James Hahn
MATH1080
Coding Assignment #2

The Q and R matrices for the Householder and MATLAB QR factorizations are almost completely identical. One thing to note is their orthogonality checks, the matrix E, are slightly different. However, they only differ by multiples $1x10^{-15}$, which can be attributed to rounding errors in modern computing systems. For any given simulation, the bold font notes the matrix supplied in the coding assignment, as well as the QR Factorization algorithm used to compute Q and R. One key differences are the size of the matrices. Gram-Schmidt is only reduced factorization, so if A is MxN, then Q is MxN and R is NxN. Meanwhile, Householder performs full factorization, so Q is MxM and R is MxN. Besides these differences in orthogonalities and resulting dimensions of Q and R, all algorithms factor A into two matrices whose product is A, successfully.

The next 3 pages display the Q, R, and E (orthogonality check) matrices for the Z, A, and B matrices respectively. Finally, the final 3 pages showcase the code for a snippet of the driver program, original Gram-Schmidt, modified Gram-Schmidt, and Householder Triangularization.

Z, original Gram-Schmidt

```
Q (gs) =
    0.1010    0.3162    0.5420
    0.4041    0.3534    0.5162
    0.7071    0.3906   -0.5248
    0.4041   -0.5580    0.3871
    0.4041   -0.5580   -0.1204

R (gs) =
    9.8995    9.4954    9.6975
         0    3.2919    3.0129
         0         0    1.9701

E (gs) =
   1.0e-14 *

         0    0.0599   -0.0455
    0.0599         0   -0.2909
   -0.0455   -0.2909   -0.0111
```

Z, modified Gram-Schmidt

```
Q (mgs) =
    0.1010    0.3162    0.5420
    0.4041    0.3534    0.5162
    0.7071    0.3906   -0.5248
    0.4041   -0.5580    0.3871
    0.4041   -0.5580   -0.1204

R (mgs) =
    9.8995    9.4954    9.6975
         0    3.2919    3.0129
         0         0    1.9701

E (mgs) =
   1.0e-15 *

         0    0.5992   -0.3680
    0.5992         0    0.1811
   -0.3680    0.1811         0
```

Z, Householder Triangularization

```
Q (house) =
   -0.1010   -0.3162    0.5420   -0.6842   -0.3577
   -0.4041   -0.3534    0.5162    0.3280    0.5812
   -0.7071   -0.3906   -0.5248    0.0094   -0.2683
   -0.4041    0.5580    0.3871    0.3656   -0.4918
   -0.4041    0.5580   -0.1204   -0.5390    0.4695

R (house) =
   -9.8995   -9.4954   -9.6975
         0   -3.2919   -3.0129
         0         0    1.9701
         0         0   -0.0000
         0         0    0.0000

E (house) =
   1.0e-15 *

         0    0.0409    0.0461   -0.0620   -0.0143
    0.0409   -0.3331    0.1465    0.1933    0.1106
    0.0461    0.1465    0.8882   -0.2407    0.0719
   -0.0620    0.1933   -0.2407   -0.1110   -0.0436
   -0.0143    0.1106    0.0719   -0.0436   -0.1110
```

Z, MATLAB QR Factorization

```
Q (qr) =
   -0.1010   -0.3162    0.5420   -0.6842   -0.3577
   -0.4041   -0.3534    0.5162    0.3280    0.5812
   -0.7071   -0.3906   -0.5248    0.0094   -0.2683
   -0.4041    0.5580    0.3871    0.3656   -0.4918
   -0.4041    0.5580   -0.1204   -0.5390    0.4695

R (qr) =
   -9.8995   -9.4954   -9.6975
         0   -3.2919   -3.0129
         0         0    1.9701
         0         0         0
         0         0         0

E (qr) =
   1.0e-15 *

         0   -0.0179    0.0568   -0.1367   -0.0662
   -0.0179   -0.1110    0.0329    0.1356    0.1348
    0.0568    0.0329         0    0.0652    0.0916
   -0.1367    0.1356    0.0652   -0.2220   -0.2434
   -0.0662    0.1348    0.0916   -0.2434   -0.1110
```

A, original Gram-Schmidt

```
Q (gs) =
     0.5774    0.8165    0.0000
     0.5774   -0.4082    0.7071
     0.5774   -0.4082   -0.7071

R (gs) =
     1.7321    1.7321    1.7322
          0    0.0001    0.0001
          0         0    0.0001

E (gs) =
   1.0e-06 *

     0.0000   -0.0000    0.0000
    -0.0000   -0.0000    0.1154
     0.0000    0.1154         0
```

A, modified Gram-Schmidt

```
Q (mgs) =
     0.5774    0.8165    0.0000
     0.5774   -0.4082    0.7071
     0.5774   -0.4082   -0.7071

R (mgs) =
     1.7321    1.7321    1.7322
          0    0.0001    0.0001
          0         0    0.0001

E (mgs) =
   1.0e-11 *

     0.0000   -0.4710    0.2720
    -0.4710   -0.0000   -0.0000
     0.2720   -0.0000    0.0000
```

A, Householder Triangularization

```
Q (house) =
    -0.5774    0.8165    0.0000
    -0.5774   -0.4082    0.7071
    -0.5774   -0.4082   -0.7071

R (house) =
    -1.7321   -1.7321   -1.7322
          0    0.0001    0.0001
          0   -0.0000    0.0001

E (house) =
   1.0e-15 *

    -0.1110    0.1472    0.0034
     0.1472   -0.3331   -0.0414
     0.0034   -0.0414         0
```

A, MATLAB QR Factorization

```
Q (qr) =
    -0.5774    0.8165    0.0000
    -0.5774   -0.4082   -0.7071
    -0.5774   -0.4082    0.7071

R (qr) =
    -1.7321   -1.7321   -1.7322
          0    0.0001    0.0001
          0         0   -0.0001

E (qr) =
   1.0e-15 *

    -0.1110    0.0768    0.0052
     0.0768   -0.1110    0.0245
     0.0052    0.0245         0
```

B, original Gram-Schmidt	B, modified Gram-Schmidt
Q (gs) = 0.5774 0.8165 0.8165 0.5774 -0.4083 -0.4082 0.5774 -0.4083 -0.4083 R (gs) = 1.7321 1.7321 1.7321 0 0.0000 -0.0000 0 0 0.0000 E (gs) = 0.0000 -0.0000 -0.0000 -0.0000 0.0000 1.0000 -0.0000 1.0000 -0.0000	Q (mgs) = 0.5774 0.8165 0.0000 0.5774 -0.4083 0.7071 0.5774 -0.4083 -0.7071 R (mgs) = 1.7321 1.7321 1.7321 0 0.0000 0.0000 0 0 0.0000 E (mgs) = 1.0e-05 * 0.0000 -0.4710 0.2719 -0.4710 0.0000 -0.0000 0.2719 -0.0000 0.0000

B, Householder Triangularization	B, MATLAB QR Factorization
Q (house) = -0.5774 0.8165 0 -0.5774 -0.4082 0.7071 -0.5774 -0.4082 -0.7071 R (house) = -1.7321 -1.7321 -1.7321 0 0.0000 0.0000 0 -0.0000 0.0000 E (house) = 1.0e-15 * -0.1110 -0.1773 -0.0052 -0.1773 -0.8882 0.3803 -0.0052 0.3803 -0.2220	Q (qr) = -0.5774 0.8165 -0.0000 -0.5774 -0.4082 -0.7071 -0.5774 -0.4082 0.7071 R (qr) = -1.7321 -1.7321 -1.7321 0 0.0000 0.0000 0 0 -0.0000 E (qr) = 1.0e-15 * -0.1110 0.0127 -0.0034 0.0127 -0.2220 -0.1088 -0.0034 -0.1088 0

driver.m – this program is used to run the other two algorithms and test their results

```matlab
Z = [1 2 3; 4 5 6; 7 8 7; 4 2 3; 4 2 2];
A = [1 1+10^(-4) 1+2*10^(-4); 1 1 1+10^(-4); 1 1 1];
B = [1 1+10^(-10) 1+2*10^(-10); 1 1 1+10^(-10); 1 1 1];

for i = 1:3
    if i == 1
        matrix = Z;
    elseif i == 2
        matrix = A;
    elseif i == 3
        matrix = B;
    end
    [Q1, R1] = gs(matrix);
    [Q2, R2] = mgs(matrix);
    [Q3, R3] = house(matrix);
    [Q4, R4] = qr(matrix);

    [m, n] = size(matrix);

    E1 = Q1' * Q1 - eye(n);
    E2 = Q2' * Q2 - eye(n);
    E3 = Q3' * Q3 - eye(m);
    E4 = Q4' * Q4 - eye(m);

    disp("Q (gs) = ");
    disp(Q1);
    disp("R (gs) = ");
    disp(R1);
    disp("E (gs) = ");
    disp(E1);
    disp("Q (mgs) = ");
    disp(Q2);
    disp("R (mgs) = ");
    disp(R2);
    disp("E (mgs) = ");
    disp(E2);
    disp("Q (house) = ");
    disp(Q3);
    disp("R (house) = ");
    disp(R3);
    disp("E (house) = ");
    disp(E3);
    disp("Q (qr) = ");
    disp(Q4);
```

gs.m – this file implements classical Gram-Schmidt Orthogonalization for QR factorization

```matlab
function [Q, R] = gs(A)
    [m, n] = size(A);
    V = A;
    Q = zeros(m, n);
    R = zeros(n, n);
    for j = 1:n
        for i = 1:(j-1)
            R(i, j) = dot(Q(:, i), A(:, j));
            V(:, j) = V(:, j) - R(i, j) * Q(:, i);
        end
        R(j, j) = norm(V(:, j));
        Q(:, j) = V(:, j) / R(j, j);
    end
end
```

msg.m – this file implements modern Gram-Schmidt Orthogonalization for QR factorization

```matlab
function [Q, R] = mgs(A)
    [m, n] = size(A);
    V = A;
    Q = zeros(m, n);
    R = zeros(n, n);
    for i = 1:n
        R(i, i) = norm(V(:, i));
        Q(:, i) = V(:, i) / R(i, i);
        for j = (i+1):n
            R(i, j) = dot(Q(:, i), V(:, j));
            V(:, j) = V(:, j) - R(i, j) * Q(:, i);
        end
    end
end
```

house.m – this file implements Householder Triangularization for QR factorization

```
function [Q, A] = house(A)
    [m, n] = size(A);
    Q = eye(m); % the Q matrix in QR Factorization
    I = eye(m); % Identity matrix to get basis vectors later on
    for k = 1:n
        % Calculate the kth column of R with Householder
        e1 = I(1:(m-k+1), 1); % basis vector
        x = A(k:m, k);
        v = sign(x(1, :))*norm(x)*e1 + x;
        v = v / norm(v); % normalize vector
        A(k:m, k:n) = A(k:m, k:n) - 2*v*(v'*A(k:m, k:n)); % treat A like our R
        Q(k:m, :) = Q(k:m, :) - 2*v*(v'*Q(k:m, :));
    end

    A(isnan(A)) = 0; % clean the matrix and replace all NaNs with 0s
    Q(isnan(Q)) = 0; % clean the matrix and replace all NaNs with 0s
    Q = Q'; % must return the transpose of Q
end
```