Assignment 3

Arifur Rahman

400300356

Feb 11, 2023

```python
import numpy as np

def householder(A):
    m, n = A.shape
    R = np.copy(A)
    Q = np.identity(m)
    for i in range(n):
        x = R[i:, i]
        e = np.zeros_like(x)
        e[0] = np.linalg.norm(x)
        u = x - e
        v = u / np.linalg.norm(u)
        H = np.identity(m)
        H[i:, i:] -= 2 * np.outer(v, v)
        R = np.dot(H, R)
        Q = np.dot(Q, np.transpose(H))
        print(f"Step ======= {i} =========")
        print(f"Q: {Q} \n R: {R}")
    return Q, R

A = np.array([[1, -1, 4], [1, 4, -2], [1, 4, 2], [1, -1, 0]])
Q, R = householder(A)

print(f"A:\n {A}")
print(f"Q Result :\n {Q}")
print(f"R Result :\n {R}")

print(f"Varify Result (multiply Q * R) -> \n got: \n {np.dot(Q, R)} \n given: \n {A}")
```

OUTPUT:

```
Step ======= 0 =========
Q: [[ 0.5  0.5  0.5  0.5]
 [ 0.5  0.5 -0.5 -0.5]
 [ 0.5 -0.5  0.5 -0.5]
 [ 0.5 -0.5 -0.5  0.5]]
R: [[ 2.  3.  2.]
 [ 0.  0.  0.]
 [ 0.  0.  4.]
 [ 0. -5.  2.]]
```

```
Step ======= 1 =========
Q: [[ 0.5 -0.5  0.5 -0.5]
 [ 0.5  0.5 -0.5 -0.5]
 [ 0.5  0.5  0.5  0.5]
 [ 0.5 -0.5 -0.5  0.5]]
R: [[ 2.00000000e+00  3.00000000e+00  2.00000000e+00]
 [ 0.00000000e+00  5.00000000e+00 -2.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  4.00000000e+00]
 [ 0.00000000e+00 -1.11022302e-15  4.44089210e-16]]

Step ======= 2 =========
Q: [[ 0.5 -0.5  0.5  0.5]
 [ 0.5  0.5 -0.5  0.5]
 [ 0.5  0.5  0.5 -0.5]
 [ 0.5 -0.5 -0.5 -0.5]]
R: [[ 2.00000000e+00  3.00000000e+00  2.00000000e+00]
 [ 0.00000000e+00  5.00000000e+00 -2.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  4.00000000e+00]
 [ 0.00000000e+00  1.11022302e-15 -4.44089210e-16]]

A:
 [[ 1 -1  4]
 [ 1  4 -2]
 [ 1  4  2]
 [ 1 -1  0]]

Q Result :
 [[ 0.5 -0.5  0.5  0.5]
 [ 0.5  0.5 -0.5  0.5]
 [ 0.5  0.5  0.5 -0.5]
 [ 0.5 -0.5 -0.5 -0.5]]

R Result :
 [[ 2.00000000e+00  3.00000000e+00  2.00000000e+00]
 [ 0.00000000e+00  5.00000000e+00 -2.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  4.00000000e+00]
 [ 0.00000000e+00  1.11022302e-15 -4.44089210e-16]]

Varify Result (multiply Q * R) ->

 got:                             given:
 [[ 1. -1.  4.]                    [[ 1 -1  4]
 [ 1.  4. -2.]                     [ 1  4 -2]
 [ 1.  4.  2.]                     [ 1  4  2]
 [ 1. -1.  0.]]                    [ 1 -1  0]]
```