Digital Communication Using GNU Radio

Prof Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

Week-02

Lecture-11

In the previous lecture, we have seen how you can alter the amplitude of a signal in order to convey a discrete set of messages. In this lecture, we will be using an approach wherein the information that you want to convey is embedded in the phase of the waveform that is we take a set of messages and generate different waveforms for each message and these waveforms differ in the phase. This approach is called phase shift keying as you have seen in the lecture. This approach is very distinct from the amplitude variation that you have seen in the previous lecture. Again, we will generate a PSK modulated digital signal. Look at how it can be transmitted to pass band, recover it again at the base band and recover the original transmitted symbols when using a rectangular pulse.

The next digital modulation approach that we will consider is phase shift keying. Here the symbols phase conveys information and the different values of phases correspond to different symbols. For example, if we define

$b[k] \in e^{j2\pi k/N}$,

where

$k = 0,1,2,\ldots,N-1$,

then there are N possible phase values and each of these corresponds to a different symbol. This particular approach can convey one of N different symbols.

Let us now take the example of 8 PSK. Let us now take the example of 8 PSK. Here one of 8 possible phase values equally spaced give you the symbol. For example,

$e^{j2\pi 0/8}$

is this particular symbol,

$e^{j2\pi 1/8}$

is this particular symbol,

$e^{j2\pi 2/8}$

is this symbol and so on. As you can see if you ascribe bits to each of these 000, 001, 011 and so on, then one of three possible bits are conveyed by these symbols.

In this case as we mentioned only the phase and not the amplitude conveys the information. Therefore, just by checking the phase of the received symbol you should be able to recover your bits. In terms of our signals perspective,

$s_k(t) = b[k]\,\psi_1(t)$,

where $\psi_1(t)$ is the waveform that you use for signaling. For example, it can be a rectangular pulse or a sinc or so on and it is a one dimensional signal, because the same signal is being scaled by just a unit magnitude and phase varying complex number. Let us now explore how we can put together a PSK simulation on GNU radio.

We will be keeping with the same parameters that we have been using that is a carrier frequency of 8 kilohertz and a symbol rate of a 1000 symbols per second. Let us begin. Let us first set our sampling rate to 64000 as we have been using so far. Next we will first find a way to generate our random PSK symbols. Before that let us also fix the PSK size.

We use PSK 4 at the beginning, but in order to have flexibility over the PSK size let us create a variable. Control F for command F, we will say VAR and we will grab the variable. We will call this variable capital N as we chose in our slides as well and keep it as 4. Now, let us first grab a random source, so that we can generate random PSK symbols. Control F or command F, random we have our random source that we will bring here.

Random is 0, maximum we will set as N which is what we chose over here as the variable and let us make it 1024 samples with repeats, so that we can get a constant stream of random symbols. We will say ok. Next we need a chunks to symbol that can map the number 0, 1, 2, 3 to one of the PSK symbols. So, let us grab a chunks to symbols, control F for command F, C H U N and we will grab this chunks to symbols, double click the chunks to symbols, the dimension will be 1. Now for symbol table it is inconvenient for us to have to write the symbol table every time we change N.

In fact, the very reason why we use these variables is for convenience in GNU radio, so

that the symbol tables and such other dependents can be automatically recalculated. So, before we complete our chunks to symbols, let us first use the power of python, we will import numpy for our assistance. So, control F for command F, we will say import, grab the import block, we will then say double click on this and type import numpy. This allows us to use the numpy functions to create our symbols very very easily. Now, let us double click on our chunks to symbols, our symbols are just

$e^{j2\pi k/N}$ where $k=0,1,2,\ldots,N-1$.

To get the numbers from 0 through $N-1$, we can say numpy dot a range N. This gives us the values which k can take. Now, we must do $j2\pi k/N$. So, let me just do 1j into 2 into numpy dot pi times this upon n and this belongs within a numpy dot exp. So, I will put this in brackets and I will say dot exp.

So, if you see what we have done, we have done numpy dot exp of j 2 pi k/n. Let us say ok and now let us connect our random source and this chunks to symbols should be connected to a throttle. So, control F for command F, we will grab our throttle. Let us then have a time sink and see what these symbols look like. So, control F for command F and we will grab a QT-GUI time sink.

And if we execute this, we will see some things essentially looking like this. The values which the blue which is the real part and the red which is the imaginary part take are either plus 1 or minus 1 or 0 that is what we see over here. But what is more useful is if we grab a QT-GUI constellation sink, control F for command F and constellation sink. If we look at the constellation sink, this will tell us the real part and imaginary part simultaneously and as you can see you have 4 possible values which correspond to

$e^{j2\pi 0/4}$, $e^{j2\pi 1/4}$

which is exactly j,

$e^{j2\pi 2/4}$

which is -1 and

$e^{j2\pi 3/4}$

which is -j. So, these are the 4 possible symbols that are being produced by the chunks to symbols module.

This makes complete sense if because we chose n as 4, we get these 4 symbols. If you choose n as 8, you will get one of 8 symbols. Next our goal will be to send these symbols

at 1000 symbols per second. To do so, let us first remove this QT-GUI time sink and place an interpolating FIR filter. So, let us make some space control F for command F, interp and grab this interpolating FIR filter.

We will connect the throttle to this interpolating FIR filter, double click this, set the interpolation to we need a 1000 symbols per second. Therefore, we must interpolate it by samp rate divided by 1000. So, we will say samp rate double divide because we want integer division by 1000 and the taps we want exactly all ones because we want to use rectangular pulse shape. Therefore, I will just say numpy dot ones samp rate. Now if we grab a time sink, control F for command F and time sink and place the time sink over here and execute this flow graph.

Let us stop this. You will see that each of these symbols begins at 9 milliseconds, ends at 10 and then this begins at 1 millisecond sorry 30 milliseconds, ends at 14. So, each of these symbols essentially takes two values and these values are separated by exactly 1 millisecond. To get a better idea of why these are the exact values, it is very easy to check. Our symbols are essentially 1, j, -1, -j. So, when you have a 1 or -1, then the imaginary part essentially is 0 which is why whenever the red is high, you can see that the blue is 0 or red is low, blue is 0.

When you have j or -j, the real part is 0 which means whenever the blue is high or low, the red is 0 which makes complete sense. In other words, you can easily verify that the complex signal being shown here, its values are exactly one of these constellation values. For example, over here the blue value is 1, the red value is 0. That means this is 1+j0 which corresponds to this symbol. Over here, the blue value is -1 while the red value is 0 which means it corresponds to this symbol.

Finally, over here the red value is 1 while the blue value is 0. This corresponds to j which is this symbol. So, this makes complete sense. Our next task will be for us to take this to the carrier frequency and then get back the baseband signal. For that let us create a variable called fc.

So, you would say control f or command f, variable, grab this variable, double click it, call it fc and value is 1000 sorry 8000. Ok. Now our task is for us to modulate this particular waveform. If you remember to modulate this, we can multiply this by

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

and take the real part. We are going to take that approach, but as the flow graph becomes messy, we do not want to take wires across.

So, let us use virtual sinks and sources. So, I am going to use control f or command f, type virtual, I will grab a virtual sink and I will also place a virtual source in anticipation. I will double click this virtual sink, call it psk baseband, connect it over here and double click this and call it psk baseband. As you can see the color gets automatically chosen based on what you connected. The next task for us is to modulate this or multiply this by

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

and as we have seen in the past, a simple way to get

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

on GNU radio is to grab a signal source, control f or command f, grab the signal source, drag it over here and we keep it as a complex signal source, ok.

We will keep the frequency as fc, ok, amplitude can be 1 that is ok. Now what we will do is we will multiply these two. So, control f or command f, we will grab the multiply, we will multiply these two. This is essentially

$$x(t)e^{j2\pi f_c t} \quad ,$$

then we need the real part. So, control f or command f, type real and we get complex to real, connect this and then we will grab a time sink, control f or command f, QT-GUI time sink and let us set the time sink to float.

Let us add a grid, let us add auto scale and connect our signal over here and execute this flow graph. Now, you will see that we get this kind of pass band signal. Now, this pass band signal looks innocuously like a cosine, but there is something special. Let us actually stop it and zoom in a little. So, exactly at 10 milliseconds there is a phase change and then we have a nice sinusoid and again exactly at let us say 12 milliseconds there is a phase change.

Of course, even at 11 there could have been, there was not. That could be because the phase was retained as is. Let us continue running this and we will you know. So, you have these potential phase changes happening. For example, over here at 14 milliseconds there is a phase change and 15 milliseconds there is a phase change.

So, the information is essentially conveyed within the phase of the waveform. This phase variation essentially captures the phase variation of the base band waveform. In fact, if you write out the equations you will find that the actual pass band waveform you get looks like

$$\cos\left(2\pi f_c t + \phi(t)\right) \quad ,$$

but this

$\phi(t)$

essentially has this particular phase within each symbol duration. So, as you can see at around 3 milliseconds, 4 milliseconds, 5 milliseconds. So, every 1 millisecond there is a phase change that conveys the symbol and by observing the phase within each of these intervals you will be able to find out which waveform or rather which symbol is being sent during that interval.

Now, however, our task is for us to get back some waveform that looks like this because this is our base band waveform. The question is how do we recover this kind of waveform from this pass band signal. For that we must take this pass band waveform multiplied by

$\sqrt{2}\cos(2\pi f_c t)$

multiplied by

$-\sqrt{2}\sin(2\pi f_c t)$

and filter out the 2fc components and then combine them to get back our symbol signal. Let us do that. Let us make some space in our flow graph by just making things a little more compact, right.

So, now what we need to do is we need to take this particular symbol and we need to this signal rather and multiply it by

$\sqrt{2}\cos(2\pi f_c t)$ and $-\sqrt{2}\sin(2\pi f_c t)$.

So, let us grab a signal source. So, control F or command F, signal source, grab a signal source, place it here, double click it, we will have a real signal source, frequency will be fc, amplitude will be $\sqrt{2}$ and then we will create a copy, control C or command C, control V or command V, place it just below, double click this. We want a sign and we want a negative sign so that we can get

$-\sqrt{2}\sin(2\pi f_c t)$.

Now we will grab a pair of multipliers. So, control F or command F, multiply, we will double click this, we call it float. Now we are going to multiply the signal source with the received signal. We will copy this, control C or command C, control V or command V, connect this over here, connect this over here and now we will create a complex signal out of these two after taking out the 2 fc component. Remember your passband

signal has an fc component, these cosines and sines have an fc component.

So, we need to take out the 2fc component. So, let us grab the interpolating FIR filter. So, control F or command F, in type interpolating FIR filter, we will double click this. Oh, let us not use an interpolating FIR filter, I am sorry, let us use a low pass filter. So, control F or command F, type low, we will grab the low pass filter because we just need to specify some few components over here. We will set the cutoff frequency as fc because we need to remove the 2fc component and we will set a transition width of about a kilohertz.

We will call it float to float interpolating. We will copy this control C, control V for paste or command C, command V for Mac users. Connect these over here because we need these two parallel filters and we will then combine these two waveforms into a single complex waveform. So, control F or command F. So, we will grab the float to complex over here, connect these and to visualize our symbol on the same time sink, we can double click this. We will set grid S, auto scale S and we will set two inputs and we will connect this over here and we can run this flow graph.

Now, if you run this flow graph, you can see that there is a some very minor scaling issues. The scaling issues are because we did not take care of the root 2 when we constructed the modulation waveform. So, let us first address that. So, over here we will choose this as 1.414 to make sure that we have the correct energy. Now, we have amplitudes correct, but there is a certain non-overlapping between 3 signal 3 and 4 and signal 1 and 2 as well as some distortion. The distortion is very simple because we are using rectangular signaling. This rectangular signal when passed through the low pass filter at the receiver side when coming back to baseband causes those high frequency components to get removed and therefore, you have this Gibbs phenomenon like effect. That is something that we will ignore, but more importantly they are not aligned. Why? The reason is because this low pass filter causes some delay, it is a causal implementation and it causes some delay.

So, one way to address this would be to just delay the output of the interpolating FIR filter over here and then just see how we can align them. So, let us see. Let us add a QT range to adjust the delay. So, control F or command F, type range QT-GUI range and we will double click it and call it delay and we will type make the type as integer default value 0 stop 100 step 1 this is fine.

Now, we are going to add a delay. So, control F or command F type delay, we will grab the delay over here and we will set the delay to actually the delay variable that corresponds to the range. Connect the output of this interpolating FIR filter to the input here, connect this over here. Now, if we execute the flow graph, you get this range and

because the FIR filter has a group delay of roughly 75, if you make this delay 75, you will now see that there is a reasonable overlap. Let us actually stop this. Signal 1 and signal 2 correspond to the original input baseband waveform.

Signal 3 and signal 4 are what you get when you go to pass band come back to baseband after those two filtering operations. Let us compare signal 1 and signal 3. So, if you look at signal 1 and signal 3, they have a beautiful overlap barring that little oscillatory behavior because of your filter. Similarly, if you look at signal 1, signal 2 and signal 4 rather, they also exhibit very close relationship. So, therefore, what have you done? You have gone from a pass band signal back to baseband and if you overlap the signals, it is very evident that the received signal and the transmit signal are almost the same.

The reason for the slight difference is because of the fact that the high frequency components are getting eliminated by the low pass filter. If you really want to honor the frequency constraints and make these waveforms match, then you must design your transmit waveform using a pulse other than a rectangular pulse, so that you have a band closer to band limited signal. One exercise that you can do is to observe the frequency components of these signals. You will find that they are very close except that the low pass filtering causes the rectangle in your signal 3 and signal 4 to be slightly distorted. As a final step, let us just modify this n to 8 by double clicking and setting it to 8 and run our simulation.

Now, you can see that there are more wave phase variations that are being observed and if you see our constellation, it takes one of 8 possible values and our waveform also seems to indicate that there are more values in the baseband. Why is this the case? The reason is because the values which you now take are for let us say the imaginary value is +1, over here it is $1/\sqrt{2}$, here it is 0, $-1/\sqrt{2}$ and -1. So, these are possible values. So, if you stop and let us look at the original waveform, you can see that it takes 1, then it takes 0.707 which is $1/\sqrt{2}$, 0 and then it takes, over here it takes -1, -0.707. So, as you can see more values are there and you can see different kinds of phase variations as well. So, if you write, if you make your flow graph rather in a very generic manner, you can easily simulate various other PSKs. Let us also check out PSK 16 and you will see that there are 16 possible phases and you can see different kinds of phase variations. In this lecture, we have put together a phase shift keying or PSK based system. In this system, we have essentially embedded the messages that you wish to transmit in the phase of the waveform and you have seen that both in the baseband waveform and in the passband waveform, variations in the phase can be used to infer information.

In the next lecture, we will be exploring different modulation formats, particularly what happens when you combine phase shift keying and amplitude shift keying in which case

you end up with quadrature amplitude modulation. We will also explore an orthogonal modulation scheme called frequency shift keying wherein the phase shift keying.