James Hahn
MATH1080
Coding Assignment #1

When viewing the output, please note the Q and R matrices for both *gs* and *mgs* differ after $\varepsilon < 10^{-4}$. Despite this, if you multiply Q and R together, they are very close to accurate approximations of A, given the ε value. Both algorithms work successfully, but they just produce slightly different matrices with small ε. For every value of ε, the Q, R, and E matrices are supplied for both *gs* and *mgs*, but please take note of the parentheses next to each matrix indicating which algorithm output that matrix. At the end of the file, you can find the three code files used to compute the below results.

$\varepsilon = 10^{-2}$	**$\varepsilon = 10^{-4}$**
Q (gs) =	Q (gs) =
0.5774 0.8165 0.0000	0.5774 0.8165 0.0000
0.5774 -0.4082 0.7071	0.5774 -0.4082 0.7071
0.5774 -0.4082 -0.7071	0.5774 -0.4082 -0.7071
R (gs) =	R (gs) =
1.7321 1.7378 1.7494	1.7321 1.7321 1.7322
0 0.0082 0.0122	0 0.0001 0.0001
0 0 0.0071	0 0 0.0001
E (gs) =	E (gs) =
1.0e-10 *	1.0e-06 *
0.0000 -0.0006 0.0005	0.0000 -0.0000 0.0000
-0.0006 -0.0000 0.1554	-0.0000 -0.0000 0.1154
0.0005 0.1554 0	0.0000 0.1154 0
Q (mgs) =	Q (mgs) =
0.5774 0.8165 0.0000	0.5774 0.8165 0.0000
0.5774 -0.4082 0.7071	0.5774 -0.4082 0.7071
0.5774 -0.4082 -0.7071	0.5774 -0.4082 -0.7071
R (mgs) =	R (mgs) =
1.7321 1.7378 1.7494	1.7321 1.7321 1.7322
0 0.0082 0.0122	0 0.0001 0.0001
0 0 0.0071	0 0 0.0001
E (mgs) =	E (mgs) =
1.0e-13 *	1.0e-11 *
0.0022 -0.6279 0.5435	0.0000 -0.4710 0.2720
-0.6279 -0.0011 0.0034	-0.4710 -0.0000 -0.0000
0.5435 0.0034 0	0.2720 -0.0000 0.0000

$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-8}$
Q (gs) =	Q (gs) =
0.5774 0.8165 0.0013	0.5774 0.8165 0.8148
0.5774 -0.4082 0.7065	0.5774 -0.4082 -0.3615
0.5774 -0.4082 -0.7077	0.5774 -0.4082 -0.4533
R (gs) =	R (gs) =
1.7321 1.7321 1.7321	1.7321 1.7321 1.7321
0 0.0000 0.0000	0 0.0000 -0.0000
0 0 0.0000	0 0 0.0000
E (gs) =	E (gs) =
0.0000 -0.0000 0.0000	0.0000 -0.0000 -0.0000
-0.0000 -0.0000 0.0015	-0.0000 -0.0000 0.9979
0.0000 0.0015 0.0000	-0.0000 0.9979 0
Q (mgs) =	Q (mgs) =
0.5774 0.8165 0.0000	0.5774 0.8165 0.0000
0.5774 -0.4082 0.7071	0.5774 -0.4082 0.7071
0.5774 -0.4082 -0.7071	0.5774 -0.4082 -0.7071
R (mgs) =	R (mgs) =
1.7321 1.7321 1.7321	1.7321 1.7321 1.7321
0 0.0000 0.0000	0 0.0000 0.0000
0 0 0.0000	0 0 0.0000
E (mgs) =	E (mgs) =
1.0e-09 *	1.0e-07 *
0.0000 -0.6280 0.7252	0.0000 -0.6280 0.1813
-0.6280 -0.0000 0.0000	-0.6280 -0.0000 0.0000
0.7252 0.0000 0	0.1813 0.0000 0

$\varepsilon = 10^{-10}$

```
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
```

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

```
power = -2;
eps = 10^(power);
n = 3;
A = [1 1+eps 1+2*eps ; 1 1 1+eps ; 1 1 1];

[Q1, R1] = gs(A);
[Q2, R2] = mgs(A);

E1 = Q1' * Q1 - eye(n);
E2 = Q2' * Q2 - eye(n);

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);
```

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

```
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

```
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
```