**Digital Communication Using GNU Radio**

**Prof Kumar Appaiah**

**Department of Electrical Engineering**

**Indian Institute of Technology Bombay**

**Week-01**

**Lecture-05**

Another practical effect is the impact of windowing. As you are aware a cosine like the one which we have shown here is a time unlimited signal whose Fourier transform is actually

$$\frac{1}{2}\delta(f-f_0)+\frac{1}{2}\delta(f+f_0) \quad .$$

In this particular situation we have a 1 Hz cosine and its Fourier transform is shown as having two impulses. Naturally, since

$$s(t)$$

is a time unlimited signal we cannot represent it or understand it in a practical scenario we have to perform a windowing or a time limitation. Let us now perform a time limitation by restricting the cosine to being between -2 and 2. Immediately you will see that in the frequency domain the Fourier transform changes significantly.

It has peaks at -1 and 1 indicating the flavor of the cosine is preserved but because you have multiplied it by a rectangle which is shown in the dotted lines over here a rectangle has a Fourier transform of a sinc. Therefore in the frequency domain the convolution of these impulses with sincs implies that you have sincs being placed at those locations. Therefore you can clearly see that the Fourier transform of this signal which is a truncated cosine has the flavor or characteristic of the original cosine but is significantly altered. To get back some of the characteristic of the cosine Fourier transform we can make the window slightly larger.

If you go from -3 to 3 then the sincs become slightly smaller and you have a narrower and narrower kind of sinc in the frequency domain. If you go from -4 to 4 also you will see that it becomes even narrower. Therefore the more you are windowing the more accurate or closer you get to that infinite cosine but you can never get to the accurate impulse because of the fact that you have to truncate in order to perform any practical

simulation based on these signals. This is something which we can observe practically in GNU radio as well. Let us now take a quick detour to see how spectral lines appear on GNU radio and how different FFT window sizes affects the spectrum.

To do this we start with a simple GNU radio example where we first get a signal source. As usual we press Ctrl F or Cmd F and type signal and you can grab the signal source, place it onto your float graph. We double click it and change the complex to float because we want a real signal. We also want to be able to vary the frequency. Therefore we will change the 1000 to the keyword freq and we will then add a QT-GUI range that allows us to alter the frequencies.

We will now click OK. We will then add a QT-GUI range first. So Ctrl F or Cmd F type range. Grab the range. Place it onto our screen.

The ID will be freq. Float is the type. We want the default value to be a 1000 Hz. We will start exactly at let's say 100 Hz and we can stop at let's say around 16000 Hz that corresponds precisely to half the sampling rate. In fact we would want to stop a little before but it's fine.

We will make the step size 0.1 Hz. We can now add a throttle. Cmd F or Ctrl F type throttle. Grab the throttle.

Double click it. Change the type to float. Say OK. We then connect the signal source to the throttle and the final component that we will need is the frequency sink. So we press Ctrl F or Cmd F and type freq.

Grab the QT-GUI frequency sink. Again change this type to float. We will also change the window type to rectangular so that we don't have any extra effects in the spectrum. We will also add a grid and set auto scale to yes. We will then say OK.

Connect the throttle output to the frequency sink and we then run our flow graph. You will see this kind of a spectrum. You will see that there are distinct lines at around minus 1 kHz and 1 kHz as expected for a cosine. But because of numerical issues you will see some other lines as well specifically at harmonics but these can safely be ignored because they are much much below 150 dB which means they are roughly 10 power minus 15 times the peak value or close to that. Now, if we now change the frequency somewhat, let us say we will change this to about 1010 Hz.

You will see that the spectral line is no longer as narrow and you have a slightly wider spectrum. But if you choose another number like let's say 1600 Hz, you will see something which is still very wide and let's say if you choose 2000 Hz, you will see

something like this. You will again start seeing distinct lines. Why is this the case? To understand why we observe these kinds of spectral characteristics, we must understand how GNU radio computes the spectrum that is being displayed. GNU radio uses the discrete Fourier transform evaluated using the fast Fourier transform algorithm for obtaining the frequency spectrum.

You would recall from your signals and DSP course that whenever you evaluate the DFT of a sampled continuous time signal, if the continuous time signal has a frequency that satisfies

$$\frac{f}{F_s} \times N_{FFT} \quad ,$$

where

$$F_s$$

is the sampling frequency, if

$$\frac{f}{F_s} \times N_{FFT}$$

is an integer, then the DFT produces distinct lines. That is, the DFT will be non-zero only at those one or two points that correspond to the frequency

$$f \quad \text{and} \quad -f$$

respectively. For other frequencies, this will not hold. Let us take an example. If we choose

$$N_{FFT}$$

to be 1024, a sampling frequency of 32000 Hz, then choosing

$$f$$

to be 1000 Hz gives us 1000 upon 32000 times 1024 is 32, which is an integer, therefore we get a distinct line.

The next number of interest will be 1031.25. You should try to figure out why that is the case. But, it is very evident that 1000, 2000 and such numbers will have this property, but numbers like 1010 or 1016 will not have this property when you choose nfft to be 1024 and fs to be 32000 Hz. Therefore, in such situations, you will get a DFT that has non-zero values in the neighboring bins as well.

To get a better idea, let us introduce the control panel by middle clicking and choosing control panel in our frequency plot. We can see that we have 1024 rectangular window. Let us now change this to 2010 Hz. You will see now that the spectrum becomes slightly fat. Again, that is because 2010 divided by 32000, that is the sampling frequency, times

$N_{FFT}$

is not an integer.

Let us stick to this 2010 and see how the windowing affects our spectrum. We will start at a 32 point DFT. In the case of a 32 point DFT, you can see that the cosine which should yield two impulses actually yields really fat curves. This is because the FFT size is small. We take a 32 length window and a 32 length window is insufficient for us to accurately produce the DFT.

If we make it 64, we can see that it heads closer to the desired impulses. At 128, even narrower, let us go to 1024. You can see that it becomes even more narrow. Now, the complexity becomes higher.

But let us choose 16384. You can see that you end up getting discrete or nearly discrete lines. If we go back to our old frequency which is 1000, you will still see very sharp lines. But 1010 will not have the same sharp lines but they are sharper than before. The reason is because choosing 16384 samples yields a much much larger window and thereby produces a more close estimate of the Fourier transform of the infinite length cosine. To summarize the past few things that we have seen, vector spaces are as applicable for vectors as they are for signals.

In fact, as applicable for signals as they are for vectors. We have also seen signals and their spectral picture using the Fourier transform. And how the Fourier transform can be used to interpret the frequency characteristics of signals. We looked at some common operations of signals such as convolution, multiplication, their norm, energy and things of that sort. And finally, we have seen both theoretically and practically impacts of windowing, the use of the discrete Fourier transform for representing signals and other such features. Thank you.