sampling theorem

```
clc;
clear all;
t=0:0.001:0.1;
fm=input("enter the frequency :");
x=sin(2*pi*fm*t);
subplot(4,1,1)
plot(t,x);
title('sine wave');
xlabel('time');
ylabel('amplitude');

fs=10
n=0:(1/fs):0.1;
xn=sin(2*pi*fm*n);
subplot(4,1,2)
stem(n,xn);
title('fs<<2fm');
xlabel('time');
ylabel('amplitude');

fs=30
n=0:(1/fs):0.1;
xn=sin(2*pi*fm*n);
subplot(4,1,3)
stem(n,xn);
title('fs=2fm');
xlabel('time');
ylabel('amplitude');


fs=60
n=0:(1/fs):0.1;
xn=sin(2*pi*fm*n);
subplot(4,1,4)
stem(n,xn);
title('fs>>2fm');
xlabel('time');
ylabel('amplitude');
```

sautooth square wave

```
f=input("Enter the Frequency");
t=-2:0.01:5;
s=square(t)
figure;
subplot(2,1,1)
plot(t,s)
title('square Wave');
xlabel('Time');
ylabel('Amplitude');

y=sawtooth(2*pi*f*t);
subplot(2,1,2)
plot(t,y);
title('Sawtooth Wave');
xlabel('Time');
ylabel('Amplitude');
```

cos sin +exponential - exponential signal

```
clc;
clear all;
close all;
t=0:0.001:1;
f=input("Enter the Frequency");
a=input("Enter the Amplitude");
s=a*sin(2*pi*f*t);
subplot(2,2,1)
plot(t,s)
title('Sine Wave');
xlabel('Time');
ylabel('Amplitude');


s=a*cos(2*pi*f*t);
subplot(2,2,2)
plot(t,s)
title('Cos Wave');
xlabel('Time');
ylabel('Amplitude');

s=a*exp(a*t);
subplot(2,2,3)
plot(t,s)
title('+ve Exp Wave');
xlabel('Time');
ylabel('Amplitude');

s=a*exp(-a*t);
subplot(2,2,4)
plot(t,s)
title('-ve Exp Wave');
xlabel('Time');
ylabel('Amplitude');
```

```
s=a*exp(-a*t);
subplot(2,2,4)
plot(t,s)
title('-ve Exp Wave');
xlabel('Time');
ylabel('Amplitude');

s=square(t)
figure;
subplot(2,1,1)
plot(t,s)
title('square Wave');
xlabel('Time');
ylabel('Amplitude');
```

sine wave (kit)

```
#include  <stdio.h>
#include <math.>
#define FREQ 500
 float m[128];
 main()
{ int i=0; for(i=0;i<127;i++) {
m[i]=sin(2*3.14*FREQ*i/24000)
; printf("%f\n",m[i]);
 }
 }
```

twiddle factor

```
clc;
clear all;
close all;
n=16;
x=randi(10,n,1)+1i.*randi(10,n,1);
tic;
wn=exp(-(j*2*pi)/n);
for a=1:1:n
   for j=1:1:n
      v_n(a,j)=wn.^((a-1)*(j-1));
   end
end
disp('Twiddle Factor Matrix');
v_n;
y=v_n*x;
time(1)=toc;
v_real=real(v_n)
v_imaginary=imag(v_n)
subplot(2,1,1)
imshow(v_real)
title("Real  Part of Twiddle factor
matrix")
subplot(2,1,2)
imshow(v_imaginary)
title("Imaginary  Part of Twiddle
factor matrix")
```

linear and circular convolution

```
clc;
clear all;
u=[3 2 1 2]
v=[1 2 1 2]
k=conv(u,v);
subplot(3,1,1);
stem(u);
subplot(3,1,2);
stem(v);
subplot(3,1,3);
stem(k);
xlabel('amplitude');
ylabel('time');
title('linear convolution');

figure;
clc;
clear all;
n=[3 2 1 2]
m=[1 2 1 2]
i=length(n);
t=length(m);
c=max(i,t);
h=cconv(n,m,c);
subplot(3,1,1);
stem(n);
subplot(3,1,2);
stem(m);
subplot(3,1,3);
stem(h);
xlabel('amplitude');
ylabel('time');
title('circular convolution');
```

unit ramp imulse step

```
clear all
n=input('ENTER THE LENGTH')
t=-2:1:n
for i=1:1:length(t)
  if t(i)>=0
    y(i)=1
  else
    y(i)=0
  end
end

subplot(3,1,1)
stem(t,y)
title("step")

t=-2:1:n
for i=1:1:length(t)
  if t(i)==0
    y(i)=1
  else
    y(i)=0
  end
end

subplot(3,1,2)
stem(t,y)
title("impulse")

t=-2:1:n
for i=1:1:length(t)
  if t(i)>=0
    y(i)=t(i)
  else
    y(i)=0
  end
end

subplot(3,1,3)
stem(t,y)
title("ramp")
```

linear convolution (kit)

```
#include <stdio.h>
#define LENGHT1 6
 #define LENGHT2 4
 int x[2*LENGHT1-1]={1,2,3,4,5,6,0,0,0,0,0};
int h[2*LENGHT1-1]={1,2,3,4,0,0,0,0,0,0,0};
 int y[LENGHT1+LENGHT2-1];
 main()
{

int i=0,j;
 for(i=0;i<(LENGHT1+LENGHT2-1);i++)
{
y[i]=0;
 for(j=0;j<=i;j++)
y[i]+=x[j]*h[i-j];
}
for(i=0;i<(LENGHT1+LENGHT2-1);i++)
printf("%d\n",y[i]);
}
```

parceval

```
 N0=5000
 X0=randi(10,N0,1)+1i.*randi(10,N0,1);
 Y0=randi(10,N0,1)+1i.*randi(10,N0,1);
 Y1=conj(Y0);
 RHS=sum(X0.*Y1)
 C=fft(X0);
 D=fft(Y0);
 F=conj(D);
 RHS1=(1/N0).*sum(C.*F)
```

real time fft without buffer (kit)

```
#include<stdio.h>
#include<math.h>
#include<complex.h>
float mag[8];
int main()
{ int i,N,L,k;
char sign;
double PI=acos(-1);
double complex X[8];
int x[8]={1,2,3,4,5,6,7,8};
N=8;

for(k=0;k<N;k++)
 { X[k]=0;
 for(i=0;i<N;i++)
 X[k]+=x[i]*cexp(-1*2*PI*k*i/N);
}
for(k=0;k<N;k++)
{sign cimag(X[k]) <0? '-':'+';
printf("%f %c%fi", creal (X[k]), sign, fabs(cimag (X[k])))
}
mag[k]=sqrt(pow(creal(X[k]),2)+pow(cimag(X[k]),2));
for(k=0;k<N;k++)
}
{
printf("\n%f", mag [k]);
return 0;
}
```

real time fft with buffer (kit)

```
#include "L138_LCDK_aic3106_init.h"
#define BUFSIZE 512
int32_t inbuffer [BUFSIZE];
int16_t buf_ptr = 0;
interrupt void interrupt4(void)
{ int16_t sample_data;
sample_data = input_left_sample();
inbuffer[buf_ptr] = sample_data;
buf_ptr (buf_ptr+1)%BUFSIZE;
output_left_sample(sample_data);
return;
}
int main(void)
{L138_initialise_intr(FS_8000_HZ, ADC_GAIN_ODB,
DAC_ATTEN_ODB, LCDK_LINE_INPUT); while(1);
}
```

ovelap add method

```
clear all;
clc;
close all;

x=[3 -1 0 1 3 2 0 1 2 1]
h=[1 1 1]
L=3;
M=3
N=L+M-1
x=[x zeros(1,L)]
h=[h zeros(1,2)]
s=size(x)
y=[]
for i=1:L:s(2)-L
   x1=[x(i:i+L-1) zeros(1,M-1)]
   x2=cconv(x1,h,N)
   if i==1
     y=x2
   else
     y=[y(1:end-(M-1)) y(end-M+2:end)+x2(1:M-1) x2(M:end)]
   end
end
subplot(2,1,1)
stem(y)
xlabel("n")
title("Linear Convolution using Overlap add method")

y1=conv(x,h)
subplot(2,1,2)
stem(y1)
xlabel("n")
title("Linear Convolution using conv() function")
```

ovelap save method

```
close all

x=[3 -1 0 1 3 2 0 1 2 1];
h=[1 1 1];
L=3;
M=3;

N=L+M-1;
x=[zeros(1,M-1) x zeros(1,M-1)];
s=size(x);
y=[];
for i=1:L:s(2)-L
x1=x(i:i+N-1);
x2=cconv(x1,h,N)
y=[y x2(M:end)];
end
subplot(211)
stem(y)
xlabel("n")
title("Linear Convolution using Overlap save method")
y1=conv(x,h);
subplot(212)
stem(y1)
xlabel("n")
title("Linear Convolution using conv() function")
```