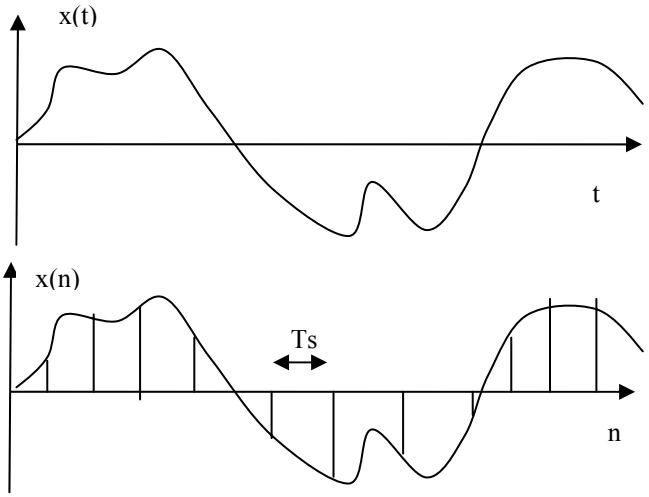
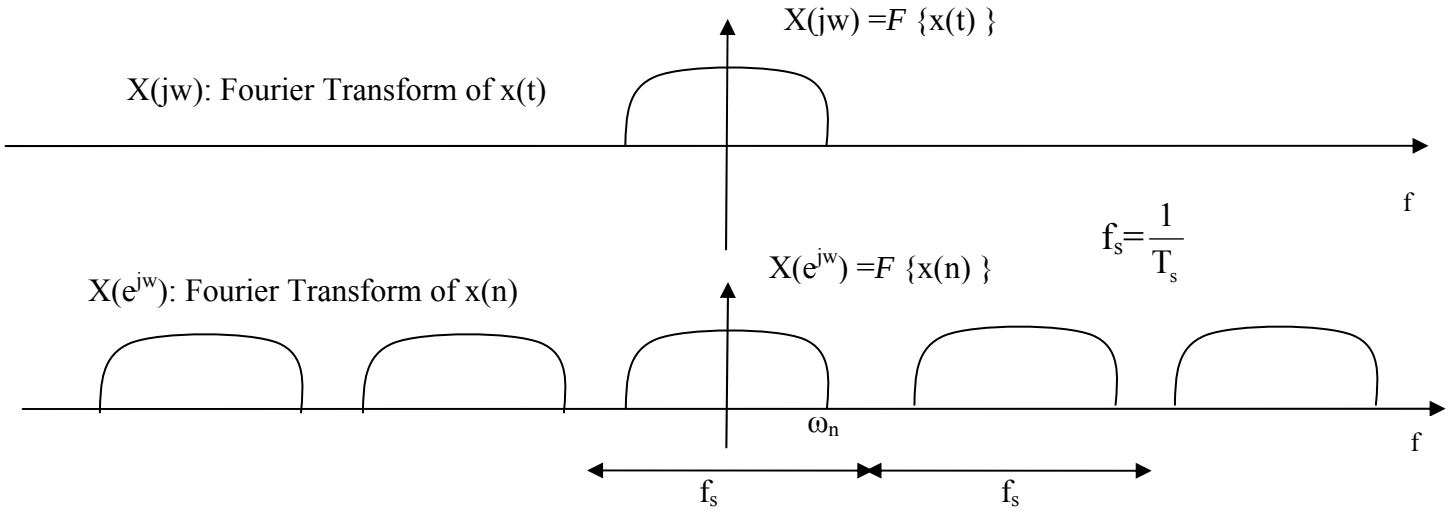
Continuous and Discrete Fourier Transform

When a continuous signal is sampled, we can calculate the discrete Fourier Transform (DFT).



x(t): is the continuous signal

$x(nT_s)=x(n)$ is the sampled version of x(t)



ω_n:The highest frequency in x(t).

$$X(jw)=\int_{-\infty}^{+\infty} x(t)\ e^{-jwt}\ dt \qquad X(e^{jw})=\sum_{k=-\infty}^{\infty} x(n)\ e^{-jwn}$$

The following statements are **wrong**
Continuous Fourier Transform of x(n)
Discrete Fourier Transform of x(t).

X(jw) is difficult, sometimes impossible to calculate. $X(e^{jw})$ is easily calculated.

Since the main lobe of $X(e^{jw})$ is same as X(jw). We examine the main lobe of $X(e^{jw})$to analyze X(jw).

Ts is sampling period. fs=1/Ts is the sampling frequency.
The highest frequency in x(t) is ω_n. f_s should be greater than 2 ω_n

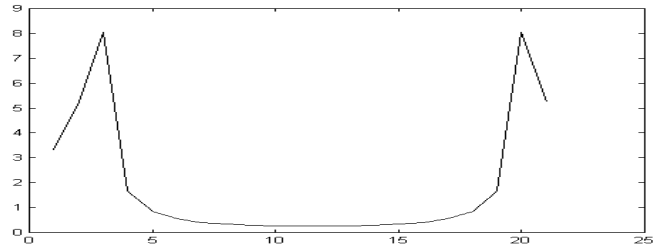
FAST FOURIER TRANSFORM
Fast Fourier Transform (FFT) is discrete Fourier Transform. It is a **fast algorithm** to calculate discrete Fourier Transform (DFT)

MATLAB command fft calculates FFT.

Example. x(t)=sin(t); This is an analog (continuous) signal. We want to see the amplitude and phase spectrum of this signal.

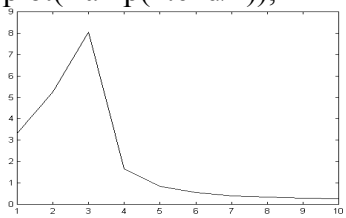
Sampling:
t = [0 0.5 1 1.5 2 ……. 9.5 10]
x(n) = [sin(0) sin(0.5) sin(1) ……. sin(9.5) sin(10)]
x(n)=[0 0.48 0.84 ..... -0.075 -0.54];

MATLAB format is
t=0:0.5:10; x=sin(t);
frx = fft(x); framp=abs(frx); plot(framp);

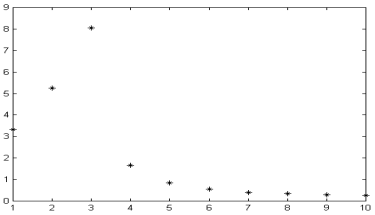


Since its symmetrical we draw only the half.

plot(framp(1:end/2));

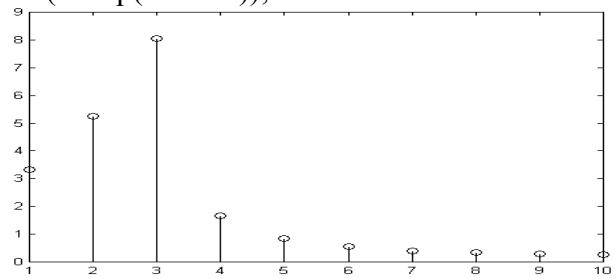


It may be a good idea to see only the values.
plot(framp(1:end/2));



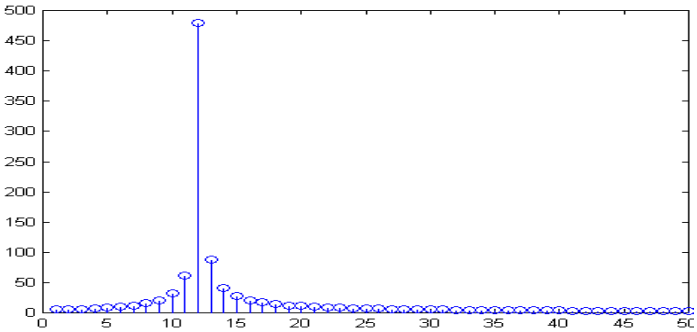
Or more clearly

stem(framp(1:end/2));



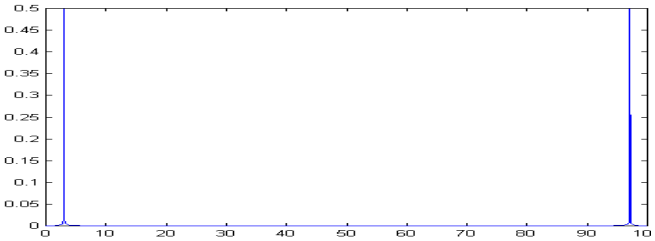
Try the followings

t=0:0.01:10; x=sin(7*t);
frx = fft(x); framp=abs(frx); stem(framp);



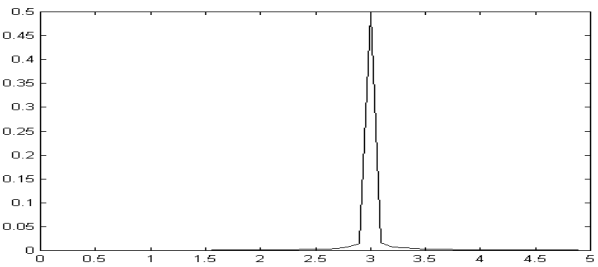
SCALING THE FREQUENCY AXIS

```
t=0:0.01:10;   x=sin(2*pi*3*t);
total_time=10;
 NN=length(t);
 frr=fft(x);    framp=abs(frr)/NN;
 Ts=total_time/NN;     fs=1/Ts;
 freq_axes=fs*[0:NN-1]/NN;
 plot(freq_axes,framp);
```

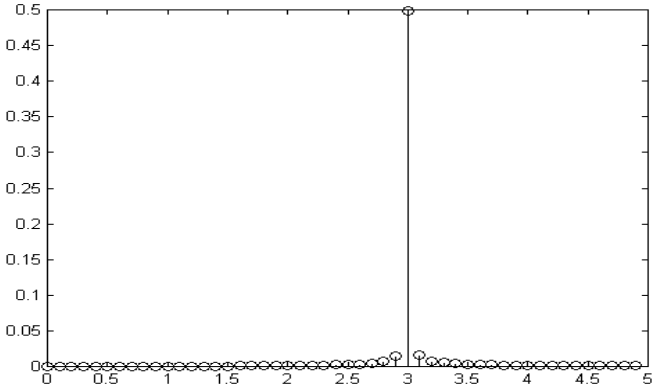


To see clearly

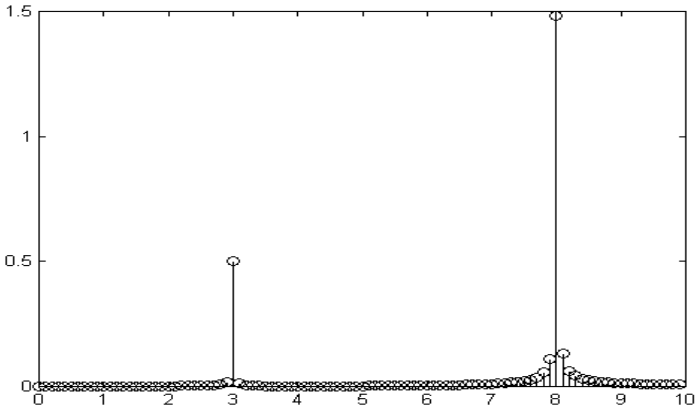
```
 plot(freq_axes(1:50),framp(1:50));
```



```
stem(freq_axes(1:50),framp(1:50));
```

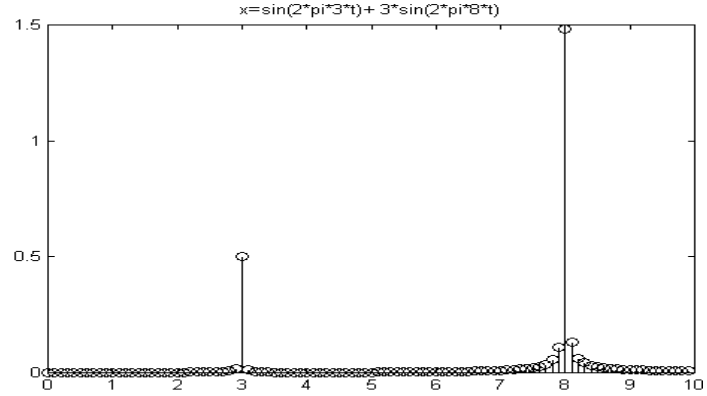


```
x=sin(2*pi*3*t);
   ----------------------------------------
t:0.01:10;
x=sin(2*pi*3*t)+ 3*sin(2*pi*8*t);
  total_time=10;
  NN=length(t);
  frr=fft(x);    framp=abs(frr)/NN;
  Ts=total_time/NN;
  fs=1/Ts;
  freq_axes=fs*[0:NN-1]/NN;
  stem(freq_axes(1:100),framp(1:100),'k');
```



Put a title on the figure

```
title('x=sin(2*pi*3*t)+ 3*sin(2*pi*8*t)');
```



CASE STUDIES

Example 1.

Run the file. See the figure. Listen to the voice.

```
  total_T=1;
  NN=16000;
   t_axis=([0:NN-1]/NN)*total_T;
   xx= sin(2*pi*1000*t_axis);
   Ts=total_T/NN;  fs=1/Ts;
   f_axis=([0:NN-1]/NN)*fs;
   yy=abs(fft(xx));
   subplot(211);   plot(t_axis,xx); grid;
    title('Time response ');
    subplot(212);   plot(f_axis,yy); grid;
    title('Amplitude spectrum ');
  wavplay(xx,16000)
```

```
Note: [0:5]  ==>   0 1 2 3 4 5
      [0:5]/5  ===>   0 0.2 0.4 0.6 0.8 1
     30*[0:5]/5  ==>  0 6  12  18  24 30
     [0:NN-1]/NN)* total_T  from zero to
total time.  step size is total_T/NN .
```

Example 2. Change frequency 400,300,200 and listen to the voice

```
 sin(2*pi*400*t_axis);
```

Example 3. Change frequency 4000,5000,7000 and listen to the voice

```
sin(2*pi*7000*t_axis);
```

Example 4. Add two frequency listen to the voice see the spectrum

```
 xx= sin(2*pi*500*t_axis)+
      sin(2*pi*7000*t_axis);
```

Example 5. Add three or more frequency listen to the voice see the spectrum

```
xx= sin(2*pi*500*t_axis)+
     sin(2*pi*2000*t_axis)+
     sin(2*pi*3000*t_axis) +
     sin(2*pi*7000*t_axis);
```

Example 3

```
x= sin(2*pi*10*t)+ sin(2*pi*20*t);
x= 0.3*sin(2*pi*10*t)+ sin(2*pi*20*t);
x= sin(2*pi*10*t)+ 0.3*sin(2*pi*20*t);

x= sin(2*pi*10*t) + sin(2*pi*20*t)+
    sin(2*pi*30*t) + sin(2*pi*40*t);
```

```
sampling_period = total_time/Number_of_sampling;
sampling_frequency = 1/sampling_period;

freq_axes=sampling_frequency*[1:NN]/NN;
```

Total program is

```
>>  t=0:0.5:10; x=sin(t); NN=length(t);
>>  frr=fft(x);    framp=abs(frr)/NN;
>>  total_time=10;
>>  sampling_time=total_time/NN;
>>  sampling_frequency=1/sampling_time;
>>  freq_axes=sampling_frequency*[0:NN-1]/NN;
>>  plot(freq_axes,framp); grid;
```

Try the following

```
>>  t=0:0.01:10; x=sin(2*t); NN=length(t);
>>  frr=fft(x);    framp=abs(frr)/NN;
>>  total_time=10;
>>  sampling_time=total_time/NN;
>>  sampling_frequency=1/sampling_time;
>>  freq_axes=sampling_frequency*[0:NN-1]/NN;
>>  plot(freq_axes,framp); grid;

>>  plot(freq_axes(1:15),framp(1:15)); grid;
>>  stem(freq_axes(1:15),framp(1:15)); grid;
```

t=0:0.5:10; x=sin(t); Ts=1; fs=1/0.5=2
time axes has 20 elements. Frequency axes has 20 elements.
t varies from zero to 10. (step size 0.5)
f varies from zero to 2. (step size is 0.1)

freq_axes=[0 0.1 0.2 0.3 0.4 ... 1.9 2]

Matlab calculates from w=0 to w=2pi. Since the spectrum is symmetrical
we need only w=0 to w=pi.

MATLAB Session
Draw amplitude and phase spectrum of the transfer function

$$H(s)=\frac{10(s+1)}{s^2+4s+13}$$

```
>> w=0; s=j*w; hh=10* (s+1)/(s^2+4*s+13); amp=abs(hh),
pp=phase(hh)

 amp =   0.769
  pp =  0

>> w=0.1;  s=j*w; hh=10* (s+1)/(s^2+4*s+13); amp=abs(hh),
pp=phase(hh)
  amp =   0.773
  pp =  0.689

>> w=0.2; s=j*w; hh=10* (s+1)/(s^2+4*s+13); amp=abs(hh),
pp=phase(hh)
    amp =   0.785
    pp =  0.1357

>> w=0.3; s=j*w; hh=10* (s+1)/(s^2+4*s+13); amp=abs(hh),
pp=phase(hh)
    amp =   0.805
    pp =  0.1988

……..
>> w=1; s=j*w; hh=10* (s+1)/(s^2+4*s+13); amp=abs(hh),
pp=phase(hh)
    amp =   1.118
    pp =  0.4636
```

Make an array

```
 >> ww  = [      0         0.1        0.2        0.3       1       ]
>>total_amp=[0.769    0.773    0.785    0.805    1.118   ]
>>total_ph=[   0        0.689    0.1357    0.198    0.463   ]

>>plot(ww,total_amp),   figure,  plot(ww,total_ph)
```

SHORT METHOD

```
>>ww=0:0.1:1; s=j*w;   num=(s+1), den=s.^2 +.4 .* s +13,
hh=num./den, amp=abs(hh), >> ph=phase(hh)
>>plot(ww,amp),   figure,  plot(ww,ph)
```