Challenge Project 2

Arifur Rahman
400300356
Feb 26, 2023

```
import numpy as np

def column_convertor(x):
    x.shape = (1, x.shape[0])
    return x

def get_norm(x):
    return np.sqrt(np.sum(np.square(x)))

def householder_transformation(v):
    vector_size = v.shape[1]
    e = np.zeros_like(v)
    e[0, 0] = 1
    vector = get_norm(v) * e
    if v[0,0] < 0:
        vector = - vector
    updatedV = (v + vector).astype(np.float32)
    H = np.identity(vector_size) - ((2 * np.matmul(np.transpose(updatedV),
updatedV)) / np.matmul(updatedV, np.transpose(updatedV)))
    return H

def qr_factorization(A):
    n, m = A.shape
    Q = np.identity(n)
    R = A.astype(np.float32)
    for i in range(min(n, m)):
        v = column_convertor(R[i:, i])
        Hbar = householder_transformation(v)
        H = np.identity(n)
        H[i:, i:] = Hbar
        R = np.matmul(H, R)
        Q = np.matmul(Q, H)
    return Q, R

def backward_substitution(U, b):
    m, n = U.shape
    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        if U[i][i] == 0.0:
            return None
        sum = 0.0
        for j in range(i + 1, n):
            sum += U[i][j] * x[j]
```

```
        x[i] = (b[i] - sum) / U[i][i]
    return x

if __name__ == "__main__":
    A = np.array([[1, 0, 0, 0],
                  [0, 1, 0, 0],
                  [0, 0, 1, 0],
                  [0, 0, 0, 1],
                  [1, -1, 0, 0],
                  [1, 0, -1, 0],
                  [1, 0, 0, -1],
                  [0, 1, -1, 0],
                  [0, 1, 0, -1],
                  [0, 0, 1, -1]])

    x1 = 2.95
    x2 = 1.74
    x3 = -1.45
    x4 = 1.32

    b = np.array([x1, x2, x3, x4, 1.23, 4.45, 1.61, 3.21, 0.45, -2.75])
    Q, R = qr_factorization(A)
    print('R matrix after Householder transformation of matrix A displayed as .3
decimal places')
    print(np.around(R, decimals=3), '\n')
    print('Q matrix after Householder transformation of matrix A displayed as .3
decimal places')
    print(np.around(Q, decimals=3))

    # get the value of the altitudes
    b_hat = np.dot(Q.T, b)
    x_hat = backward_substitution(R, b_hat)
    print("Best values of the altitudes/x_hat: \n", np.around(x_hat, decimals=3))

    # #The difference between the calculated values and the direct measurements
    deltaX = np.array([x1, x2, x3, x4]) - x_hat
    print("Difference between direct measurements and calculated values/deltaX:
\n", np.around(deltaX, decimals=3))
```

OUTPUT:

```
R matrix after Householder transformation of matrix A displayed as .3 decimal
places
[[-2.     0.5    0.5    0.5  ]
 [-0.    -1.936  0.645  0.645]
 [-0.     0.    -1.826  0.913]
 [-0.     0.    -0.    -1.581]
 [-0.    -0.     0.     0.   ]
 [-0.     0.     0.    -0.   ]
 [-0.     0.    -0.     0.   ]
 [ 0.     0.     0.    -0.   ]
 [ 0.     0.    -0.     0.   ]
 [ 0.    -0.    -0.     0.   ]]

Q matrix after Householder transformation of matrix A displayed as .3 decimal
places
[[-0.5   -0.129 -0.183 -0.316 -0.46  -0.449 -0.431  0.01   0.029  0.018]
 [ 0.    -0.516 -0.183 -0.316  0.434 -0.028 -0.01  -0.462 -0.444  0.018]
 [ 0.     0.    -0.548 -0.316  0.012  0.443 -0.022  0.431 -0.034 -0.465]
 [ 0.     0.     0.    -0.632  0.014  0.034  0.462  0.02   0.449  0.428]
 [-0.5    0.387 -0.    -0.     0.723 -0.145 -0.145  0.132  0.132  0.   ]
 [-0.5   -0.129  0.365  0.    -0.132  0.723 -0.148 -0.145 -0.016  0.129]
 [-0.5   -0.129 -0.183  0.316 -0.132 -0.129  0.723  0.002 -0.145 -0.148]
 [ 0.    -0.516  0.365  0.     0.144 -0.132 -0.003  0.723 -0.147  0.129]
 [ 0.    -0.516 -0.183  0.316  0.145  0.016 -0.132 -0.129  0.723 -0.148]
 [ 0.     0.    -0.548  0.316  0.001  0.148 -0.129  0.147 -0.129  0.723]]

Best values of the altitudes/x_hat:
 [ 2.96   1.746 -1.46   1.314]

Difference between direct measurements and calculated values/deltaX:
 [-0.01  -0.006  0.01   0.006]
```