

Research on key digital modulation techniques using GNU Radio

Tianning Shen

Yuanchao Lu

I. Introduction

Software Defined Radio (SDR) is the technique that uses software to realize the function of the traditional radio. It can get code as close to the antenna as possible, define the transmitted waveforms, and demodulate the received waveforms. Moreover it is powerful for multimedia transmission through wireless channels with different environment and suitable for testing dynamic wireless video performance under software-controlled parameters.

GNU Radio, a free/open source software toolkit of signal processing for building software radios, provides functions to define the transmitted waveforms demodulate the received signals and support a spectrum analyzer which can show the change of the concurrent multi-channel receiver and an ever-growing collection of modulators and demodulators.

Basically software radios can replace the tradition radio. Instead of expensive fixed gadgets, the flexibility of the software is more charming. It is fascinating to see that you can use just one portable device to listen to radio, watch TV, and determine your location using GPS. In this challenging area, we want to find out the performance of different modulation methods.

In this project, our first goal is to implement the $\pi/4$ -DQPSK modulation and demodulation method to transmit data. Although the DQPSK modulation and demodulation method is already there, but the codes is connected based on the graph in the old version of GNU radio. We need to change the way of their connection to base on the blocks in the new version. Then, we want to compare the performance of the different modulation methods in BER (bits error rate) and PSD (power spectral density) which are two important parameters to evaluate the quality of the communication system.

The report is organized as the following: the second section introduces the background. Section 3 describes how we implement the $\pi/4$ DQPSK. Next Section discusses the performance of the three different modulation methods, DBPSK, DQPSK and $\pi/4$ DQPSK. The fifth section is the conclusion and future work.

II. Background

The GNU radio is an open project which is still under developing, so source code release is updated at times. Lots of modulation methods have been implemented, like QPSK, DBPSK, DQPSK, and GSM. You can download these source code from internet or you can implement

what you want using your own code. In our study, we will implement the $\pi/4$ -DQPSK to do the modulation and the demodulation and then compare it with DBPSK and DQPSK.

In DBPSK, a binary '1' is transmitted by adding 180° to the current phase and a binary '0' by adding 0° to the current phase. In DQPSK, the phase-shifts are $0^\circ, 90^\circ, 180^\circ, -90^\circ$ corresponding to data '00', '01', '11', '10'.

$\pi/4$ -DQPSK is a kind of PSK (Phase shifted keying) which uses the information of the phase to transmit bits and used in the North American second generation IS-54 digital cellular system. It is the $\pi/4$ shifted QPSK combined with differential encoding. Figure 1 shows the Constellation diagram for $\pi/4$ -QPSK. It uses two QPSK signal constellations offset by $\pi/4$ relative to each other, shown in Figure 1, one QPSK (the blue points), for odd-numbered symbols and the other (the green points) for even-numbered symbols. The two constellations are used alternately. In differentially-encoded QPSK, the phase-shifts are corresponded to bits which are transmitted. Table 1 shows the detailed information of $\pi/4$ -DQPSK encoding and it is easy to figure out how the DBPSK, DQPSK and $\pi/4$ -DQPSK encode from Figure 2.

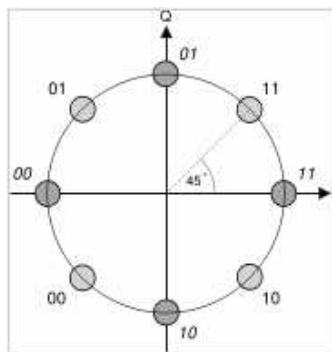


Figure 1 Constellation diagram for $\pi/4$ -QPSK¹

<i>Phase changing</i>	<i>Bits</i>
0	00
90	01
180	11
270/-90	10

Table 1

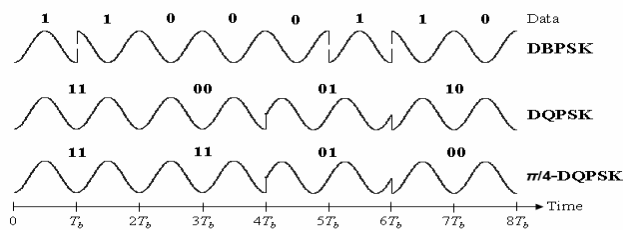


Figure 2 $\pi/4$ -DQPSK encoding²

¹ The figure is from reference 1.

There are some different performance between DBPSK, DQPSK and $\pi/4$ -DQPSK. First, DBPSK transmit one bit each time, but DQPSK and $\pi/4$ -DQPSK can transmit two bits each time. So the DQPSK and $\pi/4$ -DQPSK have the higher spectrum efficiency. Second unlike DBPSK and DQPSK, there is no phase change of π between adjacent symbols in $\pi/4$ -DQPSK. This is the reason that, after being amplified by non-linear amplifier, $\pi/4$ shifted DQPSK has a smaller envelope variation than DBPSK and DQPSK. So $\pi/4$ -DQPSK has narrower bandwidth than DBPSK and DQPSK, but higher BER (Bit error rate) than DBPSK. These are the advantages and disadvantages for traditional radio program. Since the software will take the hardware job, our interesting is to see whether these advantages and disadvantages still have strong infection or not.

III. Implement $\pi/4$ -DQPSK

First of all, we implemented the $\pi/4$ -DQPSK modulation and demodulation method to transmit a group of random numbers or an audio file which is larger than 1 Mbits through the AWGN communication channel using GNU software. The `piover4dqpsk.py` was created based on `mydqpsk.py`[2]. Then the data was changed to Gray code by using `odd_psk.py` which also was created by us. In this new gray code method, the transmitted bit sequence, say (0, 1, 2, 3) is mapped to (1, 3, 7, 5), which should be (0, 1, 3, 2) with the traditional gray code method. After feeding the bit streams into a differential encoder, we realized the first step in the $\pi/4$ -DQPSK implementation. After that, we used the 8-PSK constellation map to do the encoding. At the receiver, we only need a $\pi/4$ offset QPSK constellation to map the input bit stream to the corresponding binary number because after differential decoding, the phase of the symbol is chosen from only 4 candidates that are $\pi/4$, $3\pi/4$, $5\pi/4$ and $7\pi/4$. We only have to change the phase shift parameter for the block `gr.constellation_decoder_cb`.

IV. Performance

1) BER

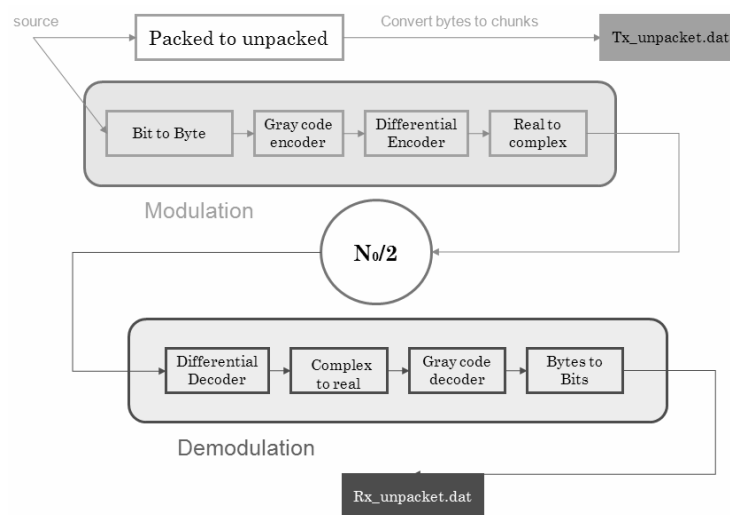


Figure 3 Flow chart of testing BER

² The figure is modified and the original one is from reference1.

Figure 3 shows the flow chart of testing the BER. At first, we used the *random.int()* function to generate a series of data, like [0,3,1,1,2,3,0...2] as the source which were converted to bits using *packed to unpacked* block and recorded in *Tx_unpacked.dat*. The reason we used packed to unpacked block to transform the content of the file into the form of bits is that calculating BER is to divide the number of the wrong bits received by the number of transmitted bits, but the contents in the transmitted file are in the form of byte, so we have to extract every bit in each byte and put that bit at the end of a new byte, the rest of which is flushed with 0. Since some functions blocks of we used the same as [10], we do not want to explain them again. The signals were transmitted through the AWGN channel to the receiver. At the receiver side, the received bits were put into *Rx_unpacked.dat*. Similar to the principle at the transmitter, a block named *unpack_k_bits_bb* is used to transform every gray decoded byte into the form of bits. In our experiment setup, we choose the AWGN channel because it is ubiquitous. Although striking speaking, the real AWGN channel is not existed, lots of channels, such as line-of-sight (LOS) radio channel [7], are approximately equal to AWGN. So it is necessary to see the performance of the different modulation methods under the AWGN channel. The noise can be introduced in the AWGN channel. The noise also can be changed by increasing or decreasing value of the parameter E_b/N_0 which is the ratio of the intensity of the signals and the noise. When we got *Tx_unpacked.dat* and *Rx_unpacked.dat*, the BER can be calculated by using the Number of different bits from two files divided by the Number of bits from *Tx_unpacked.dat*.

In our project, we tested 21 different points each modulation method then draw the figure of BER vs E_b/N_0 for all the modulation methods. The result is shown on Figure 4.

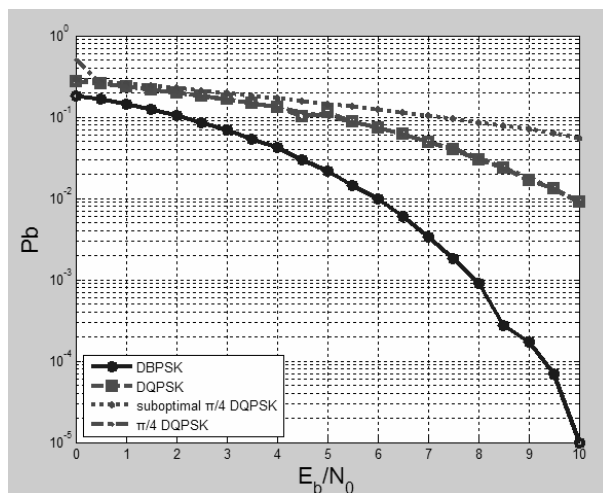


Figure 4 BER of all the modulation methods.

From the figure, we can easily see that the DBPSK has the lowest BER, DQPSK and $\pi/4$ -DQPSK almost have the same bit error rate but higher than DBPSK and suboptimal $\pi/4$ -DQPSK is the worst one. That is because the points in DBPSK constellation map have larger distance between each other than other three ones. Shorter distance brings more

mistakes. We got the same result as those from the traditional communication system. Figure 5 shows the result in theory of the traditional system.

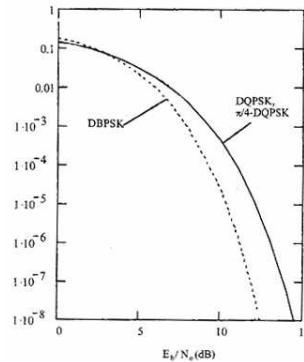


Figure 5 BER in theory.³

We also used a .wav audio file as the source to see the changes of BER. It got the same result. And by playing the received the audio file, we still can hear the music but with lots of noise when the E_b/N_0 equals to 6.0db for DBPSK, 10.0db for DQPSK and $\pi/4$ -DQPSK. When the ratio is increased to 12.0db, we almost could not hear the noise for DBPSK. However this number has to be increased to 15.0db, if we want to get the same result from DQPSK and $\pi/4$ -DQPSK.

There is one more block we have to introduce in order to restore the original audio file at the end of the receiver. That block is `gr.unpacked_to_packed_bb` which takes the inverse effect as that of `gr.packed_to_unpacked_bb`. This block packed every 8 bytes into one byte which is an integer in the range of 0 to 255 that could be recognized by the audio sink device so as to play the audio file.

2) Spectrum

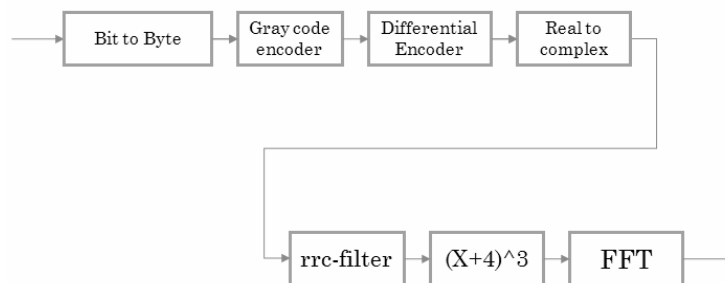


Figure 6 Flow chart of testing Spectrum

The Figure 6 shows the blocks which are used in testing the Spectrum. After the `rrc-filter`, we use a non-linear amplifier to magnify the modulated signals. Then the signals were processed by the USRP broad before transition. We first used software to see the simulation result shown in Figure 7. (a) All the signals are amplified by the linear amplifier; (b) they are

³ The figure is modified and the original one is from reference7.

amplified by the linear amplifier. In (a), they are the same, and in (b), the last one $\pi/4$ -DQPSK has narrower bandwidth. But it is not noticeable.

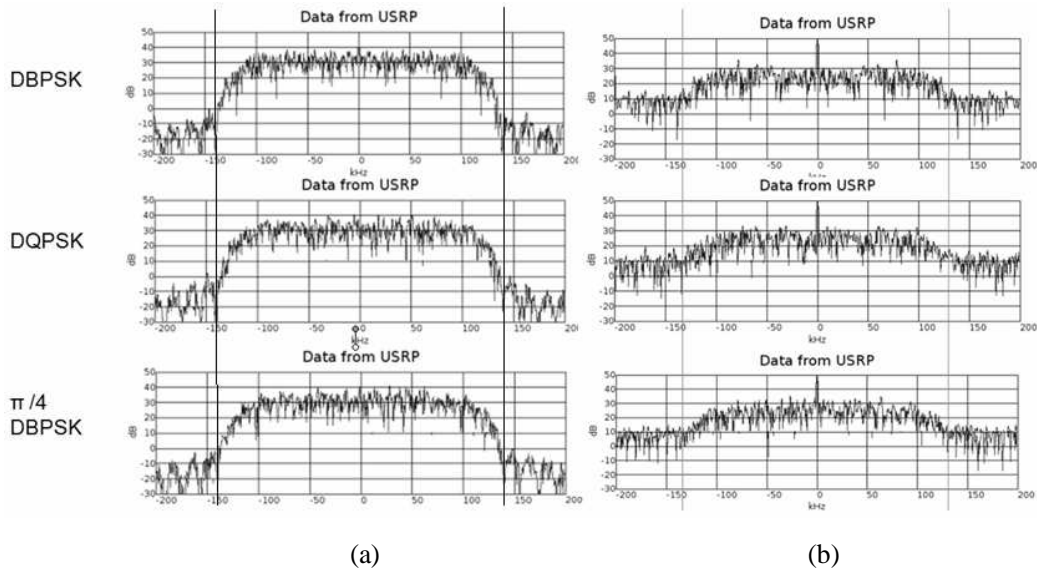
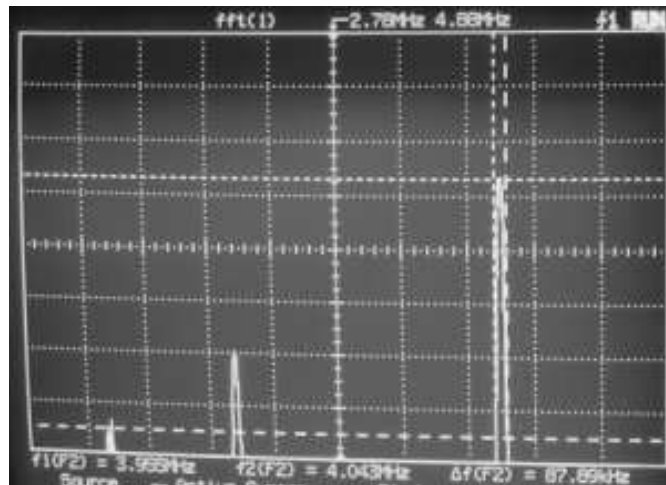
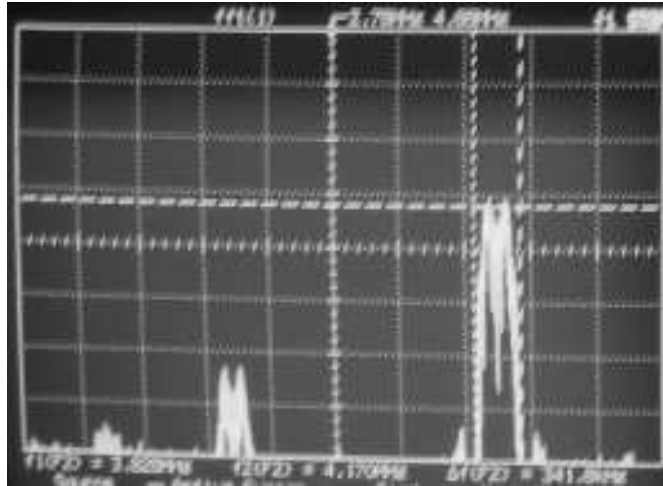


Figure 7 Simulated result of spectrum

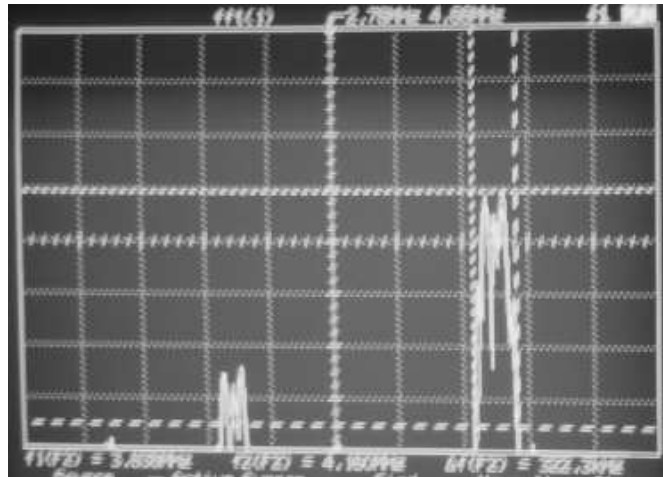
We also used the oscilloscope to see the real spectrum at the antenna. It is shown in Figure 8. The bandwidths of these spectrums are recorded when power spectrum density is equal to -50db.



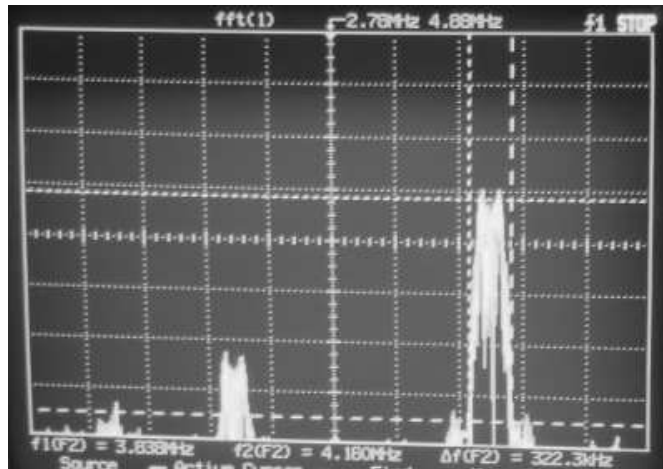
(b) Sine



(b) DBPSK



(c) DQPSK



(d) $\pi/4$ -DQPSK

Figure 8 result of spectrum

As can be seen, (a) shows the spectrum of the carrier which is a sine wave with frequency 4MHz.

The spectrum bandwidth of sine wave shown on the oscilloscope is 87.89 KHz, very narrow with respect to the central frequency. (b) Shows the spectrum of the DBPSK modulated waveform with central frequency equal to 4MHz. As we can see, the spectrum has been widened to 341.5 KHz. (c) shows the spectrum of the DQPSK modulated waveform with central frequency equal to 4MHz. As we can see, its spectrum is around 322.3 KHz. (d) shows the spectrum of the $\pi/4$ -DQPSK modulated waveform with central frequency equal to 4MHz. As we can see, its spectrum is also around 322.3 KHz. The effect of narrowing bandwidth of $\pi/4$ -DQPSK modulation technique is not obvious. One reason of this may be that the waveform resides in the linear amplifying range of the non-linear amplifier when it is being amplified.

V. Conclusion and Future work

In this project, the concept of software radio was introduced with USRP as its hardware support and Python as its development tool. Then, we simulated a communication system with GNU Radio Software and analysed the BER performance of 3 digital modulation techniques, within which $\pi/4$ -DQPSK had been implemented. We also tried to test their spectrum efficiency using the software shown on the computer and Oscilloscope. At last, an audio recorder and an audio player were implemented so as to let you feel how White Gaussian Noise affects the quality of sound.

So the affection of disadvantages of the BER is much more than the advantages of PSD in GNU Radio. For GNU Radio, the first advantage of the $\pi/4$ -DQPSK which is easy to implement into the systems no longer exists. If the system require the low BER, it is better to choose the DBPSK. If the system require higher frequency efficient, it is better to choose $\pi/4$ -DQPSK.

For future work, we plan to transmit an audio file through two USRP daughter boards and compare the quality of sound received from the Rx board and that got through computer simulation.

References

- 1 http://en.wikipedia.org/wiki/Phase-shift_keying
- 2 <http://www.wu.ece.ufl.edu/projects/softwareRadio/#Project%20paper>
- 3 <http://www.gnu.org/software/gnuradio/>
- 4 <http://www.nd.edu/~jnl/sdr/docs/>
- 5 <http://gnuradio.org/trac/wiki/UbuntuInstall>
- 6 Wireless Communications and Networking, Weihua Zhuang 2003
- 7 Digital Modulation Techniques (654 pages), Artech House, Boston/London, Fuqin Xiong. 2000
- 8 <http://www.swaroopch.com/byteofpython/>
- 9 DQPSK Format for Serial PHY, Marcus Duelk, Peter Winzer, 2006
- 10 GNU Radio, Ke-Yu, Chen and Zhi-Feng Chen

APPENDIX

All the python codes are included in the packet.

The folder *Transmitter* includes all the codes of the three modulation methods, and *Receiver* has all the demodulation files.

All our experiment can be re-achieved by following the following steps:

1 BER

Go to the director *BER*, run *Main.py*.

First, input the number of integer you want to transmit;

Second, choose the modulation method following the hit of the dialog box.

Third, choose the corresponding demodulation method and the bit energy to noise ratio, and then press the button of “calculate the BER”, you will get the answer.

Forth, with respect of different bit energy to noise radio you input, corresponding BER will be derived.

2 The affection of BER

Put the audio file which you want to transmit to the folder *audio* and then run *Main.py*

1 Choose the modulation method.

2 Input the value of E_n/N_0

3 Transmit

The audio file will be transmitted through the channel to which you introduced the noise and saved as . Then you can play it to see the influence.

3 Spectrums

For the spectrum, go to the folder transmitter, run *Main_Tx.py*.

For the test by the Oscilloscope, go to the director *Transmitter*, run *Main_Tx.py*.

We try our best to annotate our each code file and make the interface of our litter software nicely and humanity.