

# Analysis of modulation scheme using GNU radio

Zeyu Long Department of Electrical and Computer Engineering CSU

## I. Introduction

Software Defined Radios (SDR) system is a wireless communication system which can generate any frequency band and receive different modulations across a large frequency spectrum by means of a programmable hardware which is controlled by software. The impetus of SDR research is a desire to shift the radio engineering problem from the hardware problem to the software problem [1, 2]. The establishment of software is to use programming languages, for example, GNU Radio has to use both C++ and Python.

Software radio is a revolution in radio design due to its ability to create radios that change on the fly, creating new choices for users [3]. GNU radio is one of most popular applications of SDR. GNU Radio is a Toolbox for building software radios, a platform for experimenting with digital communications and a platform for signal processing on commodity hardware [4]. GNU Radio create a practical environment for experimentation & product delivery, expand the “free software ethic” into what were previously hardware intensive arenas, and also can transmit/receive any signal.

GNU Radio provides a framework for building software radios [5]. The basic procession blocks in GNU radio are written in C++. Each block has different functions and free for download. Python is the language to connect these blocks and generate a flow graph to run a GNU radio application program. SWIG works as glue between the blocks written by C++ and the flow graph applications written by python.

In this project, the main purpose is to compare the different modulation schemes (for examples, QPSK, OQPSK and CPFSK) by using GNU radio, to analyze the advantages and disadvantages of these modulation schemes with their spectrums.

## II. Background

### Background of GNU Radio:

GNU Radio is an open source SDR project and Universal Software Radio Peripheral (USRP) is the hardware designed specifically for use with the GNU Radio software, it means, the SDR is an exciting field, and GNU Radio provides the tools to start exploring, and then use USRP to realize it [7]. The following figure is the block diagram of GNU radio.

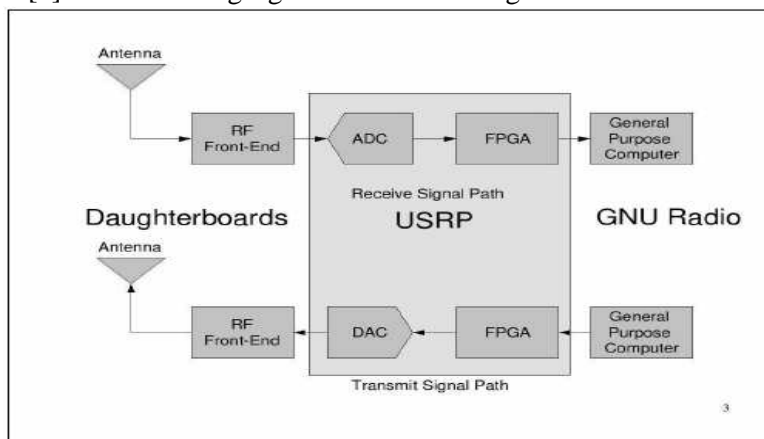


Figure 1: Block diagram of an SDR currently realizable using GNU Radio, the USRP, and associated daughterboard [7]

GNU Radio consists of a number of radio processing components referred to as blocks, which may be linked together to form a useful waveform. Traditionally, each of these blocks run in a single thread and a scheduler has been used to run each block's work task when it has a non-empty input queue.

GNU Radio is a hybrid system; the performance critical portions such as signal processing blocks are written in C++ while non-critical portions such as graph construction and policy management are written in the Python programming language. This allows developers to write highly optimized signal processing code in C++, but use the much more friendly language Python to construct applications. GNU Radio applications written in Python access the C++ signal processing blocks through interfaces automatically generated by SWIG for Python [3].

### Background of Modulation:

#### 1 QPSK (*quadrature phase-shift keying*)

QPSK is sometimes known as quaternary PSK, 4-PSK, or 4-QAM; QPSK uses four points on the constellation diagram, equispaced around a circle. Although QPSK can be viewed as a quaternary modulation, it is easier to see it as two independently modulated quadrature carriers [8]. With this interpretation, the even (or odd) bits are used to modulate the in-phase component of the carrier, while the odd (or even) bits are used to modulate the quadrature-phase component of the carrier.

The equation of QPSK is defined as:

$$S_i(t) = \sqrt{\frac{2E_s}{T}} \cos\left(2\pi f_c t + \frac{2\pi i}{M} + \frac{\pi}{4}\right)$$

In this equation,  $E_s$  is the power per-symbol and  $f_c$  is the carrier frequency.

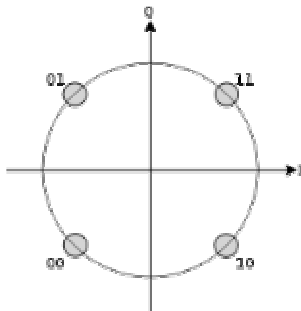


Figure 2: Constellation diagram for QPSK [8]

#### 2 OQPSK (*Offset quadrature phase-shift keying*)

Offset QPSK is a minor but important variation on QPSK. It's a variant of phase-shift keying modulation using 4 different values of the phase to transmit.

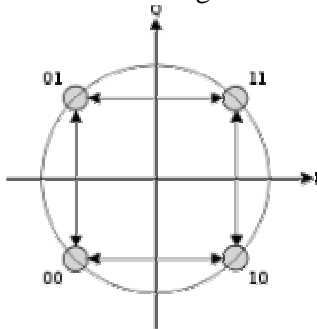
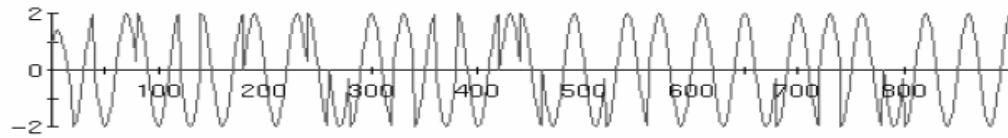


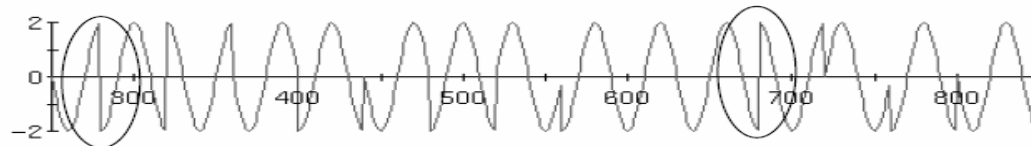
Figure 3: Constellation diagram for OQPSK [8]

The OQPSK Signal doesn't cross zero, because only one bit of the symbol is changed at a time.

And this kind of modulation can make the change of phase never larger than  $2/\pi$  [9]. The following figure compares the difference between QPSK and OQPSK, where QPSK goes through phase change of  $\pi$  for some transmission.



(a) OQPSK – All phase shifts are  $90^\circ$ .



(b) QPSK - Note the  $180^\circ$  phase shift.

Figure 4: *The compare of QPSK and OQPSK signals [9]*

### 3 CPFSK (Continuous-phase frequency-shift keying)

CPFSK is a commonly-used variation of frequency-shift keying (FSK), which is itself a special case of analog frequency modulation. In general, a standard FSK signal does not have continuous phase, as the modulated waveform switches instantaneously between two sinusoids with different frequencies. But as the name suggests, the phase of a CPFSK is in fact continuous.

$$S(t) = \sqrt{\frac{2E}{T}} \cos(2\pi f_c t + \phi(t, \alpha))$$

$$\phi(t, \alpha) = 2\pi h \int_{-\infty}^t \sum_{i=-\infty}^{\infty} \alpha_i g(\tau - iT) d\tau$$

$$g(t) = \begin{cases} \frac{1}{2T} & 0 \leq t \leq T \\ 0 & \text{elsewhere} \end{cases}$$

Figure 5: *the expression of CPFSK*

MSK can be viewed as a special case of CPFSK, with modulation index  $h=0.5$

### III. Project goals

In this project, the objectively is to plot the spectrum figures of QPSK and CPFSK. Since the modulation blocks of MPSK (When  $M=4$ , its QPSK) and CPFSK is already existed in GNU radio Top-block package and blks2 package. I may use these two blocks and some other source or sink blocks to connect my flow block.

### IV. Experiment Setup

1. At first, I need to install GNU radio to my computer. The Installing of GNU Radio and USRP on Windows are not yet routine. So I can only install it in UBUNTU system, which is a system based on Linux [10].
2. The second step is to read *psk.cc* and *cpfsk.cc* files, get some basic idea of these two blocks, knows how to use these blocks. For example, *cpfsk.cc* file contains the modulation part of CPFSK scheme. It allows a stream of binary input signal and separates it to I, Q channels then multiple the input signals with a special pulse shape.

3. Then write python files to draw the figures I need. The graphical modules in GNU radio are based on wxWidgets (or to be precise, wxPython), a platform-independent GUI toolkit [6]. To use the GNU Radio wxWidgets tools, I need to import some modules like: stggui2, fftsink2. In my experiment, I follow the example in *gnuradio-examples /python /audio /audio\_fft.py* to write my codes.
4. Use command *python \*.py* in the terminal, to run my program.

## V. Result

I compare the QPSK and CPFSK both in a high sampling rate and low sampling rate.

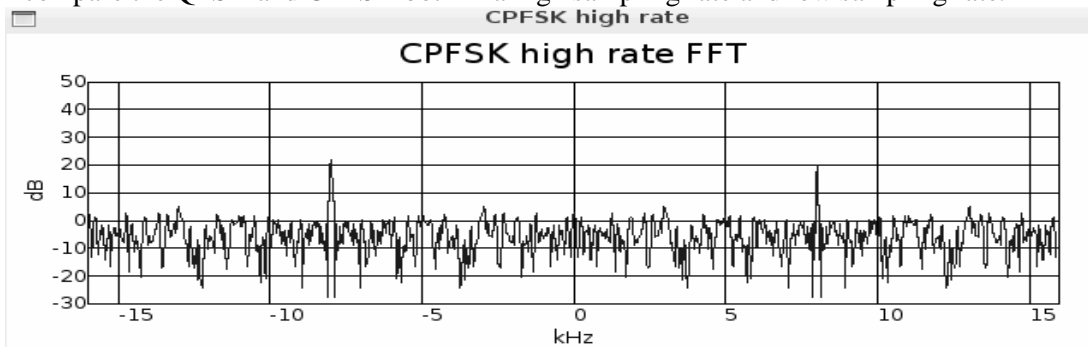


Figure 6: the CPFSK spectrum of high input signal rate

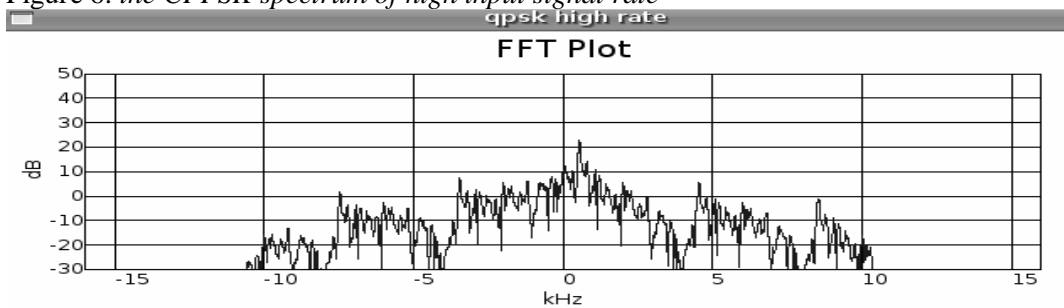


Figure 7: the QPSK spectrum of high input signal rate

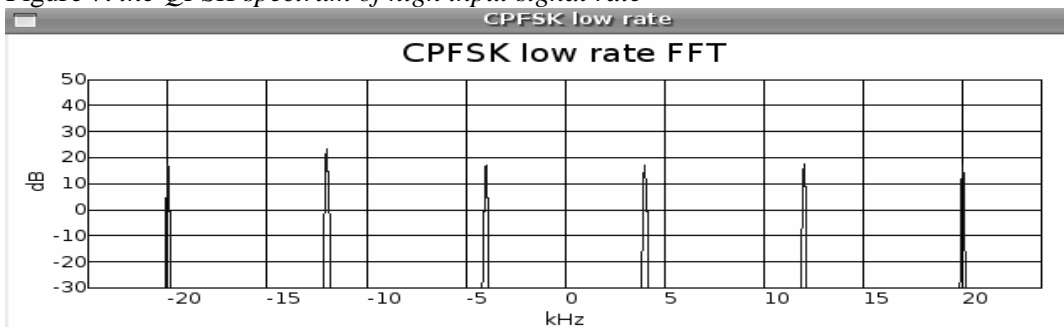


Figure 8: the CPFSK spectrum of low input signal rate

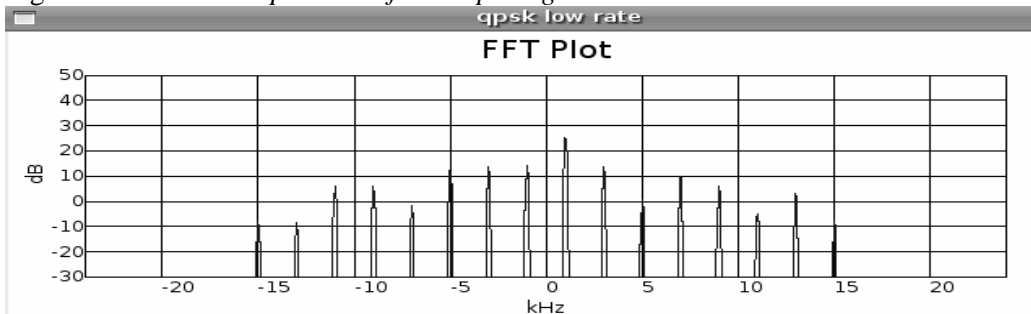


Figure 9: the QPSK spectrum of low input signal rate

## VI. Conclusion and Discussion

1. From figure 6 and 7, I find CPFSK occupies the whole bunch of bandwidth, but QPSK waste same period of bandwidth. That means the bandwidth efficiency of CPFSK is better.
2. From figure 8 and 9, it can be proved that, CPFSK has a spread spectrum. The advantage of spread spectrum is it can reduce noise and interference. So the bit error rate (BER) of CPSK is better than QPSK.
3. Also from the above figures, the spectrum of CPFSK is a flat no matter it is in high sampling rate or low sampling rate, but the spectrum of QPSK is a ramp. These figures indicate that the power of QPSK decreasing when the frequency is increasing. So, QPSK has a bad power efficiency, but CPFSK nearly doesn't loss power.

## References

- [1] Pala Trinadh, "Emergence of Soft Defined Radio", March 2009  
<http://www.hometoys.com/ezone/09.04/trinadh/index.htm>
- [2]<http://www.cs.uni-paderborn.de/en/research-group/research-group-computer-networks/projects/gsr.html>
- [3] Eric Blossom, "Exploring GNU Radio", 2004
- [4] Eric Blossom, "GNU Radio-Open Source Software and Hardware for Software Radio", 2002
- [5] Eric Blossom, "How to Write a Signal Processing Block", 2005
- [6] <http://gnuradio.org/trac/wiki/Tutorials/WritePythonApplications>
- [7] Lee K. Patton, "A GNU Radio Based Software-Defined Radar", 2007
- [8] <http://en.wikipedia.org/wiki/QPSK>
- [9] Fuqin Xiong, "Digital Modulation Techniques", 2000
- [10] <http://gnuradio.org/trac/wiki/UbuntuInstall>

## Apenndix

In my project, I think the difficulty is to write the python codes. After I learned how to write python in GNU radio, I think python is not a very difficult language.

The QPSK code is easy to write, because in *blks2* package has the *QPSK.py* file. I can easily reuse this file to generate my flow graph, but CPFSK hasn't such kind python file. So, the first challenge of the project to me is how to use block directly.

After I learned how to use CPFSK block, I began to write the source block and the FFT sink block. But I found it will gives out error if I use a signal source. So I go back to study the modulation blocks again, then I found that, the CPFSK and QPSK can only accept a binary input.

My way to generate a binary signal is to use a random source block. This block will output a series of number which is combined with 0 and 1. But the problem I cannot deal with is that, the random source block generates signals continually, and I don't how to stop it.

In my code package, there 7 files, to run the *qpsk\_low\_rate.py*, *qpsk\_high\_rate.py*, *CPFSK\_low\_rate.py* and *CPFSK\_high\_rate.py* can get my result.