**%Q1**

a = [32:2:74]

**%Q2**

x = [ 2 5 1 6 ]

% a

a = x + 16

% b

b = x

for i=1:numel(x)

if rem(i,2)==1

b(i)=x(i)+3;

end

end

b

% c

c = sqrt(x)

% d

d = x.^2

**%Q3**

% defining row arrays x, y

x = [3 2 6 8]

y = [4 1 3 5]

% taking transpose

x = x.';

y = y.';

% saving x and y to recover them after manipulation

x1 = x ;

y1 = y ;

% a

y = y + x

y = y1;

x = x1;

% b

x = x.^y

y = y1;

x = x1;

% c

y = y./x

y = y1;

x = x1;

% d

z = x.\*y

% e

w = sum(z)

% f

%dot product of x and y subracted with w

x'\*y - w

**%Q4**

%a

x=2/2\*3

% / has more priority than \*

%b

x=6-2/5+7^2-1

% ^ has more priority than /

%c

x=10/2\5-3+2\*4

% / has more priority than \, i.e. division in reverse

%d

x=3^2/4

% ^ has more priority than /

%e

x=3^2^2

%

%f

x = 2+round(6/9+3\*2)/2-3

% round to upper integer if value >0.5

% round to lower integer if value <0.5

%g

x = 2+floor(6/9+3\*2)/2-3

% round to lower integer

%h

x = 2+ceil(6/9+3\*2)/2-3

% round to upper integer

**%Q5**

%take inputs from user and define vector

n = input('type number of elements in vector : ');

n1 = input('type beginning value of vector : ');

n2 = input('type ending value of vector : ');

t =[n1:(n2-n1)/(n-1):n2]

%a

a=log(2+t+t.^2)

%b

b=exp(t).\*(1+cos(3\*t))

**%Q6**

% time

t = [1790:2000];

% population

p = 197273000./(1+exp(-0.0313.\*(t-1913.25)));

% plotting

plot(t,p)

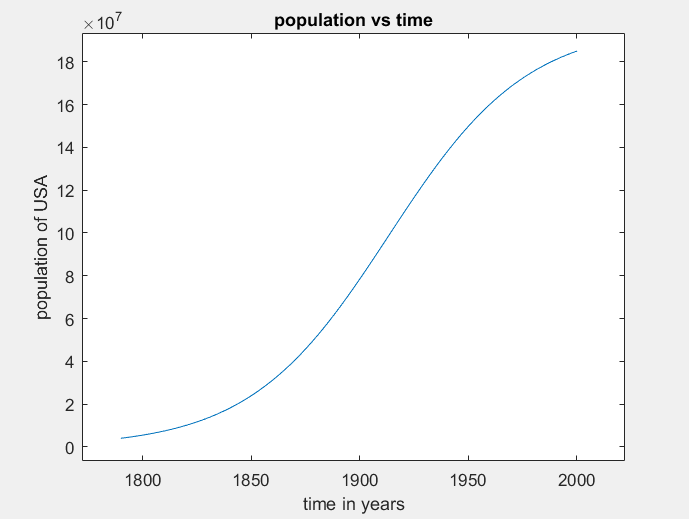
title('population vs time')

xlabel('time in years')

ylabel('population of USA')

hold

p2020 = 197273000/(1+exp(-0.0313\*(2020-1913.25)))



**%Q7**

% defining x

% to get non choppy graph steps should be small enough

x=[0.01:0.00001:0.1];

%plotting

figure()

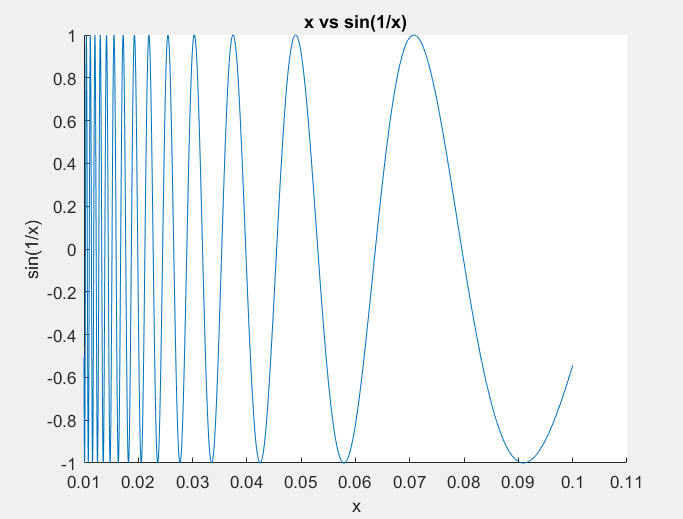
hold()

title('x vs sin(1/x)')

plot(x,sin(1./x))

xlabel('x')

ylabel('sin(1/x)')



**%Q8**

x = [3 1 5 7 9 2 6];

%a

x(3) % 3rd element

%b

x(1:7) % 1st to 7th elements of x

%c

x(1:end) % 1st to last elements of x

%d

x(1:end-1) % 1st to last but one elements of x

%e

x(6:-2:1) % 6th element to 1st element with steps of 2 in x

%f

x([1 6 2 1 1]) % the respective elements of x

%g

sum(x) %sum of elements in x

**%Q9**

A=[2 4 1 ; 6 7 2 ; 3 5 9 ]

%a

x1 = A(1,:) % assigning first row to x1

%b

y = A(2:3,:)% assigning last two rows to y

%c

c = sum(A,1)% sum of columns of A

%d

d = sum(A,2)

%e

%standard error = standard deviation / sqrt(n)

for i=1:3 % for different columns

se(i) = std(A(:,i))/sqrt(3);

end

se

**%Q10**

x = [1 4 8]

y = [2 1 5]

A = [3 1 6 ; 5 2 7]

%a

a = x + y %works

%b

b = x + A % matrix dimension not matching

%c

c = x' + y % matrix dimension not matching

%d

d = A - [x' y'] % matrix dimension not matching

%e

e = [x ; y'] % matrix dimension not matching

%f

f = [x ; y] % works

%g

g = A - 3 % works

**%Q11**

A = [2 7 9 7 ; 3 1 5 6 ; 8 1 2 5]

%a

a = A' % transpose

%b

b = A(:,[1 4]) %1st and 4th columns

%c

c = A([2 3],[3 1]) % elements 2,2 ;2,3 ;3,3 ;3,1

%d

d = reshape(A,2,6) % arrangng into 2 x 6 matrix column-wise

%e

e = flipud(A) % flips columns with mirrors

%f

f = fliplr(A) % flips rows

%g

g = [A ; A(end,:)]

%h

h = A(1:3,:) % entire matrix

%i

i = [A ; A(1:2,:)] % repeatition of first two columns

%j

j = sum(A) % sum of each column

%k

k = sum(A')% sum of each row

%l

l = sum(A,2)

%m

m = [ [ A ; sum(A) ] [ sum(A,2) ; sum(A(:)) ] ]

**%Q12**

z = input('enter the value of z : ');

if z<5

w = 2\*z;

elseif z<10

w = 9-z;

elseif z<100

w = sqrt(z);

else

w = z;

end

w

%z=1 w = 2

%z=9 w = 0

%z=60 w = 7.7460

%z=200 w = 200

**%Q13**

x = [4 1 6]

y = [6 2 7]

%a

a = x>y

%b

b = y<x

%c

c = x==y

%d

d = x<=y

%e

e = y>=x

%f

f = x|y

%g

g = x&y

%h

h = x&(~y)

%i

i = (x>y)|(y<x)

%j

j = (x>y)&(y<x)

%k

k = x'\*y

%l

l = x'./y

%m

m = sum(x.\*y)

%n

for i = 1:3

if x(i)>y(i)

n(i)= y(i);

else

n(i) = x(i);

end

n(i)= 1/n(i);

end

n

**%Q14**

x = 1:10

y = [3 1 5 6 8 2 9 4 7 0]

%a

a = (x > 3) & (x < 8) % true when greater than 3 and less than 8

%b

b = x(x > 5) % vector part of x when x greater than 5

%c

c = y(x <= 4)% vector part of y when x less than or equal to 4

%d

d = x( (x < 2) | (x >= 8) ) % vector part of x when x greater than or equal

% to 8 or less than 2

%e

e = y( (x < 2) | (x >= 8) ) % vector part of y when x greater than or equal

% to 8 or less than 2

%f

f = x(y < 0) % empty matrix

**%Q15**

x = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]

y = [ 9 8 7 ; 6 5 4 ; 3 2 1 ]

%a y=rem(x,2)vs.y=logical(rem(x,2))

yrem=rem(x,2)

ylogical=logical(rem(x,2))

%b y=~rem(x,2)vs.y=~logical(rem(x,2))

nyrem=~rem(x,2)

nylogical=~logical(rem(x,2))

**%Q16**

F=input('Input temperature in Fahrenheit Scale : ');

C=(F-32)\*5/9;

disp(sprintf('Temperature in Celsius Scale = %f',C));

**%Q17**

sum = 0;

count = 0;

while sum < 20

count = count + 1;

x = rand;

sum = sum + x;

end

disp(['To add more than 20 it number of random numbers taken is ',int2str(count)])

count = 0;

x=rand;

while x < 0.8 || x > 0.85

count = count + 1;

x = rand;

end

disp(['Number of iterations to get random number between 0.8 and 0.85 is ',int2str(count)])

count = 0;

mean =0;

while abs(mean - 0.5) > 0.01

count = count + 1;

x = rand;

mean = ((count-1)\*mean + rand(1,1))/count;

end

disp(['Number of iterations for mean to be within 0.01 of 0.5 is ',int2str(count)])

**%Q18**

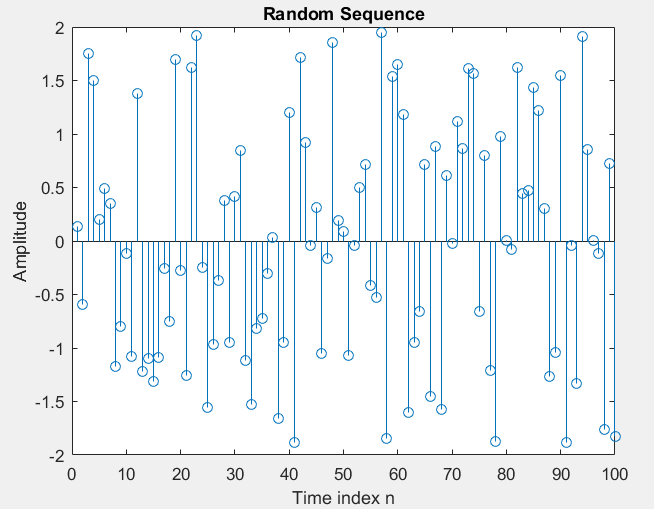
%random number between -2 to 2

ans = 4\*rand(1,100)-2;

stem(ans)

title('Random Sequence');

xlabel('Time index n');

ylabel('Amplitude');

**%Q19**

n = 0:74;

mean = 0;

std = sqrt(3);

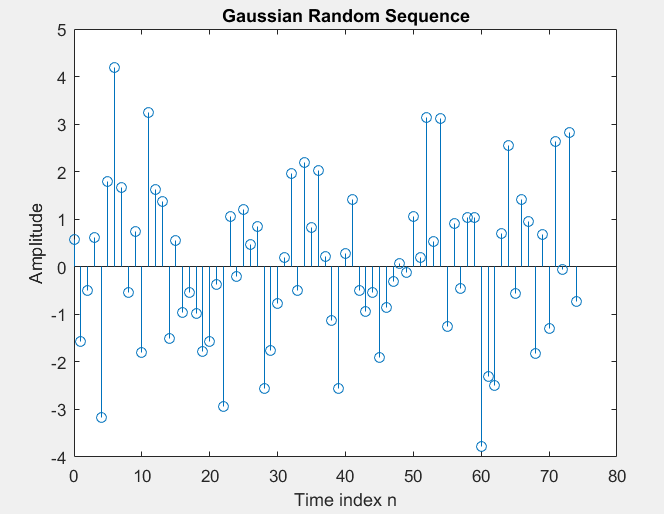
x = std\*randn(1,length(n)) + mean;

stem(n,x);

title('Gaussian Random Sequence');

xlabel('Time index n');

ylabel('Amplitude');



**%Q20**

n=0:30;

f=0.1;

maximumAmplitude=4;

maximumPhase=2\*pi;

Amplitude=maximumAmplitude\*rand;

phase=maximumPhase\*rand;

arg=2\*pi\*f\*n+phase;

x=Amplitude\*cos(arg);

stem(n,x);

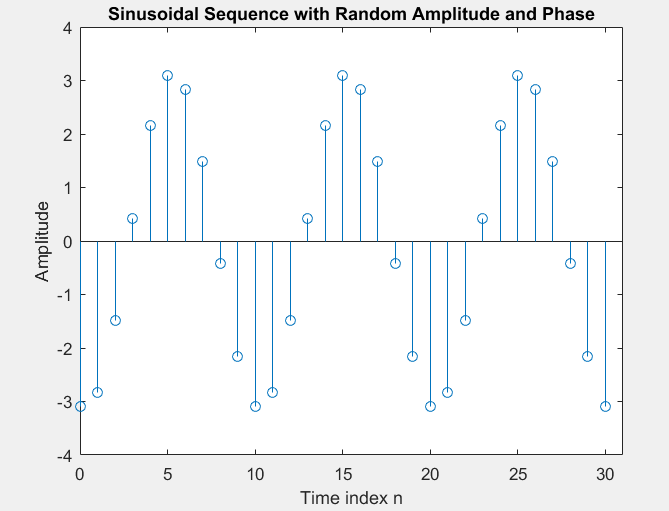
Ylimit = 4;

axis([0 length(n) -Ylimit Ylimit]);

title('Sinusoidal Sequence with Random Amplitude and Phase');

xlabel('Time index n');

ylabel('Amplitude');



**%Q21**

length=0;

n=input('enter: ');

while n>1

if rem(n,2)==0

n=n/2;

else

n=3\*n+1;

end

length=length+1;

end;

length

**%Q22**

n = -10:10;

%a) Impulse signal

figure()

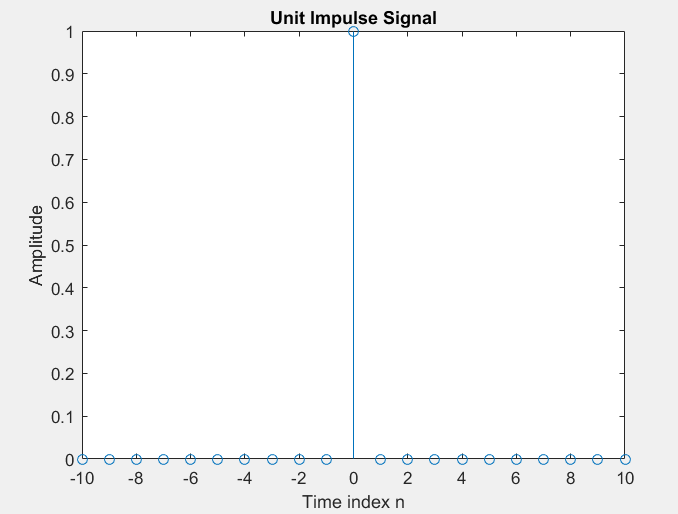
delta = [zeros(1,10) 1 zeros(1,10)];

stem(n,delta);

xlabel('Time index n');

ylabel('Amplitude');

title('Unit Impulse Signal');



%b) Step signal

figure()

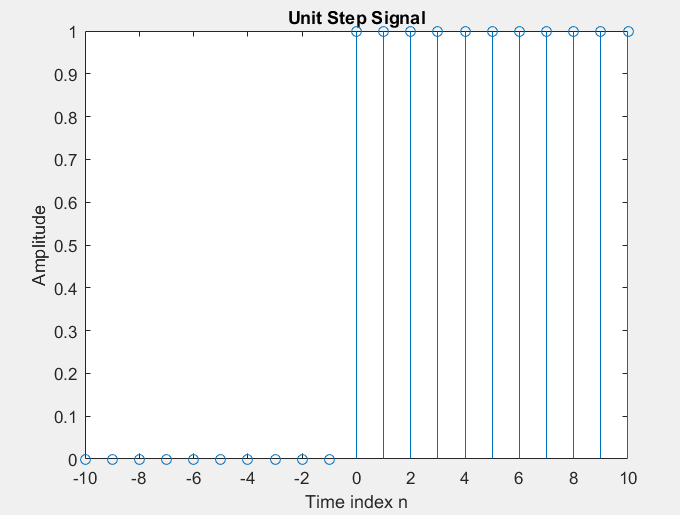
delta = [zeros(1,10) ones(1,11)];

stem(n,delta);

xlabel('Time index n');

ylabel('Amplitude');

title('Unit Step Signal');



%c) Ramp signal

n=0:10;

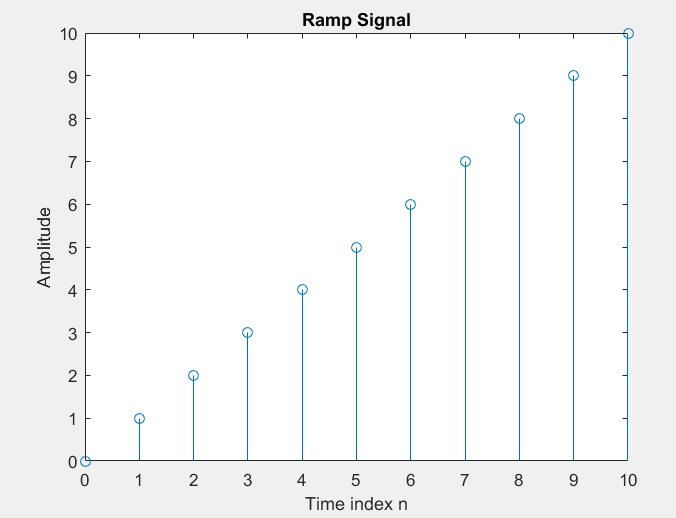
ramp=n;

figure()

stem(n,ramp)

xlabel('Time index n');

ylabel('Amplitude');

title('Ramp Signal');

%d) Real exponential signal

a=0.9;

e=exp(-a.\*n);

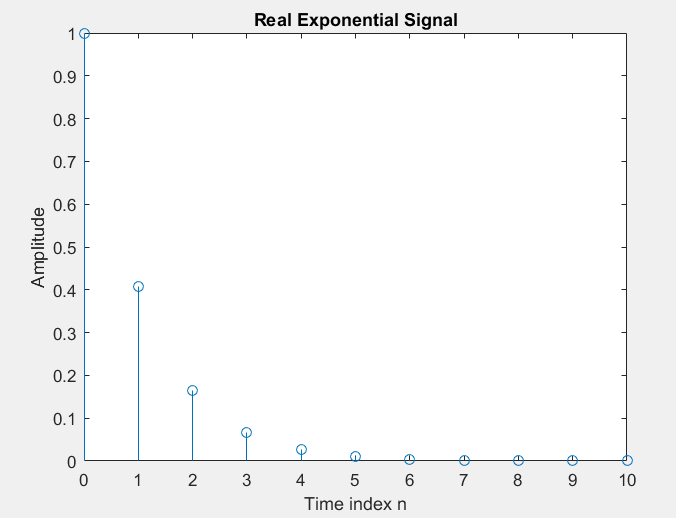
figure()

stem(n,e)

xlabel('Time index n');

ylabel('Amplitude');

title('Real Exponential Signal');



%e) Shifted impulse

n=-10:10;

n1 = (n(1)+11):(n(numel(n))+11)

u = [zeros(1,10) 1 zeros(1,10)];

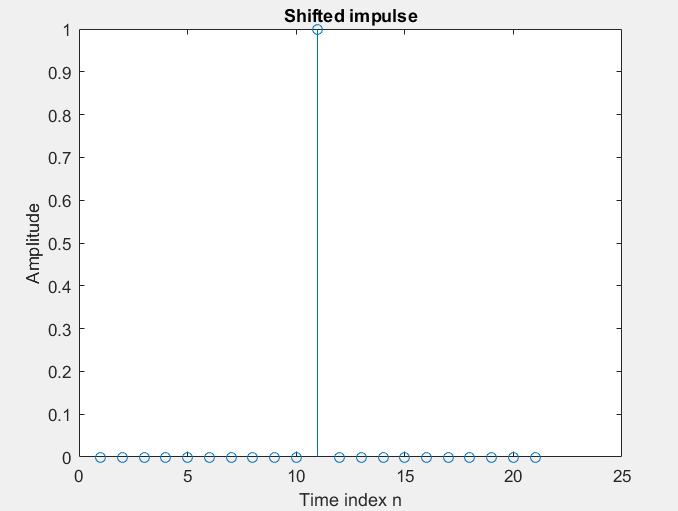
figure()

stem(n1,u);

xlabel('Time index n');

ylabel('Amplitude');

title('Shifted impulse');



%f) Advanced impulse

n2 = (n(1)-7):(n(numel(n))-7)

u = [zeros(1,10) ones(1,11)];

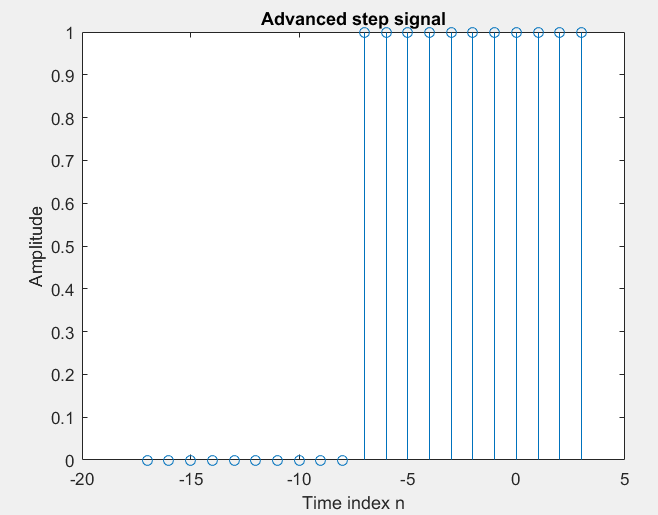
figure()

stem(n2,u)

xlabel('Time index n');

ylabel('Amplitude');

title('Advanced step signal');



**%Q23**

%a

resolution = 0.01;

n = -10:resolution:10;

a = 3\*sin(2\*pi\*n + pi/3);

figure()

hold()

plot(n,a,'b')

xlabel('n')

ylabel('Amplitude')

%b

b = 5\*cos(2\*pi\*n/3 + pi/4) + 2.5\*sin(pi\*n/3 + pi/4);

plot(n,b,'r')

legend('first','second')

**%Q24**

r=0.8;

n=0:20;

y=r.^n.\*exp(j\*pi.\*n/3);

plot(n,abs(y))

xlabel('n');

ylabel('Magnitude of y');

figure()

plot(n,angle(y))

xlabel('n')

ylabel('Angle of y')

