黄玟瑜
19335074
huangmy73@mail2.sysu.edu.cn

Homework 9

数值计算方法，2021 春

2021-05-15

周一作业

用豪斯霍尔德变换求下列矩阵的 QR 分解，并用 QR 算法求对应的特征值与特征向量。

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

解：使用 python 编程，程序如下所示：

```python
import numpy as np


def HouseHolder(mat: np.array):
    ROW = mat.shape[0]  # 获取矩阵行数n
    COL = mat.shape[1]  # 获取矩阵列数m
    Q = np.eye(ROW)     # 初始化Q为n维的单位矩阵
    R = np.copy(mat)    # 初始化R为A

    for n in range(COL - 1):
        Norm = np.linalg.norm(R[n:, n])     # 计算x的二范数
        e = np.zeros((ROW - n))
        e[0] = 1
        v = np.transpose([e]) * Norm - np.transpose([R[n:, n]])  # v = w - x
        vT = np.transpose(v)                                     # v的转置
        P = v.dot(vT) / vT.dot(v)
        H = np.eye(ROW)
        H[n:, n:] = H[n:, n:] - 2 * P       # H = I - 2P

        R = H.dot(R)    # R = HnHn-1···H2H1A
        Q = Q.dot(H)    # Q = H1H2···Hn-1Hn

    return Q, R
# A = np.array([
#     [1, -4],
#     [2, 3],
#     [2, 2],
# ])


def unshiftedQR(A: np.array, k):
    ROW = A.shape[0]
    Q = np.eye(ROW)
    Qbar = Q
```

```python
    R = A

    for n in range(k):
        [Q, R] = HouseHolder(R.dot(Q))
        Qbar = Qbar.dot(Q)

    lam = np.diagonal(R.dot(Q))

    return lam, Qbar


A1 = np.array([
    [3, -1, 2],
    [4, 1, 0],
    [-3, 2, 1],
    [1, 1, 5],
    [-2, 0, 3]
])

A2 = np.array([
    [4, 2, 3, 0],
    [-2, 3, -1, 1],
    [1, 3, -4, 2],
    [1, 0, 1, -1],
    [3, 1, 3, -2]
])

q, r = HouseHolder(A1)
print("A =")
print(A1)
print("Q =")
print(q)
print("R =")
print(r)
print("Q × R =")
print(q.dot(r))
A1 = A1.dot(A1.T)
lam, q = unshiftedQR(A1, 20)
print("特征值 = ")
print(lam)
print("特征向量 = ")
print(q)

q, r = HouseHolder(A2)
print("A =")
print(A2)
print("Q =")
print(q)
print("R =")
print(r)
print("Q × R =")
print(q.dot(r))
A2 = A2.dot(A2.T)
lam, q = unshiftedQR(A2, 20)
print("特征值 = ")
print(lam)
print("特征向量 = ")
print(q)
```

结果如下所示：

```
A =
[[ 3 -1  2]
 [ 4  1  0]
 [-3  2  1]
 [ 1  1  5]
 [-2  0  3]]
Q =
[[ 0.48038446 -0.26969003  0.57569512  0.56276351 -0.21993276]
 [ 0.64051262  0.54936859  0.19893066 -0.34730278  0.35741449]
 [-0.48038446  0.65924231  0.47804395  0.11836131 -0.30347146]
 [ 0.16012815  0.42950635 -0.57932345  0.67334367  0.02959568]
 [-0.32025631 -0.07990816  0.25467634  0.30866958  0.85493487]]
R =
[[ 6.24499800e+00 -6.40512615e-01  3.20256308e-01]
 [ 3.02903058e-16  2.56704959e+00  2.02766952e+00]
 [ 2.57042773e-16 -1.12418537e-16 -5.03154040e-01]
 [ 1.52121096e-16  9.98544685e-17  5.53661544e+00]
 [-2.18719912e-16 -2.22803269e-17  1.96944603e+00]]
Q x R =
[[ 3.00000000e+00 -1.00000000e+00  2.00000000e+00]
 [ 4.00000000e+00  1.00000000e+00  7.14755762e-16]
 [-3.00000000e+00  2.00000000e+00  1.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00  5.00000000e+00]
 [-2.00000000e+00 -2.75346609e-16  3.00000000e+00]]
A x A转置 =
[[ 14  11  -9  12   0]
 [ 11  17 -10   5  -8]
 [ -9 -10  14   4   9]
 [ 12   5   4  27  13]
 [  0  -8   9  13  13]]
特征值 =
[ 4.10032399e+01  3.83045299e+01  5.69223022e+00 -1.33515342e-15
  6.17976047e-16]
特征向量 =
[[ 0.55359365  0.12927982 -0.38071198 -0.21358579 -0.69732367]
 [ 0.45882839  0.4034859   0.62350315 -0.42995487  0.23034382]
 [-0.23206729 -0.5062094   0.59684832 -0.15525078 -0.55638665]
 [ 0.64873387 -0.49777657  0.13966397  0.52687064  0.1851058 ]
 [ 0.09110281 -0.56254457 -0.30094562 -0.68397694  0.341835   ]]
```

```
A =
[[ 4  2  3  0]
 [-2  3 -1  1]
 [ 1  3 -4  2]
 [ 1  0  1 -1]
 [ 3  1  3 -2]]
Q =
[[ 0.71842121  0.21150374  0.2258696  -0.07037921 -0.6190047 ]
 [-0.3592106   0.7684636   0.52281184  0.07968445  0.02737877]
 [ 0.1796053   0.59926061 -0.76877528  0.03001273  0.12927669]
 [ 0.1796053  -0.056401    0.05246724  0.97587244  0.0973706 ]
 [ 0.53881591  0.04935087  0.28615111 -0.18833314  0.76804302]]
R =
[[ 5.56776436e+00  1.43684242e+00  3.59210604e+00 -1.25723711e+00]
 [-3.84334641e-16  4.57553099e+00 -2.43934318e+00  1.92468407e+00]
 [-1.10595650e-16  2.68109828e-16  4.14081864e+00 -1.63950817e+00]
 [ 8.64911953e-17  2.17619426e-16  2.75622414e-17 -4.59496231e-01]
 [ 6.56036137e-16  9.37402492e-16  2.40720223e-16 -1.34752448e+00]]
Q × R =
[[ 4.00000000e+00  2.00000000e+00  3.00000000e+00 -4.59070213e-16]
 [-2.00000000e+00  3.00000000e+00 -1.00000000e+00  1.00000000e+00]
 [ 1.00000000e+00  3.00000000e+00 -4.00000000e+00  2.00000000e+00]
 [ 1.00000000e+00  3.83296697e-16  1.00000000e+00 -1.00000000e+00]
 [ 3.00000000e+00  1.00000000e+00  3.00000000e+00 -2.00000000e+00]]
A × A转置 =
[[ 29  -5  -2   7  23]
 [ -5  15  13  -4  -8]
 [ -2  13  30  -5 -10]
 [  7  -4  -5   3   8]
 [ 23  -8 -10   8  23]]
特征值 =
[ 5.94380768e+01  3.17351341e+01  7.15567359e+00  1.67111554e+00
 -8.26644998e-17]
特征向量 =
[[ 0.57625771  0.52134386 -0.02814709 -0.61449307 -0.13316772]
 [-0.31238841  0.3366958  -0.88450292  0.04765076 -0.06658386]
 [-0.41508311  0.75818541  0.44663411  0.23065459  0.01331677]
 [ 0.21449019  0.00387137  0.01427972  0.39712961 -0.8922237 ]
 [ 0.59333002  0.19994106 -0.1310587   0.63969852  0.42613669]]
```

周四作业

四、（上机题）分别用 Newton 法和 Broyden 法求解下面非线性方程组

$$\begin{cases} 3x_1 - \cos(x_2 x_3) - 0.5 = 0 \\ x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 = 0 \\ e^{-x_1 x_2} + 20x_3 + \frac{1}{3}(10\pi - 3) = 0 \end{cases}$$

（要求：用 Matlab 编程，并附上源代码及迭代五次的结果，初值可取 $(0.1, 0.1, -0.1)$）

Matlab 程序如下，实现了 Newton 法、下山法和 Broyden 法：

```
N = 5;
x0 = [0.1; 0.1; -0.1];

% 定义变量x, 函数F, 并求出F关于x的雅可比矩阵f
syms x [3 1]
F(x) = [3 * x(1) - cos(x(2) * x(3)) - 0.5; x(1)^2 - 81 * (x(2) + 0.1)^2 + sin(x(3)) + 1.06; exp(-x(1)*x(2)) + 20 * x(3) + (10 * pi - 3) / 3];
f=jacobian(F, x);


% Newton法

disp("Newton method:");
 for i=1:N
    eval(['x = x' num2str(i-1) ';' ])
    eval(['x' num2str(i) '= x - inv(vpa(f(x(1), x(2), x(3)))) * vpa(F(x(1), x(2), x(3)))' ])
 end

% Newton的改进: 下山法
% 下山因子=0.5
omega = 0.5;
disp("Mountain down method:");
for i=1:N
    eval(['x = x' num2str(i-1) ';' ])
    eval(['x' num2str(i) '= x - omega * inv(vpa(f(x(1), x(2), x(3)))) * vpa(F(x(1), x(2), x(3)))' ])
end

% Broyden法
disp("Broyden method:");
A0 = eye(3);
for i=1:N
    eval(['x = x' num2str(i-1) ';' ])
    Fx =  vpa(F(x(1), x(2), x(3)));
    eval(['s = - inv(A' num2str(i-1) ') * Fx;' ])
    sT = transpose(s);
    eval(['x' num2str(i) '= x' num2str(i-1) '+ s'])
```

```
    eval(['x = x' num2str(i) ';' ])
    y = vpa(F(x(1), x(2), x(3))) - Fx;
    eval(['A' num2str(i) ' = A' num2str(i-1) ' + ((y-A' num2str(i-1) '*s) * sT)/(sT*s);'])
end
```

结果如下所示：

```
Newton method:

x1 =

  0.49986967292642854044089497798318
 0.019466848537418112792715829350757
 -0.52152047193583068987701148127747


x2 =

   0.50001424016421887221609758575053
 0.0015885913702938952129281120160864
  -0.52355696434763834523284301947976


x3 =

     0.50000011346783422891666389794957
 0.000012444783321550719598689655296247
    -0.52359845007288941501793729477162


x4 =

          0.50000000000707563535579145188773
 0.0000000007757857226131101282584811104832
         -0.52359877557800700440135731805694


x5 =

                  0.50000000000000000002749980135443
 0.00000000000000000025229051133661697007690640688942
                 -0.52359877559829888220571487733204

Mountain down method:

x1 =

  0.29993483646321427022044748899159
 0.059733424268709056396357914675379
 -0.31076023596791534493850574063873


x2 =

  0.39994797243260067009395267223288
 0.034527946975942254059727777804186
  -0.41681347660126417144188350180 43

```

```
x3 =

   0.44997855546915082329011311717727
  0.019198073914368112785000508085241
  -0.47008042954032872637067147665507


x4 =

   0.47499354174296646109852852430999
   0.01029193215832174099260904165166
  -0.49679941433543713646775345184349


x5 =

   0.48749852967965684482952487823459
  0.005364374847434096681298122214385
  -0.51018723172375685954554624986575

Broyden method:

x1 =

   1.2999500004166652777802579337522
   2.3698334166468281523068141984106
  -8.5620253457151456990193973390775


x2 =

  0.5333435468678313442256374594347
  117.72428925421912574219059227483
  29.346562074396094516459480405031


x3 =

    1.1374953123289346031029143873425
  -0.27923131098168089191664802727743
  -0.51367491884173257748411132512407


x4 =

   -0.7910824704275635164396009570779
  -0.20338653953249973941626886883726
  -0.78401488419191001346885437578112


x5 =

   0.49821180908621788201751218766633
  -0.27607371314372074932668867851906
  -0.53519534048021061087347848735907
```

迭代稳定后的输出如下：

```
x =
```

```

                                                     0.5
-0.00000000000000000049221637676089997968474899917822
                  -0.52359877559829888228457997751389
```

综上可知 Newton 法收敛得最好。