```
clc;clear all;close all;
format long;
warning off;
```

QUESTION 1

```
m=50; n=12;
t=linspace(0,1,m)';
A=t.^(0:n-1);
f=cos(4*t);

%%% 1a %%%%%
Z = A'*A;
R = chol(Z);
c1=R\(R'\(A'*f));

%%% 1b %%%%%

[Q1,R1]=GS(A);
c2=R1\(Q1'*f);

%%% 1c %%%%%

[Q2,R2]=MGS(A);
c3=R2\(Q2'*f);

%%% 1d %%%%%
[V3,R3]=house(A);
Q3=house2q(V3);
c4=R3\(Q3'*f);


%%% 1e %%%%%
[Q4,R4]=qr(A);
c5=R4\(Q4'*f);


%%%% 1f %%%%%%
[U,S,V] = svd(A,0);
S(S>0)=1./S(S>0);
c6 = (V*S*U')*f;
```

```
T= table(c1,c2,c3,c4,c5,c6,'VariableNames',{'Cholesky','GS','MGS','Householder','Matla
```

```
T = 12×6 table
                                                                                ...
```

	Cholesky	GS	MGS	Householder	Matlab-qr
1	0.99999998099...	1.000013182519...	0.99999999852...	1.00000000099...	1.00000000099...
2	0.00000527619...	-0.00225720796...	0.00000030508...	-0.0000004227...	-0.0000004227...
3	-8.0001921291...	-7.93913160486...	-8.0000085372...	-7.9999812356...	-7.9999812356...
4	0.00275327159...	-0.65191674505...	0.00008309621...	-0.0003187632...	-0.0003187631...
5	10.6461350381...	14.27119552107...	10.6663571718...	10.6694307959...	10.6694307954...
6	0.09046194227...	-11.5678311875...	0.00003886381...	-0.0138202881...	-0.0138202860...
7	-5.9406827542...	17.06808661666...	-5.6863404584...	-5.6470756271...	-5.6470756325...
8	0.45910836422...	-27.8571950089...	-0.0034568432...	-0.0753160241...	-0.0753160151...
9	1.06554610228...	22.36563134011...	1.60875341449...	1.69360696272...	1.69360695304...
10	0.46615017180...	-8.65775021725...	0.06845915645...	0.00603210972...	0.00603211622...
11	-0.5653188824...	1.289403471510...	-0.4002642010...	-0.3742417040...	-0.3742417064...
12	0.12239000974...	0.028098490547...	0.09273441558...	0.08804057620...	0.08804057660...

```
writetable(T,"table.csv")
```

Next we compare the coefficients with those of the qr and set a tolerance of 10^{-8} to pick points that will be coloured in red.

```
find(abs(c1 - c5)> 10e-8) % Find index where the absolute difference is greater than 1
```

```
ans = 11×1
     2
     3
     4
     5
     6
     7
     8
     9
    10
    11
     ⋮
```

```
find(abs(c2 - c5)> 10e-8)
```

```
ans = 12×1
     1
     2
     3
     4
     5
     6
     7
```

```
     8
     9
    10
     ⋮
```

```
find(abs(c3 - c5)> 10e-8)
```

```
ans = 11×1
     2
     3
     4
     5
     6
     7
     8
     9
    10
    11
     ⋮
```

```
find(abs(c4 - c5)> 10e-8)
```

```
ans =

  0×1 empty double column vector
```

```
find(abs(c6 - c5)> 10e-8)
```

```
ans =

  0×1 empty double column vector
```

Cholesky	GS	MGS	Householder	Matlab-qr	svd
0.9999999809927110	1.0000131825195300	0.9999999985206780	1.0000000009966100	1.0000000009966100	1.0000000009966100
0.0000052761909671	-0.0022572079677077	0.0000003050807236	-0.0000004227430466	-0.0000004227428548	-0.0000004227428473
-8.0001921291302800	-7.9391316048631600	-8.0000085372339800	-7.9999812356847700	-7.9999812356908300	-7.9999812356908500
0.0027532715936889	-0.6519167450535330	0.0000830962160406	-0.0003187632432964	-0.0003187631666744	-0.0003187631598007
10.6461350381838000	14.2711955210786000	10.6663571718952000	10.6694307959605000	10.6694307954445000	10.6694307953955000
0.0904619422776228	-11.5678311875008000	0.0000388638106883	-0.0138202881609125	-0.0138202860729090	-0.0138202858748628
-5.9406827542509000	17.0680866166607000	-5.6863404584264300	-5.6470756271566200	-5.6470756325266400	-5.6470756331175000
0.4591083642218400	-27.8571950089237000	-0.0034568432637763	-0.0753160241699216	-0.0753160151996924	-0.0753160137983506
1.0655461022852700	22.3656313401172000	1.6087534144908700	1.6936069627290400	1.6936069530485700	1.6936069515877700
0.4661501718061210	-8.6577502172563800	0.0684591564518831	0.0060321097261301	0.0060321162250397	0.0060321179910214
-0.5653188824771300	1.2894034715100400	-0.4002642010593550	-0.3742417040194980	-0.3742417064829500	-0.3742417069654390
0.1223900097474410	0.0280984905474899	0.0927344155809599	0.0880405762040207	0.0880405766060742	0.0880405766782853

Observation

The figure above shows the values (coloured in red) that appear wrong. The qr coefficients were used as a yardstick for determining what is wrong or not . It can be observed that the clasisical Gram-Schmidt method

(GS) is the most affected due to roundoff errors, next is the modified Gram-Schmidt and the Cholesky methods. Using a tolerance of 10^{-8} , the other methods appear to be right. It appears that the Cholesky and qr looks close to each other for the first four coefficients.

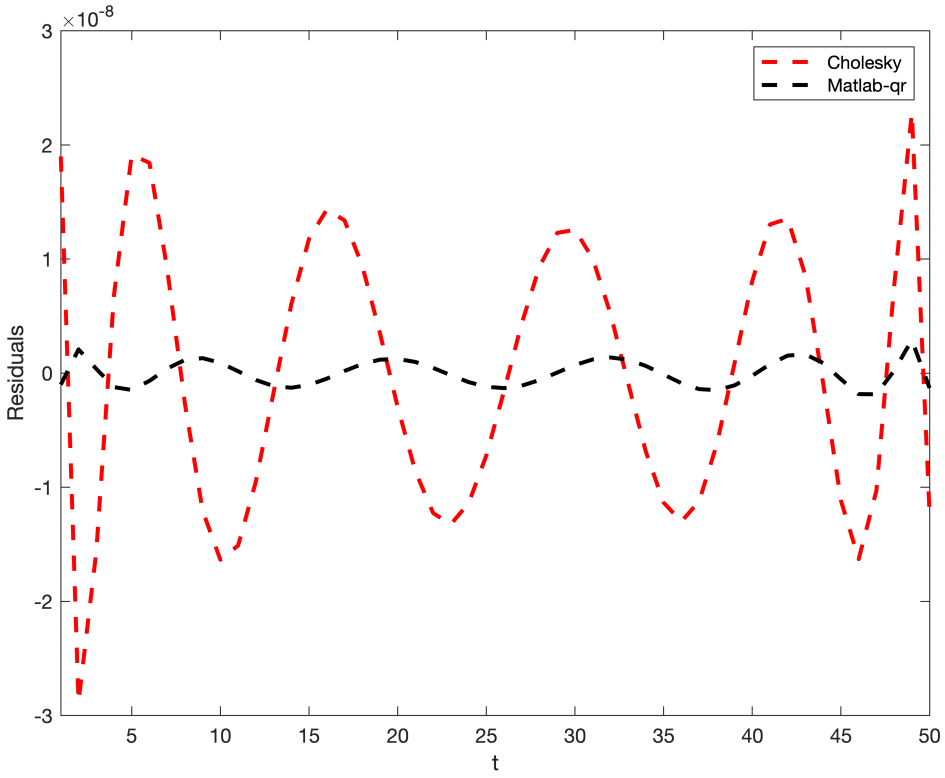
```
figure(1)

res10 = f-A*c1; % matlab Cholesky routine

res1 = f-A*c5;  % matlab qr routine

plot(res10,'--r', LineWidth=2), hold on

plot(res1,'--k', LineWidth=2)
xlim([1,size(f,1)])
legend('Cholesky', 'Matlab-qr')
xlabel('t')
ylabel('Residuals')
```




```
max(abs(res1))  % qr
```

```
ans =
     2.919108399446202e-09
```

```
max(abs(res10)) % cholesky
```

```
ans =
     2.879320337711988e-08
```

observation

From the Figure above and our observation of the maximum absolute residual in the qr and cholesky methods which is approximately $3e^{-09}$ and $3e^{-08}$ respectively, the qr factorization provides a better approximation.

Question 2

```
%%%%%%%%%%% 2a %%%%%%%%%%%%%%%%%%%%
```

$\|r\|_w^2 = (\mathrm{Ax} - b)^T W(\mathrm{Ax} - b)$

$= (x^T A^T - b^T)W(\mathrm{Ax} - b)$

$= x^T A^T \mathrm{WAx} - x^T A^T \mathrm{Wb} - b^T \mathrm{WAx} + b^T \mathrm{Wb}$

Taking the derivative with respect to x and equating to zero, we have;

$$\frac{d(||r||_w^2)}{dx} = 2A^T WA - A^T Wb - b^T WA = 0$$

But, $A^T \mathrm{Wb} = b^T \mathrm{WA}$

$2A^T \mathrm{WAx} = 2A^T \mathrm{Wb}$

$\mathrm{x} = (A^T \mathrm{WA})^{-1} A^T \mathrm{Wb}$

```
%%%%%%%%%%% 2b %%%%%%%%%%%%%%%%%%%%
```

```
W = zeros(m);
delta = 0.8;
for i = 1:m
    W(i,i) = exp(-(abs(1/23 - t(i))/delta)^2);
end
```

```
x_w = (A'*W*A)\(A'*W*f)  % weighted coefficients
```

```
x_w = 12×1
   1.000000000430680
  -0.000000194863716
  -7.999990898790073
  -0.000162651213323
```

```
  10.668141631982348
  -0.007615972142881
  -5.665673569551202
  -0.039552661117411
   1.649483340002800
   0.039790819505998
      ⋮
      ⋮
```

```
% Compute the Vandermonde matrix at a fixed t  = 1/23

m=50; n=12;
t=(1/23)*ones(m,1);
A=t.^(0:n-1);
```

```
P_weighted = A*x_w; % weighted polynomial
P_w_11 = P_weighted(11)
```

```
P_w_11 =
   0.984915205156389
```

```
P_Non_weighted = A*c5;   % non weighted polynomial
P_11 = P_Non_weighted(11)
```

```
P_11 =
   0.984915205008980
```

```
f_t = cos(4*1/23)
```

```
f_t =
   0.984915205128733
```

Comparing the weighted and non-weighted results

```
abs(P_w_11 - f_t)      % Error in weighted least squares
```

```
ans =
     2.765643269952989e-11
```

```
abs(P_11 - f_t)        % Error in non-weighted least squares
```

```
ans =
     1.197527632612605e-10
```

From the results of the error produced by the weighted and non-weighted least squares, we can conclude that the weighted least squares produce a better approximation of $f(t) = cos(4t)$ at $t = 1/23$.

Question 3

$$\kappa = \frac{\frac{||J(X)||}{||f(x)||}}{||x||}$$

$$f(x) = \frac{log(x+1)}{x}$$

$$f'(x) = \frac{x-(x+1)log(x+1)}{x^2(x+1)}$$

$$\kappa = \frac{|x-(x+1)log(x+1)|}{|(x+1)x^2|} \times \frac{|x^2|}{|log(x+1)|}$$

$$\kappa = \frac{|x-(x+1)log(x+1)|}{|(x+1)log(x+1)|}$$

```
%%%%%%%%%%%%% 3a  %%%%%%%%%%%%%%%%%%
kappa = @(x)abs(x-log(x+1)*(x+1))/abs(log(x+1)*(x+1));
x=-1e-7:1e-8:1e-7;
xk=zeros(size(x));
for i=1:length(xk)
    xk(i)= kappa(x(i));
end
xk
```

```
xk = 1×21
10^-7 ×
   0.505263599926700   0.449095685898494   0.405263585164722   0.347333412191465 ···
```

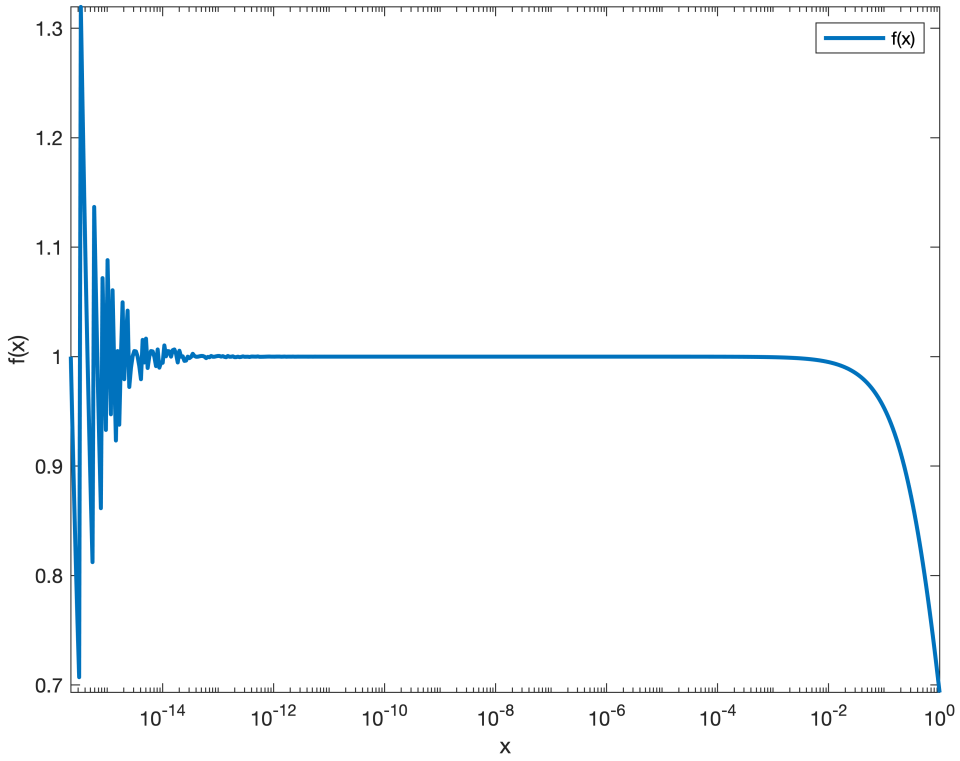
Observation

It can be obsereved that the condition number of the points near zero are very small in the order of 10^{-7}. Thus we can conclude that when the input is changed by a small amount, the corresponding output will be changed by a small amount.

```
%%%%%%%%%%% 3b %%%%%%%%%%%%%%%%
f=@(x)(log(x+1)./x);
j=0:1:520;
xj=2.^(-52+j./10);

figure(5)
```

```
semilogx(xj,f(xj), LineWidth=2)
xlabel('x')
ylabel('f(x)')
legend('f(x)')
axis tight
```



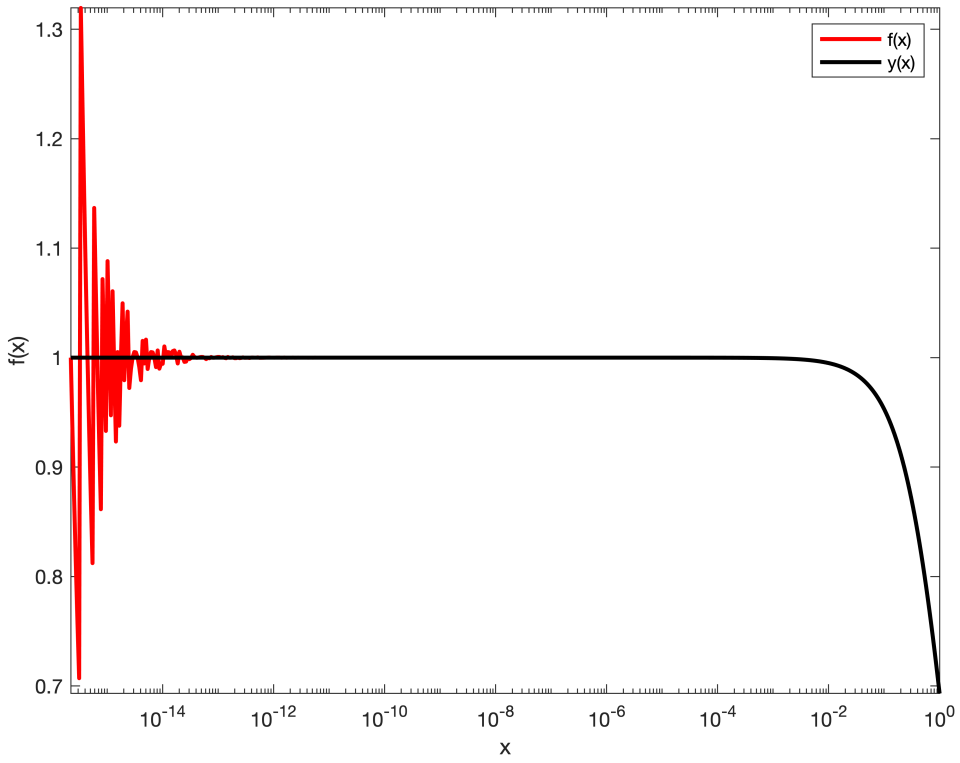
Observation

$f(x)$ appears to be unstable near x = 0.

```
%%%%%%%%% 3c %%%%%%%%%%%%%
z =1+xj;
y = @(z)(log(z)./(z-1));

figure(6)

semilogx(xj,f(xj),'r',linewidth=2), hold on
semilogx(xj,y(z),'k',linewidth=2)
hold off;
xlabel('x')
ylabel('f(x)')
legend('f(x)','y(x)');
axis tight
```



Observation

$y(x)$ appear to be stable near zero while $f(x)$ is unstable.

Question 4

```
%%%%%% 4a %%%%%%%%%%
```

Let us consider the equation, $\frac{||\delta x||}{||x||} \leq \kappa \frac{||r||}{||b||}$

Looking at the right hand side, we observe that κ is the dominant factor in determining how the output will change with respect to a perturbation in the input. Thus we conclude that the size of the residual does not give us enough information about how close the solution is to the exact but the condition number κ does.

```
%%%%%% 4b %%%%%%%%%%

A = 1./hankel(2:6,6:10);
b = [0.882 0.744 0.618 0.521 0.447]';
x = A\b
```

x = 5×1

```
   -2.519999999999232
    5.039999999991446
    2.520000000028753
    7.559999999962587
  -10.079999999983430
```

```
%%%%%% 4c %%%%%%%%%%%
```

$A\hat{x}$= b + δb

$\hat{x}$ = $A^{-1}b$ + $A^{-1}\,\delta b$

$\hat{x}$ = x + $A^{-1}\,\delta b$

$\hat{x}$ = x + δx

δx = $\hat{x}$ - x

```
kapp = cond(A)
```

```
kapp =
     1.535043895304634e+06
```

```
% small pertubation of b

db = 1e-4:2.5e-5:2*1e-4;

x_cap = x + A\db';

dx =x_cap - x;

% Upper bound
kappa_db_b = kapp*norm(db)/norm(b)
```

```
kappa_db_b =
     3.578889287846920e+02
```

```
dx_d=norm(dx)/norm(x)
```

```
dx_d =
   0.171793173727223
```

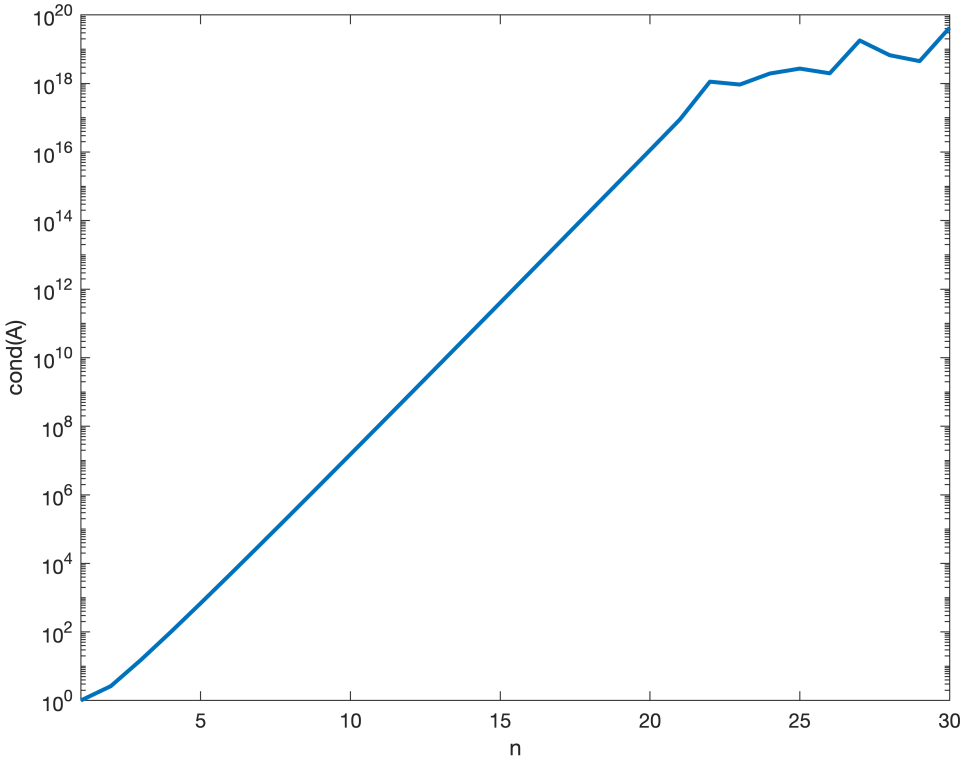
Observation

First we observe that the condition number (κ) is really large and the product of the relative error in b and κ is a lot greater than the relative error in x. This implies that a small perturbation in the input will lead to a large perturbation in the output.

QUESTION 5

```
n=30;
kapp = zeros(n,1)';
for i = 1:length(kapp)
    kapp(i) = cond(vandermonde(i,i));
end
```

```
figure(7)
semilogy(1:1:n,kapp,linewidth=2)
xlim([1,30])
xlabel('n')
ylabel('cond(A)')
```

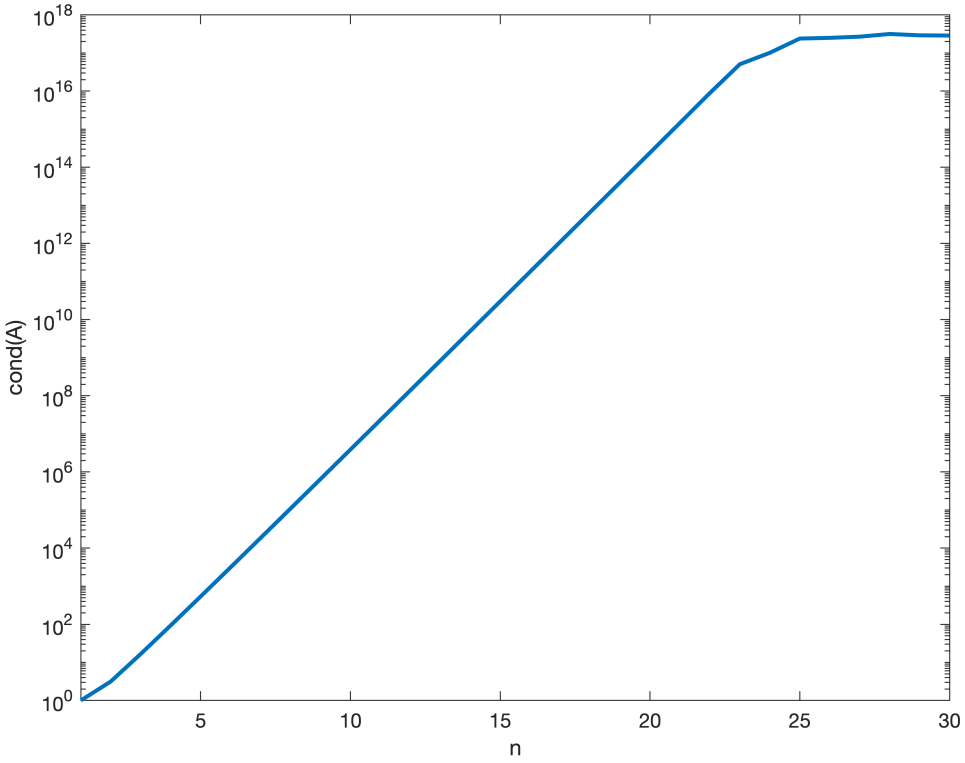


Observation

The condition number appears to grow linearly from n = 1 until n = 23 where it begins to fluctuate. This could be attributed to the intrinsic nature of the matrix which is ill-conditioned.

```
n=30;
kapp2 = zeros(n,1)';
for i = 1:length(kapp)
    kapp2(i) = cond(vandermonde(2*i-1,i));
end
```

```
figure(8)
semilogy(1:1:n,kapp2,linewidth=2)
xlim([1,30])
xlabel('n')
ylabel('cond(A)')
```

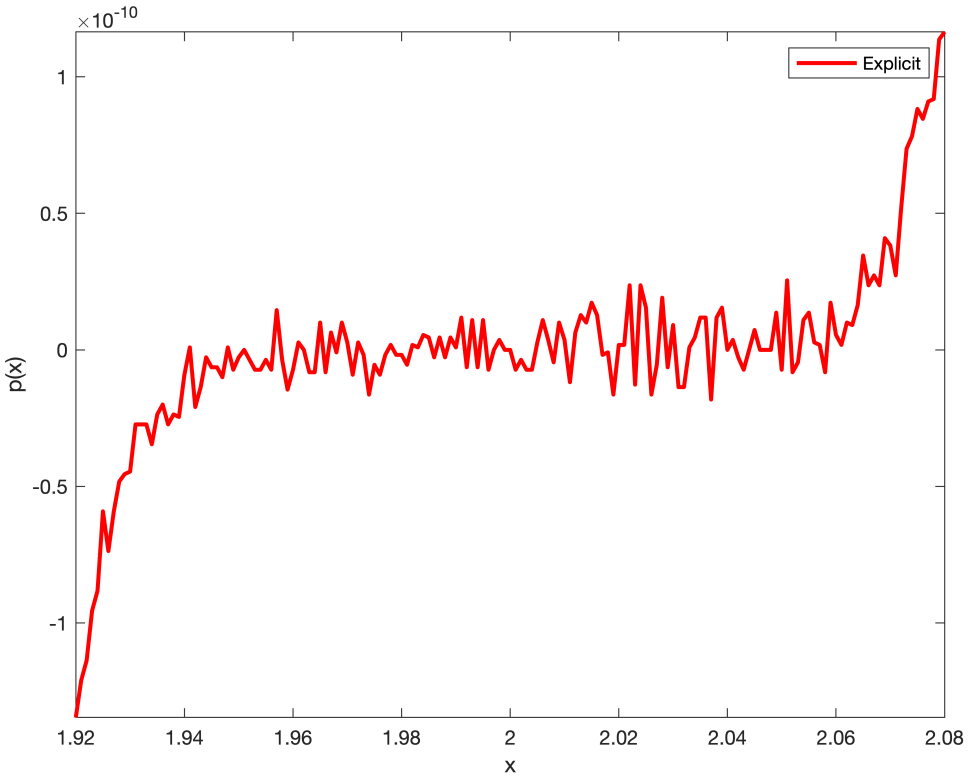


Observation

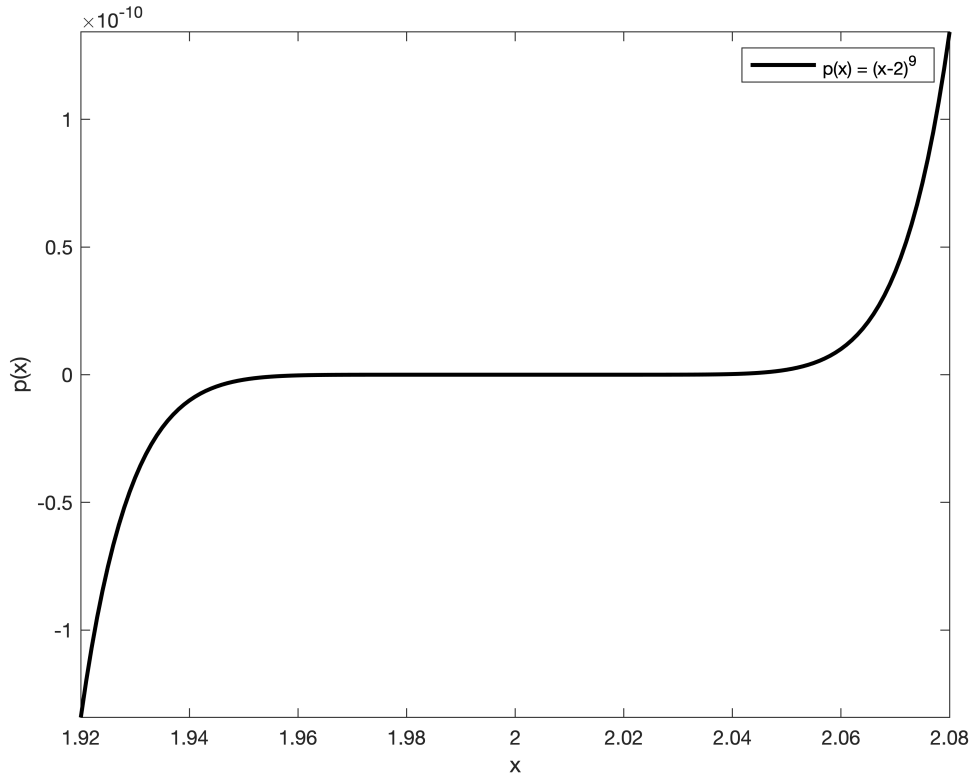
The condition number appears to grow linearly from n = 1 until n = 25 where it reaches steady state.

QUESTION 6

```
x=1.920:0.001:2.080;
r=zeros(size(x));
for i=1:length(x)
    r(i)=p(x(i));
end
figure(9)
plot(x,r,'r',LineWidth=2)
xlabel('x')
ylabel('p(x)')
legend('Explicit')
axis tight
```




```
r_q=(x-2).^9;
figure(10)
plot(x,r_q,'k',LineWidth=2)
xlabel('x')
ylabel('p(x)')
legend('p(x) = (x-2)^9')
axis tight
```



Observation

We observe that the figure produced using $p(x) = (x-2)^9$ is smoother when compared to the figure obtained from the explicit expression of $p(x)$. This is due to the accumulation of roundoff errors in the explicit case.

QUESTION 7

P is a permutation matrix, it is orthogonal and symmetric. Thus $P^2 = P$.

```
P=zeros(10);
P(1,end) = 1; P(2,5) = 1; P(3,8) = 1; P(4,1) = 1;
P(5,4) = 1; P(6,9) = 1; P(7,2) = 1; P(8,3) = 1; P(9,6) = 1; P(10,7)= 1;
```

```
skeel_K = norm(P)
```

```
skeel_K =
     1
```

```
Kappa_P = cond(P)
```

```
Kappa_P =
     1
```

```
P(:,3)=10e-11*P(:,3);
skeel_K = norm(abs(inv(P))*abs(P))
```

```
skeel_K =
     1
```

```
kappa_P = cond(P)
```

```
kappa_P =
     1.000000000000000e+10
```

It can be observed that the skeel condition number is always less than or equal to κ and it is invariant of the row or column scaling.

```
function  r = p(x)
    r = x^9 -18*x^8 + 144*x^7 - 672*x^6 + 2016*x^5 -4032*x^4 + 5376*x^3 - 4608*x^2 + 2
end

function A = vandermonde(m,n)
 t=linspace(0,1,m)';
 A=t.^(0:n-1);
end

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
```