Digital Communication Using GNU Radio

Prof Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

Week-01

Lecture-04

Welcome to this lecture on Digital Communication Using GNU Radio. In this lecture, we are going to have a brief recap of signals along with some new concepts. Many of these concepts would have been covered in your Signals and Systems course, but we will take another look at them with a perspective of digital communication in mind. In this lecture, we will recap signals and some common operations on signals that are relevant for the purposes of this course. We will then introduce the vector space picture for signals. We will then discover the frequency view in the form of the Fourier transform and see how that comes in handy for both analysis and design of signals for digital communication.

And we will have several examples to drive home the concepts that we are talking about in this lecture. You may recall that vectors are ordered collections of real or complex numbers. As an example, if you look at the vector consisting of the elements

$1+1j$

and

$-1-0.2j$,

this can be considered a two-dimensional complex vector.

Similarly, the

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 3 \end{bmatrix}$$

can be considered as a five-dimensional real or complex vector. Most operations on vectors that you are familiar with remain the same for this course as well. They will just extend to newer situations such as signals as we will see. So it is useful to have the

geometric picture wherein vectors are considered points in n-dimensional coordinate space either real or complex and it is convenient to consider them in terms of vector spaces because several operations on vectors such as dot products or taking norm are very, very useful in the context of digital communication as well. If you recall what signals are, signals are generally real or complex valued functions of a single variable, generally time or frequency.

So as an example, if you look at the signal, it is a somewhat oscillating signal. It has a single value for every value of time. Generally we will define a signal for all values of

t

or within an interval thereby implicitly assuming that it is 0 outside that said interval. There are also some special signals such as

$\delta(t)$

or

$u(t)$

The reason I call them special is because these do not fall in the regular class of functions.

For example,

$\delta(t)$

is an impulse. Therefore its value at 0 is not really defined. It is defined in terms of an integral whose area is 1. Similarly

$u(t)$

is the unit step function. It also has the issue that at 0 its definition is not very evident.

So if you look at this step function, at 0 the definition is not very clear. However we will carry forward the same meanings of these functions or signals as you have seen in the context of signals and systems. When it comes to dot products and inner products, we are going to extend the definition of these for the context of signals also. Dot products and inner products carry forward the same meaning as they have in the context of regular vectors in the maths context. Dot products and inner products have the same meaning that they have in the context of vectors in the mathematical sense, but here we will extend the concept to the context of signals as well.

Dot products and inner products are naturally defined by the vector space. It can be considered as an inner product of two complex or real vectors

and v_2

The notation that we use is

and its definition is

$v_1^H v_2$

That is you take every element of

v_1

take its conjugate, multiply it by

v_2

and add the result. That is what you have over here.

So component wise conjugate of

v_1

multiplied by the component of

v_2

and addition. The key thing is that the same concept can be extended to the case of signals because if you assume that a signal is just a long vector indexed by

t

then we can define the inner product of signals also in the form of an overlap integral which is just an extension of the inner product. For example, if you have two complex signals

$s$and r

then this angle bracket notation

$\langle s, r \rangle$

translates to the

When it comes to energy and norm, once again the same definitions as the maths sense continue for vectors, but we are able to extend these two signals. Norm and energy are notions of squared distance and for those from an electrical engineering background the squared distance also corresponds to something like power and energy.

For a vector

$$v=\begin{bmatrix} v_1 & v_2 & \dots & v_N \end{bmatrix}^T$$

the two norm is defined as

$$\|v\|=\sqrt{\sum_{k=1}^{N}|v_k|^2}$$

That is you just take the component wise magnitude squared and add them and take the square root this results in the two norm. The square of the two norm can be viewed as something like the energy in the vector. For signals, once again using a similar approach that we used earlier for the inner products, we can just define the energy of a signal as

$$\|s\|^2=\langle s,s\rangle=\int_{-\infty}^{\infty}|s(t)|^2\,dt$$

So this definition extends from the concept of vectors to signals naturally.

Let us now take a look at an example. In this example, we have two functions or signals

and $\phi_2(t)$

over here.

is 1 between 0 and 1,

$\phi_2(t)$

is 1 between 1 and 2 and both of these functions are 0 for all other values of

t

We are now going to treat

and $\phi_2(t)$

as so called basis functions more like ingredients using which you can construct new signals. One thing which comes in handy is the fact that

and $\phi_2(t)$

have some nice properties.

The first is that if you find the integral

the area under phi 1 is actually 1. Similarly, if you now find

$$\int_{-\infty}^{\infty} |\phi_2(t)|^2 dt = 1$$

the area will be 1. Another interesting observation is that

and $\phi_2(t)$

are orthogonal in the sense that if you find the

$\langle \phi_1(t), \phi_2(t) \rangle$

that is if you multiply

and $\phi_2(t)$

and integrate, you will get 0. This means that

and $\phi_2(t)$

can be considered like basis signals using which you can construct other signals conveniently. Let us take an example.

If you now look at

$s_1(t)$

over here, it is

That is you take

scale it by

-0.5

that results in a signal which is

$-\frac{1}{2}$

from 0 to 1. Similarly, you take

$\phi_2(t)$

scale it by

$\frac{3}{4}$

and that results in a signal which is

$\frac{3}{4}$

between 1 and 2. The signal

$s_1(t)$

is actually very different from

and $\phi_2(t)$

but you can express it as a linear combination of

and $\phi_2(t)$

Therefore, in the vector picture, if you now treat

and $\phi_2(t)$

as something like basis vectors or base vectors, you can represent the signal

$s_1(t)$

as a

$\begin{bmatrix} -0.5 & 0.75 \end{bmatrix}^T$

This gives you a way to express a variety of signals as just the combination of

and $\phi_2(t)$

where

and $\phi_2(t)$

's components form a vector. We will learn more about this in subsequent lectures and their utility in the context of modulation. But you must also remember that not all signals between 1 and 2 or between 0 and 2 can be expressed as a linear combination of

and $\phi_2(t)$

Let us look at

$s_2(t)$.

$s_2(t)$

has a structure where it rises from minus half to three fourths between 0 and 1 and then holds the value three fourths between 1 and 2. Between 1 and 2, it is evident that the signal looks a lot like

$\phi_2(t)$

and in fact, it is equal to

$0.75\phi_2(t)$

However, there is no way we can use

and $\phi_2(t)$

to express this ramp like structure and therefore, we can say that

$s_2(t)$

does not belong to the set of functions or signals that can be obtained as a linear combination of

and $\phi_2(t)$

also known as the span.

Thus,

$s_1(t)$

is a linear combination of

and $\phi_2(t)$

and belongs to the span of

and $\phi_2(t)$

but not

$s_2(t)$

A common operation between pairs of signals is convolution and convolution is very standard. It corresponds to the act of filtering. This is something which you have covered already in signals and systems. If you have a signal

$s_1(t)$

another signal

$s_2(t)$

the convolution is defined by the flip and overlap and add shift.

That is you take

$s_1(t)$

you flip it to get

$s_1(-t)$

and then you shift it by

τ

to get

$s_1(t-\tau)$

and you just have to perform the overlap integral at various overlaps. This is something which you have already covered and therefore, we will not be spending too much time on this. Another aspect that you are aware of is the Fourier transform. The Fourier transform can be viewed as a way by which you can express your signal as a

combination of complex exponentials as opposed to just their temporal values. That is if you find

$X(f)$

using the analysis equation, it is

$x(t)e^{-j2\pi ft}dt$

One way to interpret this is that if you take the signals

$e^{-j2\pi ft}$

for various values of

f

you are finding the weightage of

$x(t)$

along these signals for each

f

In other words, this is a way by which you can view

$x(t)$

as being expressed as a combination of complex exponentials. This becomes clear in the synthesis equation where

$x(t)$

is being expressed as a combination of complex exponentials where the weightage of the complex exponential

$e^{2\pi ft}$

is

$X(f)$

Therefore,

$X(f)$

is the weightage of the frequency

f

within

$x(t)$

Some properties of the Fourier transform that you will generally be familiar with are that the patterns for real imaginary even odd signals is something which is common.

For example, a real signal

$x(t)$

has a conjugate even Fourier transform. We will observe this in a minute. Another important aspect that will come in frequently in the context of digital communication is that time limited signals are band unlimited. Band limited signals are time unlimited. In other words, if you compress a signal in the time domain, the frequency domain it goes to infinity.

Similarly, if you have a signal which is limited in the frequency domain to a range of frequencies in the time domain, it will go from minus infinity to infinity. The convolution is very special because when you convolve two signals in the time domain, their Fourier transforms get multiplied. This is a very commonly observed property that is very useful when you design filters and this will carry over in the context of digital communication also. Finally, Parseval's theorem states that the energy when measured in the time domain which is

$$\int_{-\infty}^{\infty} |x(t)|^2 dt$$

magnitude and the energy measured in the frequency domain by integrating the mod square of the Fourier transform are the same. Another way to interpret this in the context of vectors is that if you change the basis or the axis in which you represent a vector, that doesn't change its length.

The same thing carries over in terms of interpretation for signals and that is how the Parseval's theorem can be interpreted. As an example of the first property that we saw in the previous slide, let us take a signal which is real

$x(t)$

If

$x(t)$

is real, then we have a conjugate even

$X(f)$

What does that mean? That means

$X(f)=X(-f)$

Let us see if that is the case.

$x(t)$

being real implies that

$x(t)=x^*(t)$

implies that if we now write the definition of the Fourier transform of

$x(t)$

we get

$$X(f)=\int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\,dt$$

Let us now take the conjugate of this equation. That is going to be integral, this particular integral and conjugate.

Of course,

t

is real. The other two we will place conjugates on. So, we will have

$x^*(t)\mathrm{e}^{-j2\pi ft}$

the star of that. Since

$x(t)$

is real, we just write it as

$x(t)$

and

$\left(e^{-j2\pi ft}\right)^* = e^{j2\pi ft}$.

Now evidently this integral is the same as the integral above except that we have replaced

f

with

$-f$

and therefore, we get

$X(-f)$

over here. Therefore, remember that whenever you have a real signal

$x(t)$,

its Fourier transform satisfies the so called conjugate event property that is

$X^*(f) = X(-f)$.

Some common Fourier transform pairs that we will keep seeing again and again. A rectangular signal has a Fourier transform which is a sinc. A Gaussian signal is interesting because its Fourier transform will be Gaussian and in fact if you choose to express your Gaussian as

$e^{-\pi t^2}$,

then its Fourier transform based on our definition can be easily shown to be

$e^{-\pi f^2}$.

The impulse

$\delta(t)$

has a Fourier transform 1. This indicates the fact that an impulse actually contains all frequencies in equal amounts.

Similarly,

$e^{j2\pi f_0 t}$

which is actually a complex sinusoid with frequency

f_0

has the same or similar Fourier transform structure. It is the dual of the previous result that we had. It is an impulse at

f_0 .

This is also very natural. A single frequency

f_0

is just a complex exponential

$e^{j2\pi f_0 t}$.

The key ideas where we will be seeing Fourier transform are for designing and analyzing signals that occupy specific bandwidths and in practice whenever we have multiplexing and multiple access signals, that is you want to put signals for different people in different frequency ranges, use of the Fourier transform along with these signals comes in very handy. If you now take an example of a practical use of the Fourier transform, let us take the case of a sinc and band limits. If you look at sinc and rect as we saw in the previous slide, they form a Fourier transform pair. That is,

$\text{sinc}(t)$

has a Fourier transform which is

$\text{rect}(t)$

or

$\text{rect}(f)$

rather.

$\text{rect}(f)$

is defined as being 1 between minus half and half.

Therefore, because in the frequency domain it is limited in frequency, it is said to be band limited. In other words, a sinc only has a restricted set of frequencies and naturally in the time domain you are aware that a sinc is non-zero for all, goes to non-zero values for many many values of

t

and it does not die out till infinity. If we now do something interesting and if we now consider a combination of sincs, that is, if we take the signal

$\text{sinc}(t) - 0.5\,\text{sinc}(t-1)$,

interestingly this is also band limited because the sinc is band limited,

$\text{sinc}(t-1)$

is also band limited and the linear combination is also band limited within the same interval. That is, if you look over here, this particular signal has the value 1 at 0 and minus half at the value 1. This is because of the property of the sinc wherein it goes to 0 for all integer values of

t

except for

$t = 0$.

And how does this translate in the frequency domain? The frequency domain, you can see that it is still band limited except that because of the fact that you have a combination of sincs, there is some extra information encoded in the Fourier transform. This is a common concept that we will be using again and again in order to encode information in digital communication over a band limited channel. We will now look at a small detour where we will evaluate this particular example on GNU radio. We will now take a brief detour to perform a simulation of the sinc waveforms on GNU radio. To perform these simulations, we will be introducing three new GNU radio blocks.

The first is the vector source that outputs a sequence of numbers at the sample rate 1 after the other. It is useful because you can construct arbitrary waveforms just by entering their sample values. The next block is variables that allow you to reuse values across the flow graph and changing just this variable block will reflect the change across the whole flow graph. Finally, we will be introducing the python import block that allows you to use python modules and code in your flow graph. Remember that a significant part of GNU radio is implemented in python and therefore the full power of python is available to you.

For this course, you do not need knowledge of python. Wherever needed, we will clarify what minimal knowledge you need. In this simulation, we will be using basic numpy for constructing numerical arrays and accessing some common mathematical

functions. Let us now start building our example in GNU radio. For convenience, you will be able to see the key presses at the bottom left like this.

Let us first begin with the vector source. To obtain a vector source, we take our conventional approach by pressing command F or control F and typing vector and you will see vector source under miscellaneous. Let us drag and place it within our flow graph. The vector source by default has the values 0 0 0. We would like to use it to output a desired sequence of numbers repeatedly which is why the repeat we will keep as yes. We can double click on the vector source, change the output type to float and we will change the vector which is by default 0,0,0 within

(1,2,3,4).

The parentheses and the commas are important because that is what GNU radio uses to identify the separate set of values in the form of an array. We then say ok. Since ours is a simulation, our next task will be to use a throttle which we will again say control F or command F and type throttle. We take the throttle, place it in our flow graph, double click on it and convert it to float and finally, we will take a time sink. Once again control F or command F, we type time and we grab the qt gi time sink, place it in our flow graph, we double click it, we change the complex to float.

We then also add a grid by default. We also set the auto scale to yes. We then say ok and we connect the vector source to the throttle and the throttle to the qt gi time sink. You can then save your work and then execute the flow graph. Executing the flow graph yields a waveform like this but to understand this better, we will switch to the stem plot. So I will middle click and choose stem plot and zoom in to a particular region.

You can clearly see that you have the sequence 1, 2, 3 and 4 followed by 1 again. Therefore, the vector source keeps up yielding the sequence 1, 2, 3, 4 again and again repeatedly. If you zoom in closer and find the time gap between any two sequence elements, this stem and this stem, for example this is at 13 milliseconds and this is at about 13.

031 milliseconds. This gap of 0.031 milliseconds is almost exactly equal to the reciprocal of 32000 because our sampling rate is 32000 samples per second and therefore the gap between any two of these stems is going to be 1 upon 32000. We will now close this. The next block that we will see is the variable block. To get a variable block, we can press Ctrl F or Command F and type variable.

You will see variable under core variables. We drag and place it in our flow graph. By double clicking this variable, we can change the variable id to something like t

underscore vales because we will eventually be using this variable to indicate the time values at which we will generate the sink. Let us set the values to be within parentheses 3, 4, 6, 11.

Some arbitrary set of 4 values. We can press OK. We will see at the bottom that a new variable with id t vales has now been created and its value is 3, 4, 6, 11. It is now very easy for us to make our vector source use the t vales variable. How? We can double click the vector source and we just replace this vector with t vales. We press OK. We can then execute our flow graph and in the stem plot, we will be able to see that the values are 3, 4, 6, and 11 and they keep repeating.

This indicates that the variable

t_vals

is being used by the vector source in order to output the samples. Let us now close this window and now move on to our task of creating the sink. To put together our example of a combination of sincs, we want to be able to produce the waveform a1 times

$\text{sinc}(t)$

plus a2 times

$\text{sinc}(t-1)$,

where t is measured in milliseconds. This corresponds to a sinc whose value at t equal to 0 will be a1 and t equal to 1 will be a2. Remember that the sampling rate that we are using is 32000 samples per second.

To construct one of these sinks, we will use 1024 values of t. Remember that we cannot produce continuous time waveforms. So, we must approximate them using finite discrete sets of values. So, we will restrict ourselves to 1024 values of t. These 1024 values we will choose as minus 512 upon 32, minus 511 upon 32 and so on in our GNU radio example going through 511 upon 32. What justifies this choice? If you remember in GNU radio, we are using 32000 samples per second.

Therefore substituting t as minus 512 upon 32 and so on yields 1024 values of t which are exactly 32 milliseconds long. Why is that the case? This is because 1024 divided by 32000 seconds is exactly 32 milliseconds. Therefore, we will have a sink which is truncated to 32 milliseconds duration approximately. The second thing is that the gap between any two samples is exactly 1 upon 32000 seconds and exactly 32 samples later you will get the next integer.

This will become adequately clear when we build this in GNU radio. Our first course of action is to import numpy that allows us to access numpy's numeric functions within GNU radio. So, let us hit Ctrl F or Cmd F and type import. We drag the import block into our flow graph. We now double click the import block and we will now use this block to import the numpy python module by typing import numpy.

After that we can say ok. The first thing that we will do is to construct the values of t that we wanted which is minus 512 upon 32 through 511 upon 32 step by 1 upon 32. An easy way to do this would be let us modify this t vals and we will modify this to numpy dot a range which gives us a numpy array consisting of a range of values from minus 512 to 511 through 512 divided by 32. What does this do? Numpy dot a range minus 512 to 512 gives me all integers ranging from minus 512 through 511. An important point is that 512 the last number is not included and divide by 32 divides each of these by 32.

Let us say ok. You will now see that t vals takes the values minus 16 minus 15.96 minus 15.93 and so on through 15.

96875. These are the locations where we are going to evaluate our sync. In our vector source, let us change this to evaluate a sync. How do we do that? We change the vector to numpy dot sync of t_vals. This essentially causes GNU radio to call the sync function on t_vals and replace those values as the source for this vector.

Let us say ok. Let us now execute our flow graph. What we obtain is a very nice sync. This sync has a peak at around 16 milliseconds at 1 which makes sense because we truncated our sync to 32 milliseconds and therefore, at exactly the halfway mark you will have the peak. At the next millisecond which is 17 milliseconds, you will find that the sync goes to 0. At 18 milliseconds the sync goes to 0 and so on. This is an elegant way to construct a sync and in spite of the sync being truncated, it is very evident that the sync characteristic is largely preserved.

If you want a more accurate sinc, you can keep increasing the number of samples from 1024 to larger values. However, this would increase the complexity of the simulation. Let us now close this. Our next task will be to make this sync a variable amplitude sync. To do so, let us first introduce a QT range by typing control F or command F and range.

We will drag the QT GUI range over here. Let us double click on this range, change the id to a1, make the default value 1. We will make it start at around minus 2, stop at 2 and make the step 0.1. And we will now double click our vector source and make this a1 times numpy dot sinc.

Let us now execute this. You will now see a range on the top that allows you to alter

the amplitude. Let us reduce the amplitude a little.

At minus 0.2, you can see that it goes negative. If you make it 1.4, you can see that it goes positive. The reason why the sync gets offset is because whenever you change the value of a1 using the range, the whole vector source is reevaluated and therefore, the synchronization may not be preserved. To avoid this, you must perform the multiplication outside, but for simplicity, we will retain this structure. Let us now close this and let us now add a frequency sink to observe the frequencies characteristic of this sync. We press control F or command F and type freq and we grab the QT GUI frequency sink into our flow graph.

Let us change some characteristics for convenience. We can double click. We will first change the type to float, window type to rectangular. We will add a grid and we will set the auto scale to yes. We can say ok. We can now connect our source to the sink and if you now execute the flow graph, you will see that there is a very nice rectangle like shape showing the sinc.

There are these minor oscillations. This is because of the Gibbs phenomenon that occurs due to the fact that you have truncated the sinc. This effect can become less pronounced, but will never go away because the moment you truncate the sync, you will never get a perfect rectangle. The next course of action for us is to add the second source. Let us just copy this source by selecting it and hitting control C and control V or command C and command V if you are using Mac. An alternate approach, let me redo this, would be to select this source, going edit, copy and then just saying edit, paste and that will also give you the source.

Now we want to change this vector source to yield A2 times

$\text{sinc}(t-1)$.

So, we change the appropriate parameters by double clicking on this vector source and saying ok. This becomes red because the range A2 is not specified. Let us easily just take this range and duplicate it by pressing control C or command C and control V or command V and changing the name to A2. Let us also change the default value to minus 0.5 since this will make it consistent with the example that we just saw in our lecture. We will say ok. We will then now change the GUI time sink and frequency sink to display all our signals. Let us double click the QT GUI time sink to display for three inputs, the frequency sink to display for three inputs. Let us connect this vector source to the time sink input 2, this vector source to the frequency sink input 2 and finally, the sum of these two to the third input. To add these two, we will use the add block that we can use by, we can access by pressing control F and typing add.

We can drag the add block over here. As always we double click it and make it float and say ok. We connect the vector source over here. We connect this vector source to the second input. We connect these outputs to the third input of the time sink and the frequency sink.

Let us now run this flow graph. If you run this flow graph, you will see something interesting. The blue curve is the original sinc which is at 16 milliseconds. The red curve is the second sinc whose peak amplitude is minus half. It is exactly 1 millisecond delayed from the blue sinc.

If you observe this sum which is the green one, that has the value of minus 0.5 at 17 milliseconds. It has the value of 1 at 16 milliseconds which means it is exactly of the form

$$\text{sinc}(t) - 0.5\,\text{sinc}(t-1)$$

and it displays the other sinc like characteristics that is it goes to 0 at integer millisecond values. In the spectrum you see something interesting.

The red one which is the

$$-0.5\,\text{sinc}(t-1)$$

looks rectangular in shape. The blue one which is

$$\text{sinc}(t)$$

looks rectangular in shape. But the green one does not look rectangular in shape because it is a combination of a sinc and delayed sinc with a different amplitude. Therefore, unlike a regular sinc, it has some additional information embedded in it indicating these amplitudes of 1 and minus 0.5. This concept will be used heavily in digital communication to construct band limited waveforms that convey useful information across time.

Now we can start playing with the amplitudes. If you now change A1, you will see that A1's amplitude changes. If you now change A2, A2's amplitude changes. And you will see that in spite of all these amplitude changes, the resulting summation is still band limited to be between

-0.5 KHz and 0.5 KHz

because we chose a sinc which is sinc of t upon t which is capital T is 1 millisecond. Therefore all the waveforms and their combinations are bound within

-0.5 KHz and 0.5 KHz.

Of course, it is not a brick wall, you have this kind of decay.

This can be attributed to the fact that we are truncating the sinc. Therefore, this example clearly demonstrates that you can construct waveforms that are band limited yet can convey useful information across time. Thank you.