householder

September 21, 2022

```python
import numpy as np
```

```python
def e(i, length):
    # Given a position and a length, return a vector with a 1.0 in the ith
    position and 0.0 else.
    e_i = np.zeros(length, dtype=np.float64)
    e_i[i] = 1.0
    return e_i
```

```python
def reflector(v):
    """Given a unit vector v, return the householder reflector I - 2 * np.
    outer(v, v)

    Args:
        v (arr): Vector
    Returns:
        reflector (arr): A len(v) x len(v) array representing a householder
    reflector
    """
    I = np.eye(len(v))
    outer = np.outer(v, v)


    return (I - 2 * outer)
```

```python
def formQ(W):
        """Form Q from the lower diagonal vk's formed in the householder
    algorithm.

        Args:
            W (arr): An m x n matrix containing the vk in the kth column

        Returns:
            arr: Returns an m x m orthonormal matrix Q such that QR = A
        """
        m = W.shape[0]
        n = W.shape[1]
```

```
        Q = np.eye(m, m)
        for k in range(n):
                Qk = np.eye(m, m)
                Qk[k:m, k:m] = reflector(W[k:m, k])
                Q = Q @ Qk

        return np.around(Q, decimals=6)
```

```python
def house(A):
    """Compute the factor R of a QR factorization of an m x n matrix A with m
    ⌐>= .

    Args:
        A (arr): A numpy array of shape m x n
    Returns:
        R (arr): The upper diagonal matrix R in a QR factorization
    """
    m = A.shape[0] # Get row-dim of A
    n = A.shape[1] # Get col-dim of A
    W = np.zeros((m, n))
    #Q = np.eye(m, m)
    # V = np.zeros((m, n))

    # Cast A as type np.float64
    A = A.astype(np.float64)

    for k in range(n):
        # Get the first column of the (m-k+1, n-k+1) submatrix of A
        x = A[k:m, k]

        # Compute e_1, sign function, and the norm of x
        e_1 = e(0, len(x))
        sign = np.sign(x[0])
        norm_x = np.linalg.norm(x)

        # Compute vk, the vector reflected across the Householder hyperplane
        vk = (sign * norm_x * e_1) + x
        vk = vk / np.linalg.norm(vk)

        # Store vk in W
        W[k:m, k] = vk

        # Apply the reflector to the k:m, k:n submatrix of A to put
        # zeros below the diagonal of the kth column of A
        A[k:m, k:n] = reflector(vk) @ A[k:m, k:n]
```

```
        return np.around(A, decimals=6), W
```

```
A = np.array([[1, 2, 1], [4, 5, 6], [6, 9, 8]])
```

```
R, W = house(A)
print("R: \n" + str(R))
```

```
R:
[[ -7.28011  -10.439403 -10.027321]
 [ -0.         1.00939   -0.672927]
 [ -0.         0.         0.       ]]
```

```
Q = formQ(W)
print("Q: \n" + str(Q))
```

```
Q:
[[-0.137361  0.560772  0.816497]
 [-0.549442 -0.729004  0.408248]
 [-0.824163  0.392541 -0.408248]]
```

```
np.linalg.qr(A)
```

```
(array([[-0.13736056,  0.56077215, -0.81649658],
        [-0.54944226, -0.7290038 , -0.40824829],
        [-0.82416338,  0.39254051,  0.40824829]]),
 array([[-7.28010989e+00, -1.04394029e+01, -1.00273212e+01],
        [ 0.00000000e+00,  1.00938988e+00, -6.72926585e-01],
        [ 0.00000000e+00,  0.00000000e+00, -6.16000306e-15]]))
```

```
B = np.array([[2, 3, 4], [5, 6, 7], [8, 9, 10]])
print(B)
```

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

```
R, W = house(B)
print("R: \n" + str(R))
```

```
R:
[[ -9.643651 -11.199078 -12.754506]
 [  0.         0.762001   1.524002]
 [  0.        -0.        -0.       ]]
```

```
Q = formQ(W)
print("Q: \n" + str(Q))
```

```
Q:
[[-0.20739    0.889001 -0.408248]
 [-0.518476  0.254     0.816497]
 [-0.829561 -0.381    -0.408248]]
```

```
[ ]: np.linalg.qr(B)
```

```
[ ]: (array([[-0.20739034,  0.88900089,  0.40824829],
            [-0.51847585,  0.25400025, -0.81649658],
            [-0.82956136, -0.38100038,  0.40824829]]),
      array([[-9.64365076e+00, -1.11990783e+01, -1.27545058e+01],
            [ 0.00000000e+00,  7.62000762e-01,  1.52400152e+00],
            [ 0.00000000e+00,  0.00000000e+00,  1.11022302e-15]]))
```

```
[ ]: C = np.array([[1, 2], [3, 4], [5, 6]])
     R, W = house(C)
     print("R: \n" + str(R))
```

```
R:
[[-5.91608  -7.437357]
 [-0.        0.828079]
 [ 0.       -0.       ]]
```

```
[ ]: Q = formQ(W)
     print("Q: \n" + str(Q))
```

```
Q:
[[-0.169031  0.897085  0.408248]
 [-0.507093  0.276026 -0.816497]
 [-0.845154 -0.345033  0.408248]]
```

```
[ ]: np.linalg.qr(C)
```

```
[ ]: (array([[-0.16903085,  0.89708523],
            [-0.50709255,  0.27602622],
            [-0.84515425, -0.34503278]]),
      array([[-5.91607978, -7.43735744],
            [ 0.        ,  0.82807867]]))
```

```
[ ]:
```

4