CompSci 206 Assignment 2

May 4, 2017

1 Question 6: NLA 8.2

```
In [523]: # Question 6: NLA 8.2
          import numpy as np

          def mgs(A):
              m = np.shape(A)[0]
              n = np.shape(A)[1]
              Q = np.zeros((m, n))
              R = np.zeros((n, n))
              V = A

              for i in range(0, n):
                  R[i,i] = np.linalg.norm(V[:,i])
                  Q[:,i] = (V[:,i] / R[i,i]).squeeze()
                  for j in range(i+1, n):
                      R[i,j] =  np.dot(Q[:,i], V[:,j])
                      V[:,j] =  (V[:,j].squeeze() - np.dot(R[i,j], Q[:,i])).T
              return Q, R
```

2 Question 7

```
In [524]: # Question 7
          A = np.matrix('0.70000 0.70711; 0.70001 0.70711')
          Q, R = mgs(A)

          print "Q:", Q
          print "R:", R

          ## Check Ortogonality

          print 'norm:', np.linalg.norm(np.dot(Q, Q) - np.eye(2))
```

```
Q: [[ 0.70710173  0.70711183]
 [ 0.70711183 -0.70710173]]
R: [[  9.89956565e-01   1.00000455e+00]
```

```
 [  0.00000000e+00   7.14283864e-06]]
norm: 2.30145902338e-11
```

So orthogonality is satisfied.

```
In [525]: ## Compare the one returned by the qr function:

          new_Q, new_R = np.linalg.qr(A, mode='reduced')
          print 'new Q:', new_Q
          print 'new R:', new_R
```

```
new Q: [[-0.70710173 -0.70711183]
 [-0.70711183  0.70710173]]
new R: [[ -9.89956565e-01  -1.64387919e-16]
 [  0.00000000e+00  -7.14283864e-06]]
```

Note that there is only difference in the signs between the QR factorization from the default function and my mgs function.

3 Question 9: NLA 10.2

```
In [539]: ## 10.2
          ### (a)
          def house(A):
              m = np.shape(A)[0]
              n = np.shape(A)[1]
              W = np.zeros((m, n))
              R = np.copy(A)

              for k in range (0, n):
                  x = R[k:m, k].squeeze()
                  e1 = np.zeros_like(x)
                  e1[0] = np.copysign(np.linalg.norm(x), x[0])
                  v = e1 + x
                  W[k:m,k] = v / np.linalg.norm(v)
                  R[k:m, k:n]=R[k:m, k:n]-2*np.outer(W[k:m,k],(W[k:m,k].dot(R[k:m,k
                  R = np.round(R, 3)
                  W = np.round(W, 3)

              return W, R
```

```
In [540]: ### (b)

          def formQ(W):
              m = np.shape(W)[0]
              n = np.shape(W)[1]
```

```
        Q = np.identity(m)

        for k in range(0, n):
            Q_k = np.identity(m)
            Q_k[k:m, k:m] = Q_k[k:m, k:m] - 2 * np.outer(W[k:m,k], W[k:m,k])
            Q = np.dot(Q, Q_k)
            Q = np.round(Q, 3)

        return Q
```

4 Question 10: NLA 10.3

```
In [542]: ## 10.3
          Z = np.array([[1.0,2.0,3.0], [4.0,5.0,6.0], [7.0,8.0,7.0],
                        [4.0,2.0,3.0], [4.0,2.0,2.0]])
          print

          ## 1. Reduced QR using mgs in 8.2:
          Q1, R1 = mgs(Z)
          print 'QR factorization by Modified Gram-Schmidt Routine:'
          Q1 = np.round(Q1, 3)
          R1 = np.round(R1, 3)
          print 'Q1:', Q1
          print
          print 'R1:', R1
```

```
QR factorization by Modified Gram-Schmidt Routine:
Q1: [[ 0.101  0.316  0.542]
 [ 0.404  0.353  0.516]
 [ 0.707  0.391 -0.525]
 [ 0.404 -0.558  0.387]
 [ 0.404 -0.558 -0.12 ]]

R1: [[ 9.899  9.495  9.697]
 [ 0.     3.292  3.013]
 [ 0.     0.     1.97 ]]
```

```
In [541]: ## 2. Reduced QR using using 10.2:
          print
          Z = np.array([[1.0,2.0,3.0], [4.0,5.0,6.0], [7.0,8.0,7.0],
                        [4.0,2.0,3.0], [4.0,2.0,2.0]])
          W2, R2 = house(Z)
          Q2 = formQ(W2)
          print 'QR factorization by Householder Relection:'
          print 'W2:', W2
          print
```

```
print 'Q2:', Q2
print
print 'R2:', R2
```

```
QR factorization by Householder Relection:
W2: [[ 0.742  0.     0.   ]
 [ 0.272  0.787  0.   ]
 [ 0.477  0.119 -0.98 ]
 [ 0.272 -0.428  0.184]
 [ 0.272 -0.428 -0.075]]

Q2: [[-0.101 -0.315  0.543 -0.685 -0.357]
 [-0.404 -0.354  0.515  0.329  0.581]
 [-0.708 -0.389 -0.524  0.009 -0.268]
 [-0.404  0.558  0.387  0.367 -0.491]
 [-0.404  0.558 -0.121 -0.538  0.47 ]]

R2: [[-9.899 -9.495 -9.697]
 [-0.    -3.292 -3.013]
 [-0.    -0.     1.97 ]
 [-0.     0.     0.   ]
 [-0.     0.     0.   ]]
```

```
In [530]: ## 3.Reduced QR using build-in command:
          Z = np.array([[1,2,3], [4,5,6], [7,8,7],
                        [4,2,3], [4,2,2]])
          Q3, R3 = np.linalg.qr(Z, mode='reduced')
          print 'QR factorization by default:'
          print 'Q3:', Q3
          print
          print 'R3:', R3
```

```
QR factorization by default:
Q3: [[-0.10101525 -0.31617307  0.5419969 ]
 [-0.40406102 -0.3533699   0.51618752]
 [-0.70710678 -0.39056673 -0.52479065]
 [-0.40406102  0.55795248  0.38714064]
 [-0.40406102  0.55795248 -0.12044376]]

R3: [[-9.89949494 -9.49543392 -9.69746443]
 [ 0.         -3.29191961 -3.01294337]
 [ 0.          0.          1.97011572]]
```

The first and third method give basically the same result; however the result of the householder reflection method presents deviations from the other two.