# Biweekly quiz 5 Matrix Decomposition

December 27, 2019

[119]:
```
#use different method to find a certain eigenvalue and eigenvector pair
#(a)A = LU
#(b)modified LU decomposition and solve Ax=b
#(c)A = LLt, cholesky algorithm
#(d)modidy cholesky algorithm and solve Ax=b

#usage = for biweekly quiz 5 for class PMS.CM Chang @ NCTU AM 11
#Biweekly quiz 5 Matrix Decomposition
#Pactice of Mathematics Software

#author = maxwill lin =  yt lin
#school number = 0712238@NCTU

#created 2019.12.24
#modified 2019.12.27

#show ans by printing out
```

[122]:
```
import numpy as np
import scipy.linalg
np.set_printoptions(suppress=True)

A = np.eye(12)*4
for i in range(0,3):
    A[i][i+1] = A[i+1][i] = -1
for i in range(0,3):
    A[11-i][10-i] = A[10-i][11-i] = -1
for i in range(0,4):
    A[5+i][3+i] = A[3+i][5+i] = -1
A[4][0] = A[0][4] = -1
A[7][11] = A[11][7] = -1
print(A)
b = np.asarray([220., 110., 110., 220., 110., 110., 110., 110., 220., 110., 110.
↪, 220.]).T
print(b)

eps = 1e-5
```

```
x0 = np.zeros(A.shape[0])
```

```
[[ 4. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.]
 [-1.  4. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  4. -1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  4.  0. -1.  0.  0.  0.  0.  0.  0.]
 [-1.  0.  0.  0.  4.  0. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.  4.  0. -1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. -1.  0.  4.  0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. -1.  0.  4.  0.  0.  0. -1.]
 [ 0.  0.  0.  0.  0.  0. -1.  0.  4. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -1.  4. -1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  4. -1.]
 [ 0.  0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  4.]]
[220. 110. 110. 220. 110. 110. 110. 110. 220. 110. 110. 220.]
```

[10]:
```
#direct solve
Ainv = np.linalg.inv(A)
print("direct solution:")
print(np.dot(Ainv, b))
```

```
direct solution:
[88. 66. 66. 88. 66. 66. 66. 66. 88. 66. 66. 88.]
```

[121]:
```
print("check for scipy LU library result")
LU0,P0 =  scipy.linalg.lu_factor(A) #store in one matrix
print(LU0)
#U0 = np.triu(LU0)
#L0 = np.tril(LU0, -1)
scipy.linalg.lu_solve((LU0, P0), b)
```

```
check for scipy LU library result
[[ 4.         -1.          0.          0.         -1.          0.
   0.          0.          0.          0.          0.          0.        ]
 [-0.25        3.75       -1.          0.         -0.25        0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.         -0.26666667  3.73333333 -1.         -0.06666667  0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.         -0.26785714  3.73214286 -0.01785714 -1.
   0.          0.          0.          0.          0.          0.        ]
 [-0.25       -0.06666667 -0.01785714 -0.00478469  3.73205742 -0.00478469
  -1.          0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.         -0.26794258 -0.00128205  3.73205128
  -0.00128205 -1.          0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.         -0.26794872 -0.00034352
   3.73205084 -0.00034352 -1.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.         -0.26794916
  -0.00009205  3.73205081 -0.00009205  0.          0.         -1.        ]
```

```
 [ 0.          0.          0.          0.          0.          0.
  -0.26794919 -0.00002466  3.73205081 -1.          0.         -0.00002466]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.         -0.26794919  3.73205081 -1.         -0.00000661]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.          0.         -0.26794919  3.73205081 -1.00000177]
 [ 0.          0.          0.          0.          0.          0.
   0.         -0.26794919 -0.00000661 -0.00000177 -0.26794967  3.46410067]]
```

[121]:
```
array([88., 66., 66., 88., 66., 66., 66., 66., 88., 66., 66., 88.])
```

[117]:
```python
def LU_decomp(A):
    LU = np.copy(A).astype(float)
    n  = len(A)
    for j in range(n-1):
        for i in range(j+1,n):
            LU[i][j] /= LU[j][j]
            for k in range(j+1,n):
                LU[i][k] -= LU[i][j]*LU[j][k]
    return LU

def LU_forwardsub(L,b):
    n = len(L)
    y = np.copy(b).astype(float)
    for i in range(1,n):
        for j in range(i):
            y[i] -= L[i][j]*y[j]
    return y

def LU_backwardsub(U,y):
    n = len(U)
    x = np.copy(y).astype(float)
    for i in range(n-1,-1,-1):
        for j in range(i+1,n):
            x[i] -= U[i][j]*x[j]
        x[i] /= U[i][i]
    return x

def LU_solve(A,b):
    print("LU decomposition")
    LU = LU_decomp(A) #store in one matrix
    print(LU)
    print("solve for Ly = b")
    y = LU_forwardsub(LU,b)
    print(y)
    print("solve for Uy = x")
    x = LU_backwardsub(LU,y)
```

```
    print(x)
    return x
```

[118]:
```
LU_solve(A,b)
```

```
LU decomposition
[[ 4.         -1.          0.          0.         -1.          0.
   0.          0.          0.          0.          0.          0.        ]
 [-0.25        3.75       -1.          0.         -0.25        0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.         -0.26666667  3.73333333 -1.         -0.06666667  0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.         -0.26785714  3.73214286 -0.01785714 -1.
   0.          0.          0.          0.          0.          0.        ]
 [-0.25       -0.06666667 -0.01785714 -0.00478469  3.73205742 -0.00478469
  -1.          0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.         -0.26794258 -0.00128205  3.73205128
  -0.00128205 -1.          0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.         -0.26794872 -0.00034352
   3.73205084 -0.00034352 -1.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.         -0.26794916
  -0.00009205  3.73205081 -0.00009205  0.          0.         -1.        ]
 [ 0.          0.          0.          0.          0.          0.
  -0.26794919 -0.00002466  3.73205081 -1.          0.         -0.00002466]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.         -0.26794919  3.73205081 -1.         -0.00000661]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.          0.         -0.26794919  3.73205081 -1.00000177]
 [ 0.          0.          0.          0.          0.          0.
   0.         -0.26794919 -0.00000661 -0.00000177 -0.26794967  3.46410067]]
solve for Ly = b
[220.         165.         154.         261.25       180.
 180.23076923 158.29268293 158.30725331 262.41830065 180.31477174
 158.31519747 304.84085862]
solve for Uy = x
[88. 66. 66. 88. 66. 66. 66. 66. 88. 66. 66. 88.]
```

[118]: array([88., 66., 66., 88., 66., 66., 66., 66., 88., 66., 66., 88.])

[115]:
```
def Cholesky_decomp(A):
    n = len(A)
    L = np.zeros((n,n)).astype(float)
    for i in range(n):
        for j in range(i+1):
            L[i][j] = A[i][j]
            for k in range(j):
                L[i][j] -= L[i][k]*L[j][k]
```

```
            if(i == j):
                L[i][j] = np.sqrt(L[j][j])
            else:
                L[i][j] /= L[j][j]
    return L

def Cholesky_forwardsub(L, b):
    y = b.copy()
    for i in range(len(b)):
        for j in range(i):
            y[i] -= L[i][j]*y[j]
        y[i] /= L[i][i]
    return y

def Cholesky_backwardsub(U, y):
    n = len(U)
    x = np.copy(y)
    for i in range(n-1,-1,-1):
        for j in range(i+1,n):
            x[i] -= U[j][i]*x[j]
        x[i] /= U[i][i]
    return x

def Cholesky_solve(A,b):
    L = Cholesky_decomp(A)
    print("Cholesky decomposition\n", L)
    y = Cholesky_forwardsub(L,b)
    print("Solve for Ly = b\n", y)
    x = Cholesky_backwardsub(L,y)
    print("Solve for LTx = y\n", x)
    return x
```

[116]:

```
Cholesky_solve(A,b)
```

```
Cholesky decomposition
 [[ 2.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.        ]
 [-0.5         1.93649167  0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.         -0.51639778  1.93218357  0.          0.          0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.         -0.51754917  1.93187548  0.          0.
   0.          0.          0.          0.          0.          0.        ]
 [-0.5        -0.12909944 -0.03450328 -0.00924342  1.93185336  0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.         -0.51763171 -0.00247674  1.93185178
   0.          0.          0.          0.          0.          0.        ]
```

```
 [ 0.          0.          0.          0.         -0.51763763 -0.00066364
   1.93185166  0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.         -0.51763806
  -0.00017782  1.93185165  0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
  -0.51763809 -0.00004765  1.93185165  0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.         -0.51763809  1.93185165  0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.          0.         -0.51763809  1.93185165  0.        ]
 [ 0.          0.          0.          0.          0.          0.
   0.         -0.51763809 -0.00001277 -0.00000342 -0.51763901  1.86120946]]
Solve for Ly = b
 [110.          85.20563362  79.7025721  135.23128336  93.17477374
  93.29430525  81.93832171  81.94586424 135.83770798  93.33779408
  81.94997647 163.78643277]
Solve for LTx = y
 [88. 66. 66. 88. 66. 66. 66. 66. 88. 66. 66. 88.]
```

[116]:
```
array([88., 66., 66., 88., 66., 66., 66., 66., 88., 66., 66., 88.])
```

[120]:
```python
print("check for np library result")
Ach = np.linalg.cholesky(A)
print(Ach)
y = np.linalg.solve(Ach, b)
x = np.linalg.solve(Ach.T, y)
print(y, '\n', x)
print("###############################")
print(Cholesky_decomp(A))
```

```
check for np library result
[[ 2.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.        ]
 [-0.5         1.93649167  0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.         -0.51639778  1.93218357  0.          0.          0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.         -0.51754917  1.93187548  0.          0.
   0.          0.          0.          0.          0.          0.        ]
 [-0.5        -0.12909944 -0.03450328 -0.00924342  1.93185336  0.
   0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.         -0.51763171 -0.00247674  1.93185178
   0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.         -0.51763763 -0.00066364
   1.93185166  0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.         -0.51763806
  -0.00017782  1.93185165  0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
```

```
 -0.51763809 -0.00004765  1.93185165  0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
  0.          0.         -0.51763809  1.93185165  0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
  0.          0.          0.         -0.51763809  1.93185165  0.        ]
 [ 0.          0.          0.          0.          0.          0.
  0.         -0.51763809 -0.00001277 -0.00000342 -0.51763901  1.86120946]]
[110.          85.20563362  79.7025721  135.23128336  93.17477374
  93.29430525  81.93832171  81.94586424 135.83770798  93.33779408
  81.94997647 163.78643277]
 [88. 66. 66. 88. 66. 66. 66. 66. 88. 66. 66. 88.]
##############################
[[ 2.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.        ]
 [-0.5         1.93649167  0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.        ]
 [ 0.         -0.51639778  1.93218357  0.          0.          0.
  0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.         -0.51754917  1.93187548  0.          0.
  0.          0.          0.          0.          0.          0.        ]
 [-0.5        -0.12909944 -0.03450328 -0.00924342  1.93185336  0.
  0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.         -0.51763171 -0.00247674  1.93185178
  0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.         -0.51763763 -0.00066364
  1.93185166  0.          0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.         -0.51763806
 -0.00017782  1.93185165  0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
 -0.51763809 -0.00004765  1.93185165  0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
  0.          0.         -0.51763809  1.93185165  0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
  0.          0.          0.         -0.51763809  1.93185165  0.        ]
 [ 0.          0.          0.          0.          0.          0.
  0.         -0.51763809 -0.00001277 -0.00000342 -0.51763901  1.86120946]]
```

[ ]: