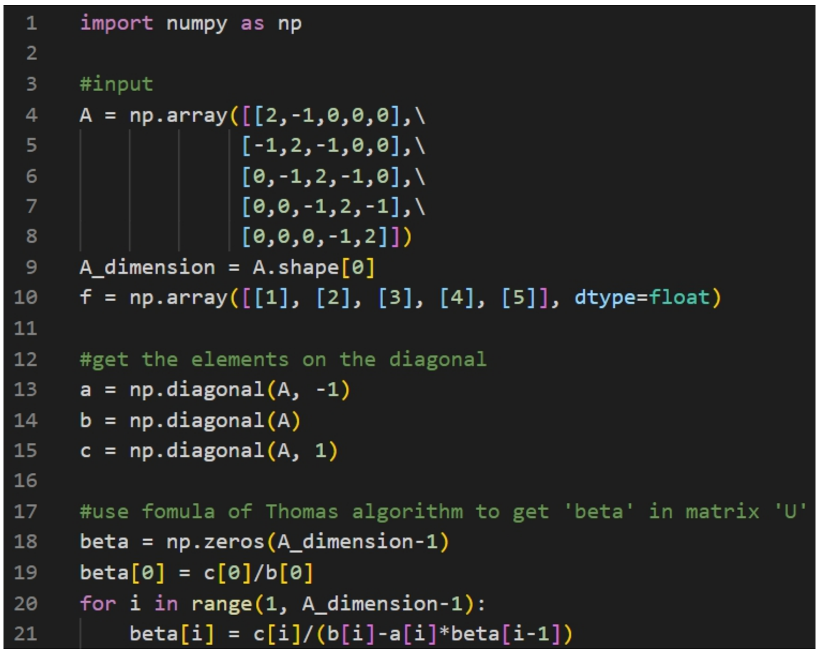
一、问题叙述

分别用追赶法和平方根法（Cholesky分解）求解线性方程组 $Ax=b$，其中

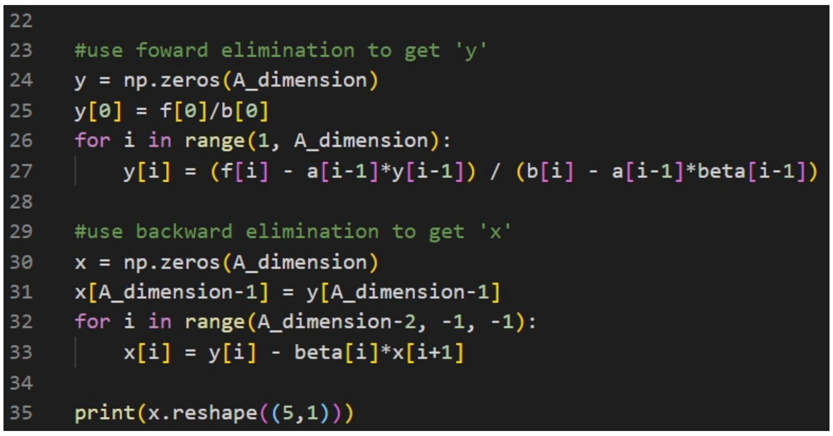
$$A=\begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \qquad b=\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

二、追赶法（Thomas）

代码如下：

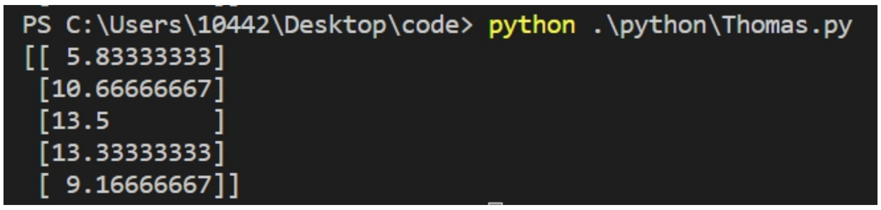


```python
import numpy as np

#input
A = np.array([[2,-1,0,0,0],\
              [-1,2,-1,0,0],\
              [0,-1,2,-1,0],\
              [0,0,-1,2,-1],\
              [0,0,0,-1,2]])
A_dimension = A.shape[0]
f = np.array([[1], [2], [3], [4], [5]], dtype=float)

#get the elements on the diagonal
a = np.diagonal(A, -1)
b = np.diagonal(A)
c = np.diagonal(A, 1)

#use fomula of Thomas algorithm to get 'beta' in matrix 'U'
beta = np.zeros(A_dimension-1)
beta[0] = c[0]/b[0]
for i in range(1, A_dimension-1):
    beta[i] = c[i]/(b[i]-a[i]*beta[i-1])
```



```python

#use foward elimination to get 'y'
y = np.zeros(A_dimension)
y[0] = f[0]/b[0]
for i in range(1, A_dimension):
    y[i] = (f[i] - a[i-1]*y[i-1]) / (b[i] - a[i-1]*beta[i-1])

#use backward elimination to get 'x'
x = np.zeros(A_dimension)
x[A_dimension-1] = y[A_dimension-1]
for i in range(A_dimension-2, -1, -1):
    x[i] = y[i] - beta[i]*x[i+1]

print(x.reshape((5,1)))
```

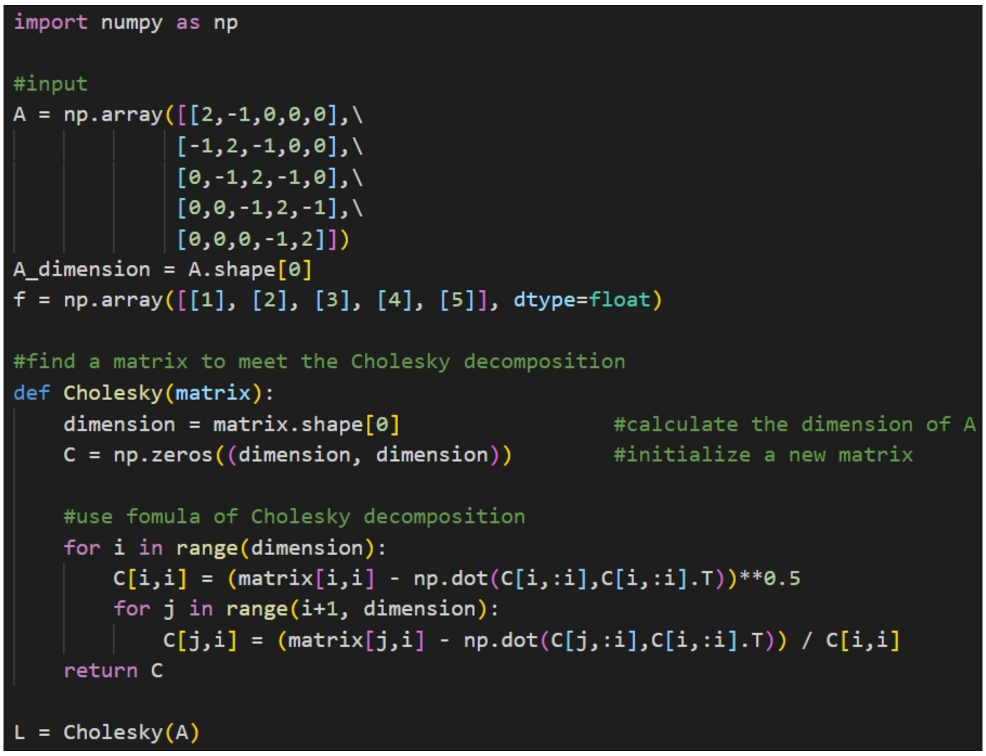
输出结果如下：



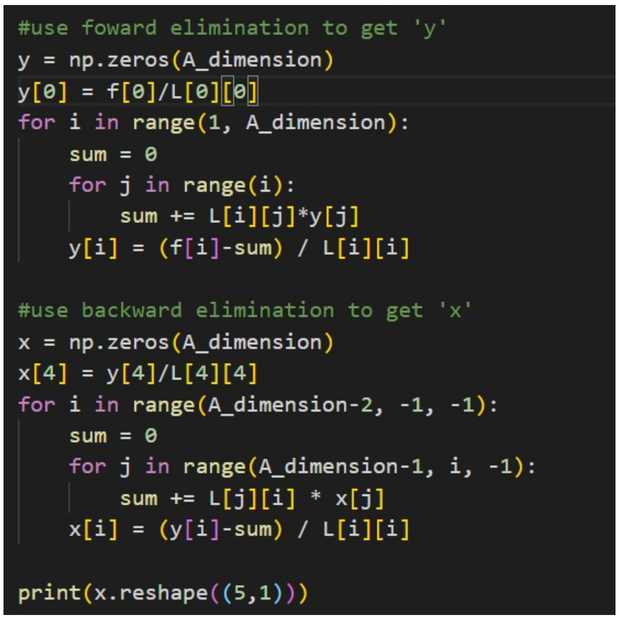
```
PS C:\Users\10442\Desktop\code> python .\python\Thomas.py
[[ 5.83333333]
 [10.66666667]
 [13.5       ]
 [13.33333333]
 [ 9.16666667]]
```

三、Cholesky 分解

代码如下：

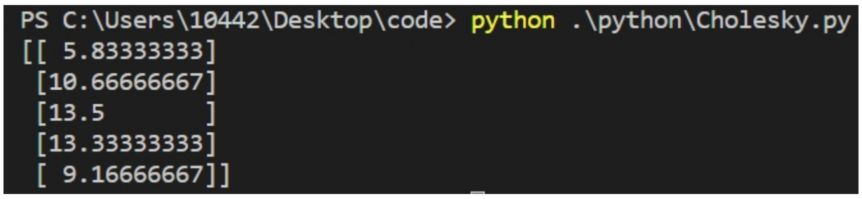


```python
import numpy as np

#input
A = np.array([[2,-1,0,0,0],\
              [-1,2,-1,0,0],\
              [0,-1,2,-1,0],\
              [0,0,-1,2,-1],\
              [0,0,0,-1,2]])
A_dimension = A.shape[0]
f = np.array([[1], [2], [3], [4], [5]], dtype=float)

#find a matrix to meet the Cholesky decomposition
def Cholesky(matrix):
    dimension = matrix.shape[0]               #calculate the dimension of A
    C = np.zeros((dimension, dimension))      #initialize a new matrix

    #use fomula of Cholesky decomposition
    for i in range(dimension):
        C[i,i] = (matrix[i,i] - np.dot(C[i,:i],C[i,:i].T))**0.5
        for j in range(i+1, dimension):
            C[j,i] = (matrix[j,i] - np.dot(C[j,:i],C[i,:i].T)) / C[i,i]
    return C

L = Cholesky(A)
```



```python
#use foward elimination to get 'y'
y = np.zeros(A_dimension)
y[0] = f[0]/L[0][0]
for i in range(1, A_dimension):
    sum = 0
    for j in range(i):
        sum += L[i][j]*y[j]
    y[i] = (f[i]-sum) / L[i][i]

#use backward elimination to get 'x'
x = np.zeros(A_dimension)
x[4] = y[4]/L[4][4]
for i in range(A_dimension-2, -1, -1):
    sum = 0
    for j in range(A_dimension-1, i, -1):
        sum += L[j][i] * x[j]
    x[i] = (y[i]-sum) / L[i][i]

print(x.reshape((5,1)))
```

输出结果如下：

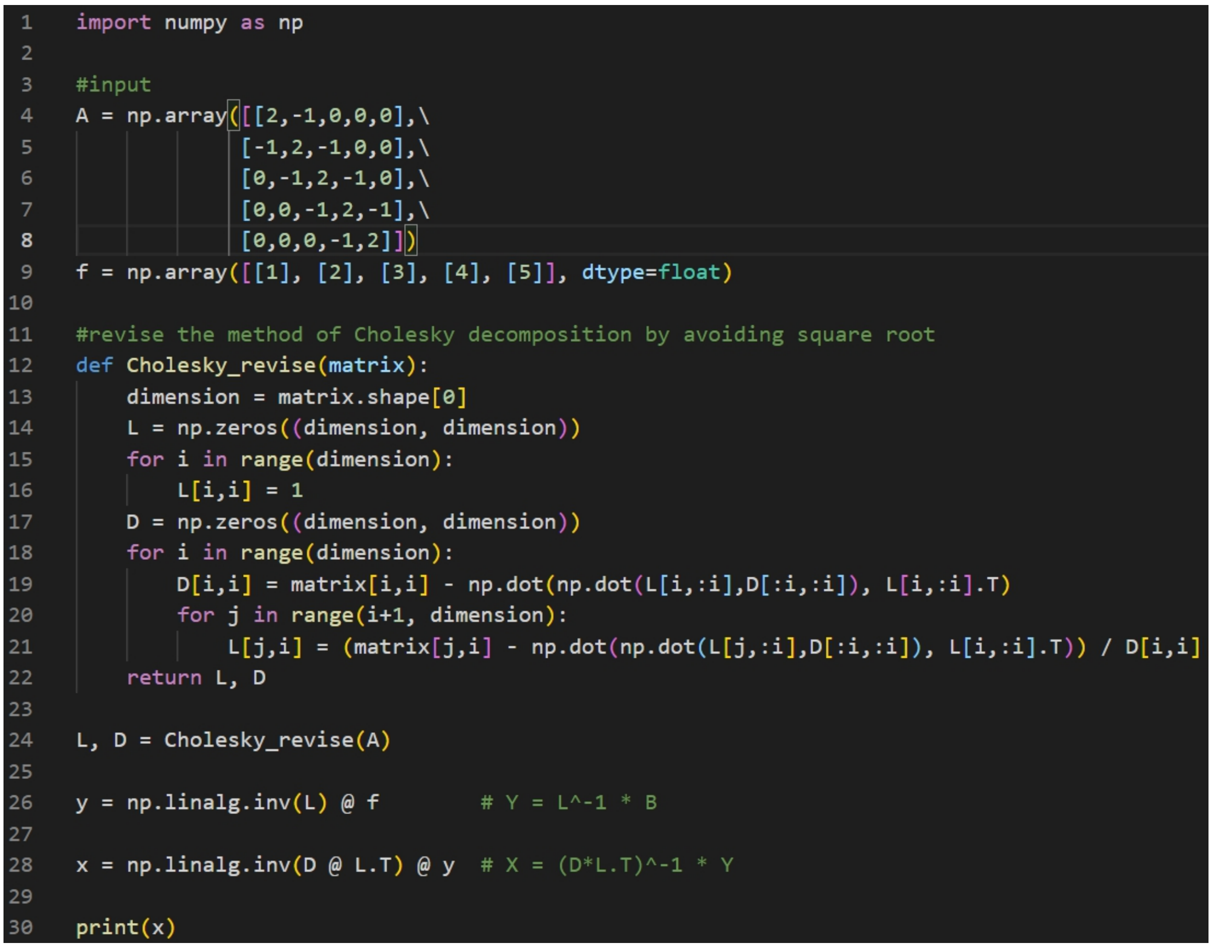


```
PS C:\Users\10442\Desktop\code> python .\python\Cholesky.py
[[ 5.83333333]
 [10.66666667]
 [13.5       ]
 [13.33333333]
 [ 9.16666667]]
```

四、改进的 Cholesky 分解

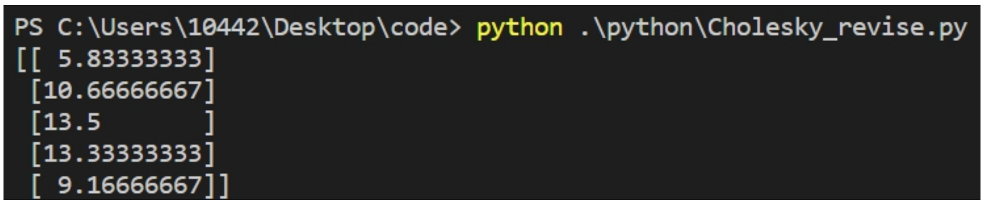
为了避免开方运算，可以将利用改进的 Cholesky 分解使得 A = LDL^T。在编程过程中利用 numpy 自带的矩阵运算进行求逆和求转置等操作，省去了原本手动计算的过程。

代码如下：



```python
import numpy as np

#input
A = np.array([[2,-1,0,0,0],\
              [-1,2,-1,0,0],\
              [0,-1,2,-1,0],\
              [0,0,-1,2,-1],\
              [0,0,0,-1,2]])
f = np.array([[1], [2], [3], [4], [5]], dtype=float)

#revise the method of Cholesky decomposition by avoiding square root
def Cholesky_revise(matrix):
    dimension = matrix.shape[0]
    L = np.zeros((dimension, dimension))
    for i in range(dimension):
        L[i,i] = 1
    D = np.zeros((dimension, dimension))
    for i in range(dimension):
        D[i,i] = matrix[i,i] - np.dot(np.dot(L[i,:i],D[:i,:i]), L[i,:i].T)
        for j in range(i+1, dimension):
            L[j,i] = (matrix[j,i] - np.dot(np.dot(L[j,:i],D[:i,:i]), L[i,:i].T)) / D[i,i]
    return L, D

L, D = Cholesky_revise(A)

y = np.linalg.inv(L) @ f        # Y = L^-1 * B

x = np.linalg.inv(D @ L.T) @ y  # X = (D*L.T)^-1 * Y

print(x)
```

输出如下：



```
PS C:\Users\10442\Desktop\code> python .\python\Cholesky_revise.py
[[ 5.83333333]
 [10.66666667]
 [13.5       ]
 [13.33333333]
 [ 9.16666667]]
```

五、结果分析

三种方法计算的结果都一致，事实上只有在精确到小数点后 16 位时（python 的浮点精度），才会有细微的差别，可见通过追赶法和 Cholesky 分解求解的精度都非常高。