Digital Communication Using GNU Radio

Prof Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

Week-02

Lecture-12

Welcome to this lecture on Digital Communication Using GNU Radio. Continuing with our exploration of evaluating digital communication methods using GNU radio. In this lecture, we are going to start with quadrature amplitude modulation. This approach can be viewed as a combination of both amplitude shift keying as well as phase shift keying that you have viewed in the previous few lectures. There are several advantages in terms of the way signals can be generated in this approach are, and in particular, this provides a more efficient way to transmit information as well as we shall see when we discuss demodulation. The other thing that we will explore as part of this lecture is aspects related to frequency shift keying wherein variations in frequency are used to convey information and that is a special form of orthogonal modulation and this also has several tradeoffs as we shall explore when we look at demodulation.

In the past couple of lectures, we have seen modulation formats amplitude shift keying and phase shift keying. In the case of QAM that is quadrature amplitude modulation, this can be viewed as a combination of both amplitude and phase shift keying. As an example, the QAM 16 constellation is shown here. In this case, there are 16 symbols laid out in the form of a grid.

A sample bit allocation for the symbols is also provided. As you can see, purely using the phase as criterion will not allow you to decode these symbols because this particular symbol and this particular symbol both of them have the same phase. Therefore, you need a combination of both the phase as well as the amplitude in order to decide which symbol was sent. The reason quadrature amplitude modulation is sometimes preferred over PSK that is phase shift keying is that this arrangement of constellation points as we shall see is more robust to noise than when you compare it to the phase shift keying analog that is phase shift keying 16 in this case. In this case as well, we are still performing one dimensional signaling where

$\psi_1(t)$

is related to

$g_{TX}(t)$

and

$b[k]$

is one of these 16 symbols.

Therefore, you can assume that you have one signal as the basis signal and these

$b[k]$

are essentially values that are taken by the this signal and these values essentially with magnitude and phase convey the bit information. Let us now just modify our previous GNU radio simulation which involved PSK to accommodate QAM. Before we move to GNU radio, let us observe how these symbols are laid out. For example, this symbol is 1 +1j, this symbol is 3+1j, this symbol is 3+3j while this symbol is 1+3j. So in essence, this constellation takes the values -3, -1, 1, 3 on the real or I axis and -3, -1, 1, 3 on the imaginary or the Q axis.

This is useful for us to be able to construct this constellation in GNU radio along with the chunks to symbols block. In GNU radio, if we now recall our PSK example, we just need to make two changes. The first is that this variable m has to become 16. So let us double click it and make it 16 to indicate that we are going to have one of 16 possible symbols. We shall say ok over here and the next is that in the chunks to symbols, we must enumerate these 16 possible symbols.

For that, we can remove this existing symbol table by selecting it and hitting delete. Then we can start typing -3-3j, -3-1j, -3+1j, -3+3j then -1+3j, -1-3j, -1-1j and so on and enumerate all the 16 symbols. However, we can use a bit of python trickery to do this in a more neat way. I will show you how. Let us expand this a little bit and we will write a python expression that will simplify the generation of the symbol table.

So, we can say a plus j times 1 j times b for a in and we will say minus 3 minus 1 1 b for b in minus 3 minus 1 1 3 and we must change the output type to complex. Now this is actually a list comprehension that takes all possible values in a that is minus 3 through minus 1 and all possible values in b that is minus 3 to minus 1 1 3 and takes all combinations and constructs a list. If you put your mouse over it, you will see that you get minus 3 minus 3 j minus 3 minus 1 j minus 3 plus 1 j and you get all of these and the length is 16. This is much less error prone although it uses python. But if you want to enumerate it fully, you can go ahead and do that as well.

Let us say ok and now we are done and we are ready to look at how our QAM constellation and QAM modulation looks like. Let us execute this flow graph. Now in this case, let us start by looking at this particular place. Let us again set the delay to 75 that we match this. Now in this case, what do we see? We see that these waveforms take 1 2 3 4 possible values which are minus 3 minus 1 1 3 and the other waveform also take similar values.

If you look down at the constellation, if you do the auto scale, you will see 1 of 16 possible values appearing in this constellation. Finally, how does this look in the case of the carrier waveform? When you modulate the carrier, then remember that in this case, this is a combination of amplitude and phase shift keying. Therefore, you are seeing over here, there is a phase change as well as an amplitude change. Over here again, let me zoom in and show you, there is a phase change as well as a magnitude change. That is what the feature of QAM is, in which case you are changing both the phase and the amplitude at every time interval.

This is 10 milliseconds, this is 11 milliseconds, this is 12 milliseconds. So, we are still sending symbols at the rate of a 1000 symbols per second, but each symbol when you look at the carrier looks in this form, wherein both the amplitude and phase undergo a change. And as you can see from this squiggly waveforms, you are able to detect the, rather you are able to receive the signal and recover them. Of course, your choice of the rectangular waveform leads to the slight oscillations. Nevertheless, these are recoverable and reasonably match the transmit waveform.

So, one small lesson is to consider the QAM 16 modulation constellation as a combination of two amplitude shift keying constellations. That is, if you look only at the real part, it is minus 3 minus 1 1 3, only on the imaginary part it is minus 3 minus 1 1 3. So, it is almost as though you are stuffing an amplitude shift keying signal on the real axis or in phase part and an amplitude shift keyed signal on the imaginary axis or the quadrature part. And this is a simple way to understand how QAM works. More complicated QAM constellations exist, typically even powers of 2 although odd ones exist some examples include QAM 64 and say QAM 256 and so on.

To visualize it, let us just do one thing really quickly. We will first just make the constellation sink auto scale for convenience and let us make a let us say QAM say minus 3 minus 1 1 3. Let us make a QAM 64 constellation. In the case of QAM 64, we are going to have not 4, but 8 possible values. So, let us generate these somewhat automatically.

You say np dot arange. For simplicity, I am just going to go from 0 to 8 and I am going

to subtract, subtract 3.5 because 0 through 7 the midpoint is 3.5 and let us do this in the second part as well. Let us put this around parenthesis.

It should be numpy. So numpy dot a range 8 gives us numbers from 0 through 7 and you subtract 3.5 that gives you numbers minus 3.5 minus 2.5 minus 1.5 minus half half 1.5 2.5 3.5 and this gives you another 8 and thus you have an 8 cross 8 64 constellation and one more change is that we need to make M as 64. And now if you execute this flow graph, you get all 64 possible constellation points and as you can see the number of amplitudes if you look carefully for the red and blue curves will be 8 possible amplitudes namely minus 3.5 minus 2.5 minus 1.5 and so on up to 3.5 and thus this is a combination of an M-ASK that is 8-ASK on the real, 8-ASK on the imaginary. So two amplitude shift keyed waveforms one on the real, one on the imaginary. So this is a summary of the quadrature amplitude modulation and its key differences between plane phase shift keying and amplitude shift keying. The last basic digital modulation format that we will be looking at is frequency shift keying wherein change in the frequency of the signal is used to convey information.

For a simple description of an M-FSK system, you can define

$$s_1(t) = e^{j2\pi f_1 t} I_{[0,T]}$$

indicating that this

$$I_{[0,T]}$$

limits this function to between 0 and T. In other words,

$$s_1(t)$$

is this

$$e^{j2\pi f_1 t}$$

between 0 and T and 0 elsewhere.

$$s_2(t)$$

is

$$e^{j2\pi f_2 t}$$

within the duration of one symbol. So this

$$I_{[0,T]}$$

is essentially a rectangular pulse between 0 and T. Similarly,

$$s_M(t)=e^{j2\pi f_M t} I_{[0,T]} \quad .$$

For a simple visualization, we can look at a 2-FSK system wherein to send 0, you send this particular cosine while to send 1, you send this particular cosine. These two cosines have a distinct property in that in the duration of one symbol which is from here to here, you have two cycles while for sending 1, you have four cycles. So whenever you get two cycles, you can conclude that the bit or symbol 0 was sent. Whenever you have four cycles within the same interval, you can conclude that the bit or symbol 1 was sent. Therefore, by varying the frequency within the duration of a symbol interval, you are able to send different symbols.

Now in this case, unlike ASK, PSK and QAM, the dimension of signaling is actually more. The reason is because this particular cosine assuming that it completes two cycles and this particular cosine assuming that it completes four full cycles within the duration of one symbol are essentially orthogonal. Therefore, to model this particular FSK system in our signal basis picture, we need to have two particular basis signals,

$$\psi_1(t)=\sqrt{2/T}\, I_{[0,T]}\cos(2\pi f_1 t) \quad ,$$

$$\psi_2(t)=\sqrt{2/T}\, I_{[0,T]}\cos(2\pi f_2 t)$$

assuming that

$$f_1 T \quad \text{and} \quad f_2 T$$

are integers. This means that the signals which are actually being sent are proportional to

$$\begin{bmatrix}1\\0\end{bmatrix} \quad \text{or} \quad \begin{bmatrix}0\\1\end{bmatrix}$$

because you are either sending

$$\psi_1(t) \quad \text{or} \quad \psi_2(t)$$

with of course some appropriate scaling, but this corresponds to orthogonal signaling because you are either sending

$$\begin{bmatrix}1\\0\end{bmatrix} \quad \text{or} \quad \begin{bmatrix}0\\1\end{bmatrix}$$

which corresponds to sending

$\psi_1(t)$

for that duration or

$\psi_2(t)$.

So, this is an example of two dimensional orthogonal signaling and there are various tradeoffs that are associated with this that we will look at when we look closely in the demodulation of the frequency shift keyed symbols.

For now, let us inspect how we can put together a frequency shift keyed signal in GNU Radio. One minor aspect is that you can also write this as

$\cos(2\pi f_1 t)$ $\cos(2\pi f_2 t)$

and so on without much of a loss because these are essentially conveying the same information. For example,

$s_1(t) = e^{j2\pi f_1 t}$

its real value also similarly conveys the same information. In general, if you use

$e^{j2\pi f_1 t}$

it is not very common to use

$e^{-j2\pi f_1 t}$

unless you use bi-orthogonal signaling which is something we will look at later. Let us now put together a FSK modulated system in GNU radio.

Let us now start working on an FSK system on GNU radio. To build this, we will be using a new GNU radio block called VCO short for the voltage controlled oscillator. Let us first look at how it works and then build our FSK system. So I am going to first do control f for command f and grab VCO and take the VCO object and place it in our flow graph. I will get a throttle so control f for command f, Throttle make the throttle float. We will do a real FSK experiment as opposed to a complex one. Let us also grab a time sink so control f for command f and then we will do a short for command f as opposed to a complex one. Let us also grab a time sink so control f for command f time sink. Double click this, make it float, add a grid, make it auto scale. We will connect these three items and for the VCO we need three parameters.

Double click it. First we need the sample rate. Let us first set the sample rate to 64000 for easier visualization and come here and set the sample rate to samp underscore rate.

The next thing is sensitivity. So the voltage controlled oscillator essentially gives you a signal of the form cos of 2 pi t times whatever is the sensitivity. In other words you are able to control the frequency as a signal.

So whatever input you give you will get as a frequency. So let us actually call this let us say 2 times let us say 3.1415 because we are approximating pi times a 1000. In other words we are going to get whatever f we give gets multiplied by 1000. So we are going to actually choose our f as 1 and 2 which will correspond to 1000 hertz and 2000 hertz.

Great. Now let me add a coordinate function. So let us say constant source. So control f or command f constant source double click this make it float and make it value 1 and connect this over here. Save it. Now let us say that the amplitude is 0.

We have to set the amplitude to 1. If you execute this flow graph you will get a sinusoid. Of course there is some bit of movement that is because we did not do the pi correctly. We can actually handle it. Let us actually do control f or command f say import. We will grab here double click it and say import numpy and we will change the sensitivity to 2 times numpy.pi times 1000. Let us execute this now. Now it is reasonably stable and if you zoom in you can see that this is a cosine 9 milliseconds 10 milliseconds a cosine of exactly 1000 hertz. If I change this constant source to 2 you can see twice the number of cycles. This starts at 9.5 this starts at 10 this is at 10.5. So this is a cosine for of 2 kilohertz. In other words your understanding is that whatever the sensitivity is gets multiplied by the input and that becomes the frequency of your cosine. This is used very commonly in the context of phase lock loops and frequency trackers as well. We are using the voltage controlled oscillator to produce a frequency shift keyed signal.

Let us now replace our constant source. With an actual useful source that we want. So we will use first a random source. So control f or command f, Random source. Minimum is 0 maximum is 2. It will give us values of the form 0 and 1. We will next grab our chunks to symbols. So control f or command f, Grab the chunks to symbols. Place it on our flow graph. And the chunks to symbol we will double click. Make it float. And the symbol table we will make is a 1 and 2. Because we want a 1000 hertz and a 2000 hertz sinusoid or cosines.

So we will make this 1 and 2. Dimension is 1. We will connect this over here. Connect this over here. But before we go ahead we do not want to vary the sinusoid very very quickly and in fact we have a mandate to make this particular symbol rate a 1000 symbols per second. Therefore we are going to hold this particular value for a 1000 symbols.

So I am going to get an interpolating FIR filter. Control f or command f, Interpolating

FIR filter. Much like we have been doing so far.

Let us make this float. Let us make the interpolation. Sample rate double divide 1000. This will give us sampling rate divided by 1000 which is 64 in this case. And we need 1s of the exact same number. This essentially holds this value and therefore will hold the frequency of the voltage controlled oscillator for 1 millisecond. So connect these and now if we visualize we will now get our frequency shift keyed waveform.

Let us stop this and analyze. Over here 6 milliseconds over here is let us say here is 6 milliseconds here is 7 milliseconds. You have about 1 cycle which means that this is exactly a 1 kilohertz cosine. But over here let us say between 8 milliseconds and let us say this 8 milliseconds and 8.5 milliseconds again you guess I guess you have 1 cycle this corresponds to a 2 kilohertz sinusoid.

So between 8 and 9 milliseconds you have 2 cycles. Again here from 9 milliseconds to 10 milliseconds you have 1 cycle. From 10 milliseconds to 11 milliseconds you have 2 cycles. So this way you are able to encode the 0 as one particular frequency and the 1 as another frequency. So this essentially is your frequency shift keyed waveform.

Let us also inspect the baseband spectrum. Control F or command F and type fraq. Let the QT-GUI frequency sink float and connect it. You will see if you execute this that there are these 2 peaks one is at minus 1 another is around minus 2. So therefore, there is a strong presence of 1 kilohertz and minus sorry 1 kilohertz and 2 kilohertz which is explained by the fact that these are the 2 signals that we are essentially sending.

I will now remove this QT-GUI frequency sink. Now in order to go to passband and come back to baseband we will use the same procedure. We will first multiply this voltage controlled oscillator by an fc valued signal carrier meaning a carrier whose frequency is fc. Then again multiply it by a similar carrier and we will not use a sinus-sin analog because we are only doing real signaling. So there is only I and no Q and we will try to recover the information.

Let us go about this in the exact same way that we have been doing it. So, control F for command F, signal source, we will double click the signal source, make it float and make the frequency fc. Of course, we need to declare a variable called fc. So, control F for command F and say variable, grab the variable over here, call it fc and make it 8000 hertz as we have been doing so far. Next we say multiply, so control F for command F, multiply, get the multiplication block, double click it, convert it to float and we will multiply the output of our voltage controlled oscillator with this signal source.

Let us also change the amplitude of the signal source to root 2, so 1.414 and this will

become our modulated passband signal. Let us just copy this time sink by doing clicking on it say control C or command C and go to C command V visualize this. Visualizing this clearly shows that the carrier is modulated and if we stop this and view it, we will see that sometimes there are close peaks, sometimes there are far off peaks indicating that the frequency is slightly changed. So, therefore, what you will end up having is fc and affected by 1 kilohertz and fc affected by 2 kilohertz. Therefore, 8 kilohertz is effectively going to become 7 or to 9 or 6 to 10 kilohertz.

This is something which you can confirm by looking at the frequency domain waveform for this. Let us now demodulate this or rather bring this down convert this waveform to the baseband. Let us remove this QT-GUI time sink. We will duplicate this signal source by doing control C and control V. We will again duplicate this multiplier control C, control V and grab this multiplier over here.

We will multiply this signal with the output of the signal source and we will also add a filter to remove the components near 2 fc. So, control F or command F, low pass filter, grab the low pass filter, double click it. As always we will add a low frequency to fc and transition will make it a 1000 hertz.

We will also make it float to float. We will connect it over here. Now, we need a delay. So, let us again do the same thing. I am going to delete this. I am going to do control F or command F, grab delay and in this case I am not adding a range. I am just going to hard code the delay as 75, connect the throttle to the input, connect this to the output.

Finally, increase the number of inputs over here and connect this output over here. This was a mistake, apologies. This output should go here. Now, if you visualize the output, you can clearly see that the baseband waveform has been recovered that is the red one and is reasonably close to the original waveform indicating that we have been able to recover the signal carefully. You will not see significant aberrations in this particular situation although they are there, but the frequency spectral of your occupation does correspond to a rectangular pulse in this case as well.

So, in this way you can put together a frequency shift keyed waveform. If you want, you can expand the random source and this table to have more frequencies as well. For example, if we make this 4 and if we make this 1, 2, 3, 4, 5, 6, 3 and 4, you will now get 4 frequencies. You can see that there are now a multitude of frequencies and you can analyze and find out that these are 1 kHz, 2 kHz, 3 kHz and 4 kHz. If you look at the frequency spectrum, you will see something very similar. In this lecture, we have carefully gone through both quadrature amplitude modulation and frequency shift keying as you would have observed both in the baseband as well as the passband waveforms.

In case of quadrature amplitude modulation, you are able to see both the amplitude changes as well as the phase changes. In the case of FSK or frequency shift keying, you have seen that within a particular symbol block, just estimating the frequency is able to convey information and in this case, you are essentially using frequency variations in order to convey information. Of course, you can extend this to say amplitude plus frequency shift keying as well and many such variations, all of which are used and which have their own advantages and disadvantages. One major limitation that we encountered in our past few lectures is that the use of the rectangular signal results in those ringing effects primarily because it is not band limited. In practice, having a band unlimited pulse

$g_{TX}(t)$

is not practically feasible because of the fact that you have spectrum restrictions.

So, in the next few lectures, we will explore ways by which you can get inter-symbol interference free communication that is how we can recover these symbols while not using a rectangular waveform and being within the spectral constraints that are imposed upon us. Thank you.