Lecture 6

DSP Crash Course

EE 521: Instrumentation and Measurements

Lecture Notes Update on October 26, 2009

Aly El-Osery, Electrical Engineering Dept., New Mexico Tech

6.1

Contents

6.2

1 Bandlimited and Timelimited Signals

Rect Window

See Figure 1

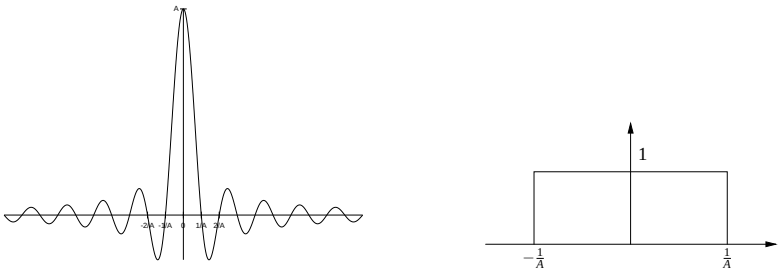

Figure 1: Fourier transform pair - rectangular window and sinc function

- Fourier transform of a rectangular window is a sinc.
- Inverse Fourier transform of a rectangular window is also a sinc.
- We can only have either timelimited or bandlimited but not both.

6.3

2 Fourier Transform Overview

Transform Equations

DTFT

$$x(n) = \frac{1}{2\pi}\int_{2\pi} X(\omega)e^{j\omega n}d\omega \tag{1}$$

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \tag{2}$$

6.4

Transform Equations

DFT

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \qquad k = 0,1,2,\ldots,N-1 \tag{3}$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N}, \qquad n = 0,1,2,\ldots,N-1 \tag{4}$$

6.5

CTFT

See Figure 2

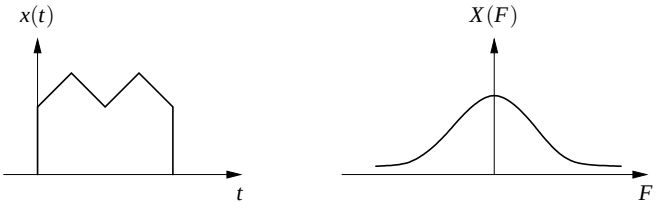

Figure 2: Continuous Fourier transform of continuous signal

6.6

DTFT

See Figure 3

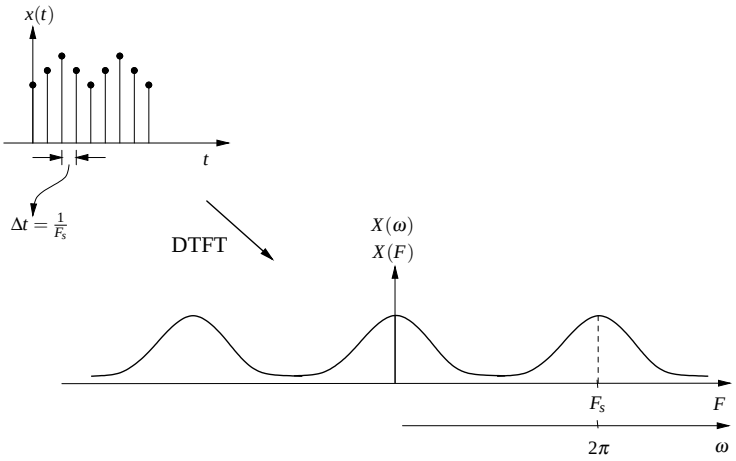

Figure 3: Discrete time Fourier transform

6.7

DFT

See Figure 4

6.8

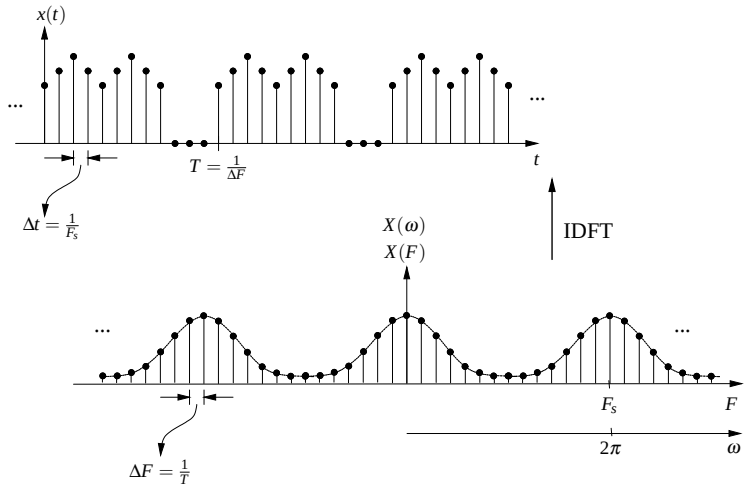

Figure 4: Discrete Fourier transform

3 A Closer Look on DFT

Limitations

1. The number of data points must be finite.
2. The computation time required increase as the number of data points increase.
3. Frequency resolution is important in determining the signal content.
4. Limiting the number of points of a continuous time signal results in *spectral leakage*.

6.9

Number of Points

Assume the number of point in the time domain is N_t and the number of points in the frequency domain is N_F.

$$N_t = \frac{T}{\Delta t} = \frac{1/\Delta F}{1/F_s} = \frac{F_s}{\Delta F} = N_F = N \tag{5}$$

6.10

Zero Padding

We can increase the time series sequence by adding zeros and that would not affect it. By doing so the number of points in the time domain, N_t is increased, and consequently, also is the number of points in the frequency domain, N_F. Referring to Eq. 5, this means that ΔF is decreased.

Zero padding shows more details but not more information

6.11

Window Size

To demonstrate the effect of the different window sized we look at a signal with three different frequencies and then compute the FFT for different length of the same signal. See Figure 5

6.12

Window Size

See Figure 6

6.13

Window Size

See Figure 7

6.14

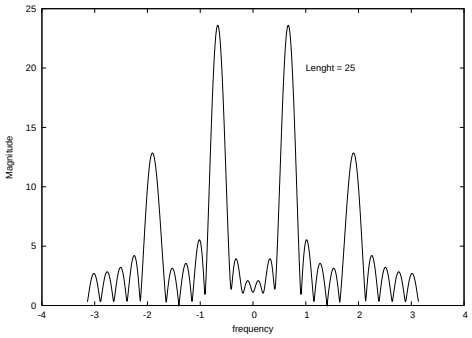

Figure 5: $N = 4096$

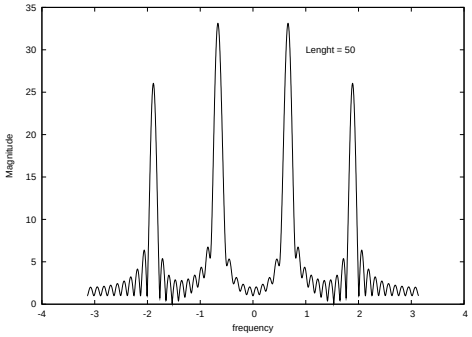

Figure 6: $N = 4096$

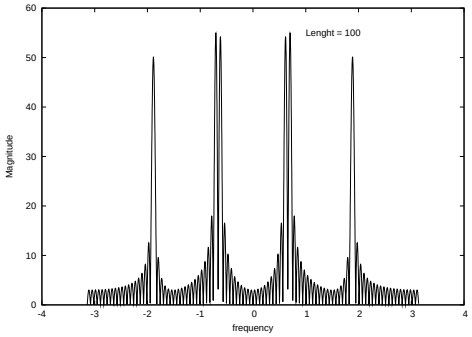

Figure 7: $N = 4096$

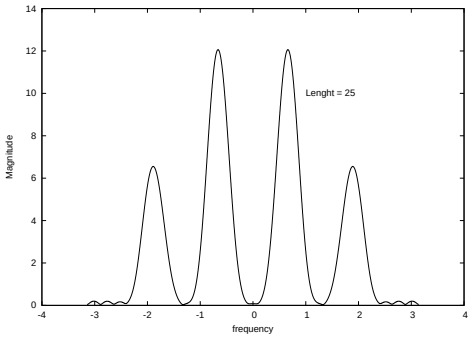

Figure 8: $N = 4096$

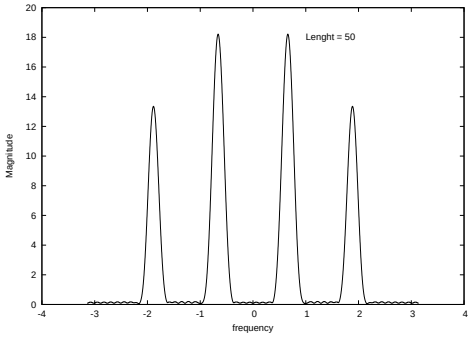

Figure 9: $N = 4096$

Min. Resolvable Resolution

The DTFT of a rectangular window of length L is given by

$$W(\omega) = \frac{\sin(\omega L/2)}{\sin(\omega/2)} e^{-j\omega(L-1)/2} \tag{6}$$

To avoid main lobes of overlapping

$$|\omega_1 - \omega_2| > 2\pi/L \tag{7}$$

6.15

Different Windows

Spectral leakage is due to the sharp cut-off rectangular window. To reduce this effect different windows with smoother roll-off are used. This is at the cost of wider main lobe which may be undesirable.

6.16

Hamming Window

Using the same example as earlier of a signal with three frequencies we apply the Hamming window before computing the FFT and then investigate different window length. See Figure 8

6.17

Hamming Window

See Figure 9

6.18

Hamming Window

See Figure 10

6.19

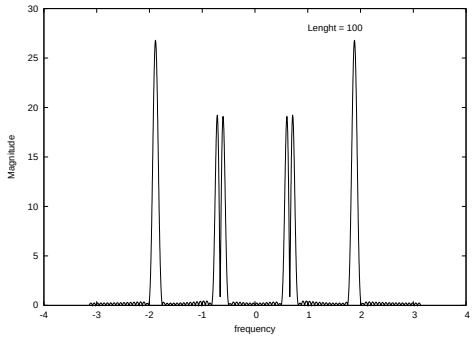

Figure 10: $N = 4096$

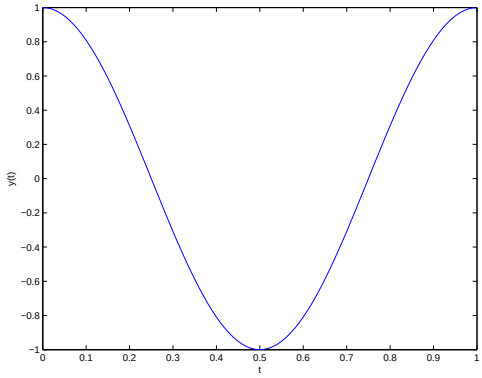

Figure 11: 1sec sinwave of 1Hz

4 FFT

- An efficient way to compute DFT.
- Direct computation of DFT requires approx. N^2 complex multiplications and N^2 complex additions.
- FFT algorithm requires approximately $N/2\log_2 N$.
- For a 1024 point signal direct computation requires 1,048,576 complex computation versus 5,120 of the FFT.

6.20

4.1 FFT in MATLAB

Example

Best way to explain that is using an example. Assume that we have a sinusoidal signal that we want to determine its Fourier transform. See Figure 11

```
>> fs=100;
>> t=0:1/fs:1;
>> y=cos(2*pi*t);
>> plot(t,y)
>> xlabel('t')
>> ylabel('y(t)')
```

6.21

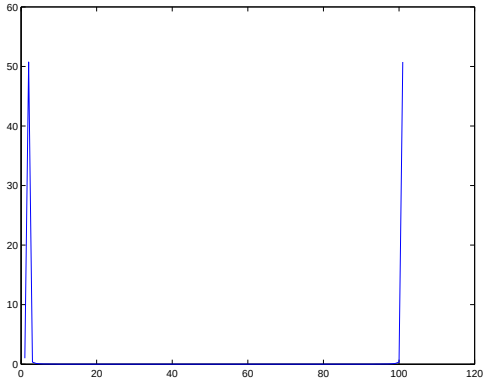

Figure 12: 101-point FFT

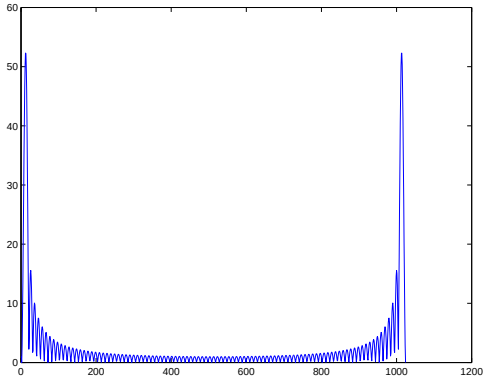

Figure 13: 1024-point FFT

Straight FFT

Use the following commands to compute the FFT, find its length and plot the magnitude of Y. See Figure 12

```
>> Y=fft(y);
>> length(Y)
ans =
   101
>> plot(abs(Y))
```

6.22

More Details

We can specify the length of the FFT to be longer by See Figure 13

```
>> N=1024;
>> Y=fft(y,N);
>> plot(abs(Y))
```

Now we can see more details, but still we can't extract a lot of information from the plot. If we compute the Fourier transform of a sine or a cosine function we expect to see two impulses at the frequency of the sine or the cosine. But, that's not what we are seeing. Here is how we can modify the plot to see something more familiar.

6.23

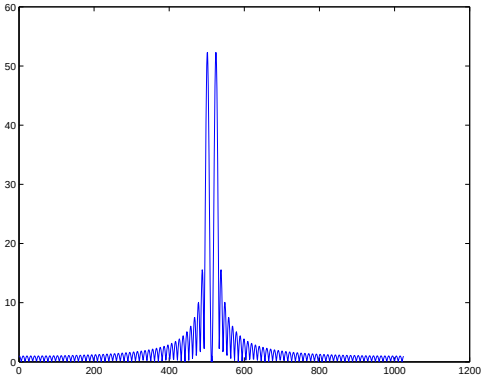

Figure 14: `fftshift`

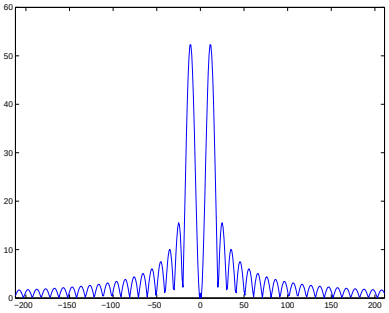

Figure 15: `fftshift` with $k = -N/2 : N/2 - 1$

fftshift

```
>> plot(fftshift(abs(Y)))
```

See Figure 14 6.24

Axes Mapping

But, it is not really the impulses that we expected. We have all those ripples and also the x-axis does not match the frequency of the cos. Let's deal with the second problem first.

We need to map the x-axis value to reflect frequency in Hz. To do that all you need to remember that the length of the FFT corresponds to the sampling rate F_s for continuous frequencies and corresponds to 2π for discrete frequency. Therefore to get the positive and negative frequencies you need to get the following.

```
>> k=-N/2:N/2-1;
>> plot(k,fftshift(abs(Y)))
```

See Figure 15 6.25

Axes Mapping

Now we get the axis containing positive and negative values. All we have left to do is to map it to actual frequencies.

```
>> plot(k*fs/N,fftshift(abs(Y)))
```

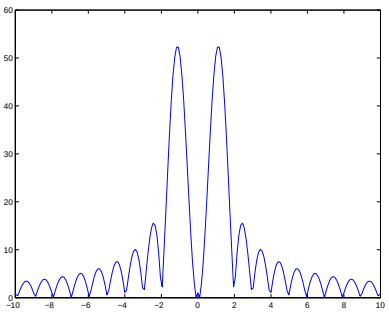

Figure 16: Frequency mapping

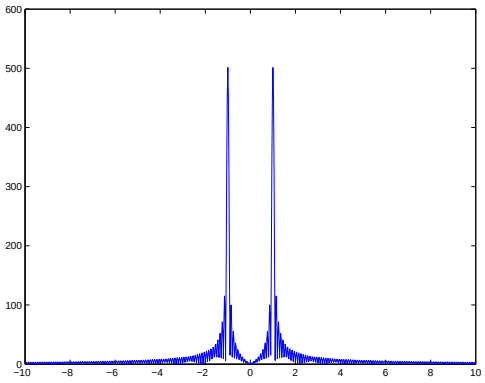

Figure 17: FFT of a 10sec 1Hz sinwave

See Figure 16

Now are getting something meaningful. We have the two peaks at $\pm$1Hz. But, they are not really impulses like we wanted. The reason is we really wanted an infinitely long sinusoid to get the impulses, due to the fact that we can't have an infinitely long sinusoid to process, we have effectively multiplied the sinusoid by a rectangular window of length 1sec. If we multiply in the time domain by a rectangular window, we convolve in the frequency domain with a `sinc` function. That is what we are getting, two `sinc` functions centered around $\pm$1Hz. 6.26

Window Length

If we increase the length of the window, we expect the `sinc` function to look more and more like an impulse. To do that lets generate a sinusoidal signal that last for 10sec instead.

See Figure 17

```
>> fs=100;
>> t=0:1/fs:10;
>> y=cos(2*pi*t);
>> N=4096;
>> Y=fft(y,N);
>> k=-N/2:N/2-1;
>> plot(k*fs/N,...
   fftshift(abs(Y)))
```

6.27