```
function [W,R]=house(A)
%householder algorithm
[m,n]=size(A);

for k=1:n
    I=k:m;
    x=A(k:m,k);
    e=zeros(m-k+1,1);
    e(1)=1;
    if x(1)==0
        V(I,k)=norm(x,2)*e+x;
    else
        V(I,k)=sign(x(1))*norm(x,2)*e+x;
    end
    V(I,k)=V(I,k)/norm(V(I,k),2);
    A(k:m,k:n)=A(k:m,k:n)-2*V(I,k)*(V(I,k)'*A(k:m,k:n));
end
W=V;
R=A(1:n,1:n);


function Q=formQ(W)

[m,n]=size(W);

for i=1:n
    x=zeros(m,1);
    x(i)=1;
    for k=n:-1:1
        x(k:m)=x(k:m)-2*W(k:m,k)*(W(k:m,k)'*x(k:m));

    end
    Q(1:m,i)=x;
end


function [Vmn,b]=newVandermonde(m,n)
%create an mxn vandermonde matrix with
%input points uniformly spaced from 0 to 1
b=zeros(m,1);
t=zeros(m,1);
Vmn=zeros(m,n);
for i=1:m
    t(i)=i*.02;
    for j=1:n
        Vmn(i,j)=t(i)^(j-1);
    end
    b(i)=cos(4*t(i));
end
```

```
function [xa,xb,xc,xd,xe,xf,xg]=leastSquaresSolver(A,b)
[m,n]=size(A);
%Normal Equations

B=A'*A;
R=chol(B);
w=R'^(-1)*A'*b;
xa=R^(-1)*w;


%QR factorization and Classical Gram-Schmidt

[Qc,Rc]=clgs(A);
xb=Rc^(-1)*Qc'*b;

%QR factorization and Modified Gram-Schmidt

[Qm,Rm]=mgs(A);
xc=Rm^(-1)*Qm'*b;

%QR factorization and Householder triangularization

[Qh,Rh]=house(A);
xd=Rh^(-1)*Qh'*b;

%QR factorization and Matlab qr

[Q,R]=qr(A,0);
xe=R^(-1)*Q'*b;

%Matlab backslash

xf=A\b;

%Matlab SVD

[U,S,V]=svd(A,0);
y=S^(-1)*U'*b;
xg=V*y;

fprintf('    Normal                CLGS                    MGS
HOUSE\n');
for i=1:n
    fprintf(' %22.15e %22.15e %22.15e %22.15e\n', xa(i),xb(i),xc(i),xd(i))
end;

fprintf('    Matlab QR             Matlab backslash        SVD\n');
for i=1:n
    fprintf(' %22.15e %22.15e %22.15e\n',xe(i),xf(i),xg(i))
end;
```