QUESTION 1

```
clc;clear all;
%%%%%% Set up Vandermonde matrix %%%%%%
m=100;
n=15;
A=zeros(m,n);
A(:,1)=ones(m,1);
for j=1:m
    tj=(j-1)/(m-1);
    for i=2:n
        A(j,i)=tj^(i-1);
    end
end
```

1(a)

```
function [Q,R]=GS(A)
[m, n]=size(A);
assert(m>=n,'The row count must be greater than or equal to the column count')
Q=zeros(m,n);
R=zeros(n,n);
for j=1:n
    aj=A(:,j);
    vj=aj;
    for i=1:j-1
        qi=Q(:,i);
        R(i,j)=qi'*aj;
        vj=vj-R(i,j)*qi;
    end
    R(j,j)=norm(vj);
    Q(:,j)=vj/R(j,j);
end

end
```

1(b)

```
function [Q,R]=MGS(A)
[m, n]=size(A);
assert(m>=n,'The row count must be greater than or equal to the column count')
Q=zeros(m,n);
R=zeros(n,n);
for i=1:n
    vi=A(:,i);
    R(i,i)=norm(vi);
    Q(:,i)=vi/R(i,i);
```

```
            for j=i+1:n
                qi=Q(:,i);
                R(i,j)=qi'*A(:,j);
                A(:,j)=A(:,j)-R(i,j)*qi;

            end
        end
        end
```

1(C)

```
[Q,R]=GS(A);
[q1,r1]=MGS(A);
```

```
norm(A-Q*R,'inf')
```

```
ans = 1.1657e-15
```

```
norm(A-q1*r1,'inf')
```

```
ans = 7.7716e-16
```

```
norm(eye(n)-Q'*Q)  % substantial accumulation of roundoff errors
```

```
ans = 4.9871
```

```
norm(eye(n)-q1'*q1) % minimal accumulation of roundoff errors
```

```
ans = 2.1379e-07
```

COMMENT

Each vector in the classical Gram-Schmidt (CGS) method is taken one at a time and made orthogonal to all previous vectors. However, in modified Gram-Schmidt (MGS), each vector is adjusted to be orthogonal to all subsequent vectors. As a result, the CGS is more susceptible to numerical roundoff errors, thus leading to a loss of orthogonality.

Question 3

(a)

```matlab
function [V,R]=house(A)
[m, n]=size(A);
assert(m>=n,'The row count must be greater than or equal to the column count')
R=A;
V=zeros(m,n);
for k=1:n
    x=R(k:end,k);
    I=eye(m-(k-1));
    vk=x;
    vk(1)=vk(1)+sign(x(1))*norm(x);
    vk=vk/norm(vk);
    V(k:end,k)=vk;
    P=I-2*(vk*vk');
    R(k:end,k:n)=P*R(k:end,k:n);
end
end
```

(b)

```matlab
function [Q]=house2q(V)
[m, n]=size(V);
assert(m>=n,'The row count must be greater than or equal to the column count')
Q=eye(m);
for k=n:-1:1
    Q(k:m,:) = Q(k:m,:) - 2*V(k:m,k)*(V(k:m,k)'*Q(k:m,:));
end
end
```

(c)

```matlab
b=[1 2 3;4 5 6; 7 8 7; 4 2 3; 4 2 2];

[V,R]=house(b);
```

```matlab
[Q]=house2q(V);

[Q_m,R_m]=qr(b);  % MATLAB qr routine
```

```matlab
error_q = norm(Q_m-Q) % norm of difference
```

```
error_q = 7.7441e-16
```

```
error_r=norm(R_m-R) % norm of difference
```

```
error_r = 1.7014e-15
```

```
Q*R
```

```
ans = 5×3
    1.0000    2.0000    3.0000
    4.0000    5.0000    6.0000
    7.0000    8.0000    7.0000
    4.0000    2.0000    3.0000
    4.0000    2.0000    2.0000
```

```
function [V,R]=house(A)
[m, n]=size(A);
assert(m>=n,'The row count must be greater than or equal to the column count')
R=A;
V=zeros(m,n);
for k=1:n
    x=R(k:end,k);
    I=eye(m-(k-1));
    vk=x;
    vk(1)=vk(1)+sign(x(1))*norm(x);
    vk=vk/norm(vk);
    V(k:end,k)=vk;
    P=I-2*(vk*vk');
    R(k:end,k:n)=P*R(k:end,k:n);
end
end


function [Q]=house2q(V)
[m, n]=size(V);
assert(m>=n,'The row count must be greater than or equal to the column count')
Q=eye(m);
for k=n:-1:1
    Q(k:m,:) = Q(k:m,:) - 2*V(k:m,k)*(V(k:m,k)'*Q(k:m,:));
end
end




function [Q,R]=MGS(A)
[m, n]=size(A);
```

```
assert(m>=n,'The row count must be greater than or equal to the column count')
Q=zeros(m,n);
R=zeros(n,n);
for i=1:n
    vi=A(:,i);
    R(i,i)=norm(vi);
    Q(:,i)=vi/R(i,i);
    for j=i+1:n
        qi=Q(:,i);
        R(i,j)=qi'*A(:,j);
        A(:,j)=A(:,j)-R(i,j)*qi;

    end
end
end

function [Q,R]=GS(A)
[m, n]=size(A);
assert(m>=n,'The row count must be greater than or equal to the column count')
Q=zeros(m,n);
R=zeros(n,n);
for j=1:n
    aj=A(:,j);
    vj=aj;
    for i=1:j-1
        qi=Q(:,i);
        R(i,j)=qi'*aj;
        vj=vj-R(i,j)*qi;
    end
    R(j,j)=norm(vj);
    Q(:,j)=vj/R(j,j);
end
end
```