

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 2028

**BASEBAND SIGNAL PROCESSING CHAIN FOR A
SATELLITE DIGITAL TRANSMITTER**

Mario Šimunić

Zagreb, June 2020

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 2028

**BASEBAND SIGNAL PROCESSING CHAIN FOR A
SATELLITE DIGITAL TRANSMITTER**

Mario Šimunić

Zagreb, June 2020

MASTER THESIS ASSIGNMENT No. 2028

Student: **Mario Šimunić (2401028282)**
Study: Electrical Engineering and Information Technology
Profile: Electronic and Computer Engineering
Mentor: prof. Davor Petrinović

Title: **Baseband Signal Processing Chain for a Satellite Digital Transmitter**

Description:

It is necessary to describe and simulate the digital signal processing chain for use in a digital satellite transmitter for transmission rates of up to 16 Mbps and BPSK, QPSK and O-QPSK modulations together with the transmitter filter. The carrier signal frequency is 10.45 GHz. The input to the system is a 16 bit (or less) parallel data stream. Show under what conditions, and with respect to the sampling frequency, amplitude resolution, number of filter states and memory size, unwanted emissions outside the transmission band of less than -40 dBc can be achieved. Define a digital-to-analog converter (specify and select the chip) whose output will be IQ signals representing the input of the modulator in the next stage of the transmitter. The processing chain shall be implemented in the Matlab software environment, to evaluate the computational complexity of signal processing and its feasibility on the STM32 or Zynq-7000 platform. By simulation, estimate the bit error rate for a given noise power at the receiver.

Submission date: 30 June 2020

DIPLOMSKI ZADATAK br. 2028

Pristupnik: **Mario Šimunić (2401028282)**
Studij: Elektrotehnika i informacijska tehnologija
Profil: Elektroničko i računalno inženjerstvo
Mentor: prof. dr. sc. Davor Petrinović

Zadatak: **Lanac za obradu signala u osnovnom pojasu u satelitskom digitalnom predajniku**

Opis zadatka:

Potrebno je opisati i simulirati lanac digitalne obrade signala za primjenu u digitalnom satelitskom predajniku za brzine predaje do 16 Mbps i modulacijske postupke BPSK, QPSK i O-QPSK te odašiljački filter. Signal nosioc je frekvencije 10.45 GHz. Ulaz u sustav je paralelni podatkovni tok širine 16 bita (ili manje). Pokazati pod kojim uvjetima, a s obzirom na frekvenciju očitavanja, amplitudnu rezoluciju, broj stanja filtra i veličinu memorije, se mogu ostvariti neželjene emisije izvan pojasa propuštanja manje od -40 dBc. Definirati digitalno analogni pretvornik (specificirati i odabrati čip) na čijem izlazu će biti IQ signali koji predstavljaju ulaz modulatora u sljedećem stupnju predajnika. Lanac je potrebno implementirati u programskom sustavu Matlab, procijeniti računalnu složenost obrade signala te izvodivost na platformi STM32 ili Zynq-7000. Simulacijom estimirati vjerojatnost pogreške bita u prijenosu (engl. bit error rate) za zadanu snagu šuma na prijemniku.

Rok za predaju rada: 30. lipnja 2020.

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND
COMPUTING

MASTER THESIS no. 2028

**Baseband Signal processing
Chain for a Satellite Digital
Transmitter**

Mario Šimunić

Zagreb, September 2020.

Hvala

*mentoru prof. dr. sc. Davoru Petrinoviću
na iznimnom strpljenju i povjerenju
prof. dr. sc. Dubravku Babiću na podršci i
motivaciji.*

*Ovaj rad posvećujem voljenoj Evi
koja mi je tako mnogo pomogla.*

CONTENTS

List of Figures	vi
List of Tables	viii
1. Introduction	1
2. Digital Communications Theory Overview	2
2.1. Continuous and discrete time signals	2
2.1.1. Modulation and complex envelope of bandpass signal	6
2.2. Digital modulation formats	7
2.2.1. BPSK modulation	8
2.2.2. QPSK modulation	10
2.2.3. Offset QPSK (O-QPSK) modulation	11
2.3. Nyquist filter and pulse shaping	11
2.4. Peak to Average Power Ratio (PAPR)	13
3. Proposed system architecture	15
3.1. System requirements	15
3.2. Digital modulator architecture	16
4. Matlab simulation	17
4.1. Pseudo-random Bit-Stream generator	17
4.2. PRBS program implementation	18
4.3. Symbol mapping	20
4.4. Raised Cosine filter	22
4.5. Pulse shaping	23
5. Simulation results	27
5.1. Long RC filter span results	27

5.1.1. BPSK results	27
5.1.2. QPSK results	28
5.1.3. O-QPSK results	28
5.2. Shortening the RC filter span	29
5.3. Further shortening the RC filter span	31
6. Conclusion	33
Bibliography	34
A. Spectral emission limits for Radio-Amater Band in X-Band radio frequencies	37
B. Source code	39
B.1. Source code of main Matlab simulation script	39
B.2. Source code of PRBS generator	54
B.3. Source code of symbol mapper	57
B.4. Source code of filter design	63

LIST OF FIGURES

2.1. $\delta[n - \frac{f_s}{2}]$ signal	5
2.2. Spectrum of $\delta[n - \frac{f_s}{2}]$ signal	5
2.3. Single rectangular pulse signal	5
2.4. Spectrum of rectangular pulse	5
2.5. Sine wave signal	5
2.6. Spectrum of sine wave signal	5
2.7. Block diagram of modulation	6
2.8. Basic block diagram of direct conversion modulator	7
2.9. Message signal pulses representing binary "1" and "0"	8
2.10. BPSK - modulated symbols "1" and "0"	8
2.11. BPSK signal space	9
2.12. QPSK signal space	10
2.13. Rectangular function - frequency response of a brick-wall filter	12
2.14. Graph of the normalized sinc function - impulse response of a brick wall filter	12
2.15. Frequency response of raised-cosine filter with various roll-off factors β [1]	13
2.16. Impulse response of raised-cosine filter with various roll-off factors β [1]	14
2.17. Envelope of QPSK modulation with ideal rectangular pulses	14
2.18. Envelope of QPSK modulation with RC shaped pulses	14
3.1. Block diagram of digital system for baseband signal processing	16
4.1. LFSR architecture [2]	17
4.2. LFSR polynomials [3]	18
4.3. PRBS and sequence periodicity [1]	18
4.4. BPSK mapped symbol pulses	20

4.5. QPSK and O-QPSK mapped symbol pulses	20
4.6. Impulse response samples of the designed RC filter	23
4.7. Magnitude and Phase response of the designed RC filter	23
4.8. BPSK upsampled symbol pulses	24
4.9. QPSK upsampled symbol pulses	24
4.10. O-QPSK upsampled symbol pulses	25
4.11. BPSK filtered and shaped I and Q impulses	25
4.12. QPSK filtered and shaped I and Q impulses	26
4.13. O-QPSK filtered and shaped I and Q impulses	26
5.1. BPSK signal transition in signal space	27
5.2. Envelope of the BPSK signal	27
5.3. BPSK signal spectrum	28
5.4. QPSK signal transition in signal space	28
5.5. Envelope of the QPSK signal	28
5.6. QPSK signal spectrum	29
5.7. O-QPSK signal transition in signal space	29
5.8. Envelope of the O-QPSK signal	29
5.9. O-QPSK signal spectrum	30
5.10. Magnitude response of 257 tap RC filter, span 16 symbols, $L=16$, $\alpha=0.22$	30
5.11. Magnitude response of 257 tap RC filter, span 16 symbols, $L=16$, $\alpha=0.30$	31
5.12. BPSK spectrum using 257 tap RC filter	31
5.13. QPSK spectrum using 257 tap RC filter	31
5.14. BPSK spectrum using 129 tap RC filter	32
5.15. QPSK spectrum using 129 tap RC filter	32
A.1. Spectral emission limits for 10 MHz amateur RF channel in X-Band using single channel communication and digital phase modulation . .	38

LIST OF TABLES

1. Introduction

Satellites are man made objects, ie. technical systems which orbits our planet. Essentially, there are four types of orbits: high Earth orbit (HEO), medium Earth orbit (MEO), and low Earth orbit (LEO). Many weather and some communications satellites tend to have a high Earth orbit, farthest away from the planet surface. Most scientific satellites, including NASA's Earth Observing System fleet, have a low Earth orbit [4]. Over last twenty years LEO have become increasingly interesting for academic satellite projects. Hundreds of nano-satellites, called Cubesat, with weight about one kilogram and volume of one liter are launched in LEO space over last ten years.

FERSat is academic project with objective to build own nano-satellite and launch it into LEO at about 600 km altitude. Nano-satellites such as FERSat have a few subsystems and one of them is Communications subsystem. Sensor and imaging data will be transferred over a high-bandwidth digital radio-frequency (RF) link. For that purpose FERSat need to have high-bandwidth digital RF transmitter onboard.

To achieve FERSat objective of building own RF transmitter subsystem it is necessary to understand practical limits of the signal processing communications system.

First part of this Thesis focuses on defining digital signal processing chain for M-PSK modulator. Simulation of 2-PSK (BPSK), 4-PSK (QPSK) and Offset-QPSK (O-QPSK) modulator is made in Matlab. Simulations gives an answer to question what is lower limits of processing complexity under which requirements are still satisfied. Simulation results of In-phase and Quadrature signal spectrum for a few different transfer rates will be compared. Second part of Thesis focuses on effects of limited digital word length, ie. limited memory register width.

Next chapters gives phase modulation and pulse transmission theory overview, disseminated modulator requirements and constraints, chosen modulator architecture, simulation explanation and results comparison.

2. Digital Communications Theory

Overview

In digital communications information is represented using binary coded words. This means that information or data in a computer is represented as discrete states of binary bits stored in memory registers. In digital computers, time and signals are discrete. Signals which are discrete in time and amplitude are called digital signals. During transmission of information through communication channel, information needs to be represented with kind of signals suited for particular communication channel.

The need to transfer information from eg. satellite to earth station computer basically involves series of data transformation. First of all, communication channel is analog RF signal, which means continuous-amplitude and continuous-time signal. In order to transmit digital signals, representing data, through analog communications channel they need to be converted into continuous time signals. During transmission it is necessary to conform regulations. Radio-frequency bandwidth is very constrained and limited. One of the most important parameters of RF transmitter is RF bandwidth used for transmission.

2.1. Continuous and discrete time signals

Continuous-time signal $u(t)$ can be represented with its spectrum $U(\omega)$. Connection between continuous-time signal and its spectrum is given by Continuous-Time Fourier Transform (CTFT) [5].

$$\text{CTFT}[u(t)] = U(\omega) = \int_{-\infty}^{+\infty} u(t)e^{-j\omega t} dt \quad (2.1)$$

Inverse Continuous-time Fourier Transformation is given by:

$$\text{ICTFT}[U(\omega)] = u(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} U(\omega)e^{j\omega t} d\omega \quad (2.2)$$

Digital signal $u[n]$, a series of data points describing time evolution of a real world phenomena, is obtain by taking samples of continuous signal at equidistant time points T_s . Sampling, a process of taking samples, is a discretization in time. After sampling, discrete time signals are quantized, that is discretized in amplitude, to B bits. B is word length of analog to digital converter.

Nyquist-Shannon sampling theorem establishes a sufficient condition for a sample rate that permits a discrete sequence of samples to capture all the information from a continuous-time signal of finite bandwidth. C. E. Shannon made sampling theorem in it's famous work [6] as Theorem 13. It states that if sampling of a bandlimited signal is done with sampling frequency $f_s \geq 2f_M$ where f_M is a maximum frequency component of the signal, then signal can be perfectly reconstructed. Sampling is written as in Eq. (2.3) [7, Eq.2.6] where $x^*(t)$ is sampled function, $f_s = \frac{1}{T_s}$ is a sampling frequency and $\delta(t)$ is a Dirac delta function.

$$x^*(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) \quad (2.3)$$

Spectrum of sampled function x^* is given in Eq. (2.4).

$$X^*(\omega) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X(\omega - k\omega_s) \quad (2.4)$$

It can be seen that sampled faction spectrum X^* have original spectrum $X(\omega)$ scaled by $\frac{1}{T_s}$ and periodized with frequency period ω_s . If Nyquist-Shannon sampling theorem is satisfied there is no overlapping between replicas of $X(\omega)$. Discrete time signal is $x_d[n] = x(nT_s)$

Discrete-time signal $u[n]$ can be represented with it's spectrum $U(\omega)$. Connection between discrete-time signal and it's spectrum is given by Discrete-Time Fourier Transform (DTFT) [5].

$$\text{DTFT}[u(t)] = U(\omega) = \sum_{n=-\infty}^{+\infty} u[n]e^{-j\omega n} \quad (2.5)$$

Inverse Discrete-time Fourier Transformation is given by:

$$\text{IDTFT}[U(\omega)] = u[n] = \frac{1}{2\pi} \int_{-\pi}^{+\pi} U(\omega)e^{j\omega n} d\omega \quad (2.6)$$

DTFT and IDTFT operates at infinite series signals. Infinite series signal practically

isn't achievable. In reality, computers work with limited amount of memory and limited number of signal samples. Thus, we must use different tool for transformations of finite-length discrete-time signals. This is Discrete Fourier Transform and Inverse Discrete Fourier Transform Eq. (2.7), [5].

$$\text{DFT}_N[u[n]] = U[k] = \sum_{n=0}^{N-1} u[n]W_N^{nk}, 0 \leq k \leq N-1 \quad (2.7)$$

Inverse Discrete-time Fourier Transformation is given by:

$$\text{IDFT}_N[U[k]] = u[n] = \frac{1}{N} \sum_{k=0}^{N-1} U[k]W_N^{-nk}, 0 \leq n \leq N-1 \quad (2.8)$$

$$W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \quad (2.9)$$

Fast Fourier Transform is a group of algorithms for efficient calculating DFT transform. It's important to notice few details about FFT and DFT to understand signal spectrum in simulations. First, discrete frequency have range from $-\pi$ to π . Whole frequency range $[-\frac{f_s}{2}, \frac{f_s}{2}]$ of continuous-time signal is mapped into range $[-\pi, \pi]$. DFT assumes periodic signal, ie. it assumes that finite-length signal $u[n]$ is exactly one period of a infinite-length periodic signal. If this isn't true, spectrum isn't perfectly clean.

DFT can be observed as array of matched filters, each of which is matched to k -th frequency. Input signal excites more or less each of the filter, and each filter output is proportional to how much signal frequency is matching that filters frequency.

Using finite-length sequence of infinite-length signal is a windowing operation. FFT by default operates with rectangular window. Windowing the signal produces amplitude and frequency effects in calculated spectrum of the signal. Rectangular window with amplitude of A and width T have frequency spectrum $X(\omega) = 2AT \text{sinc}(\frac{\omega T}{\pi})$. Thus multiplication of infinite-length signal with window in time domain produces convolution of signal spectrum with window spectrum in frequency domain. Eq. (2.10) is modified DFT equation which includes time window samples $w[n]$.

$$U[k] = \sum_{n=0}^{N-1} w[n]u[n]W_N^{nk} \quad (2.10)$$

From Eq. (2.10) it can be seen that amplitude of each spectral component k calculated by DFT is increased by a sum of window samples $w[n]$. To retain physical meaning of

spectrum amplitude, FFT need to be scaled by a sum of window samples. Corrected FFT transform is shown in Eq. (2.11).

$$U[k] = \frac{1}{\sum_{n=0}^{N-1} w[n]} \sum_{n=0}^{N-1} w[n] u[n] W_N^{nk} \quad (2.11)$$

There are three discrete-time signals for which it is important to know their spectra. Those signals are rectangular impulse, $\delta[n]$ (single non-zero sample) and sine wave signal.

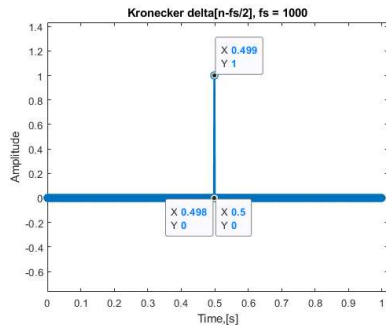


Figure 2.1: $\delta[n - \frac{f_s}{2}]$ signal

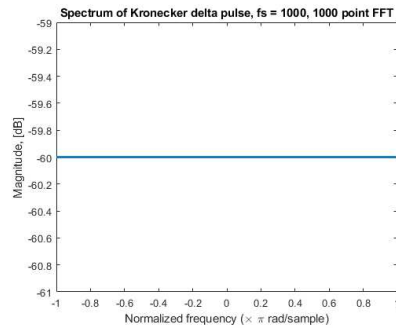


Figure 2.2: Spectrum of $\delta[n - \frac{f_s}{2}]$ signal

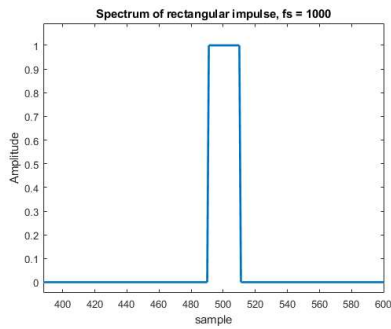


Figure 2.3: Single rectangular pulse signal

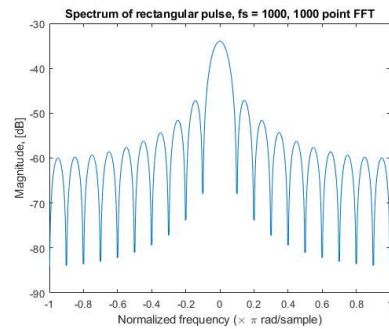


Figure 2.4: Spectrum of rectangular pulse

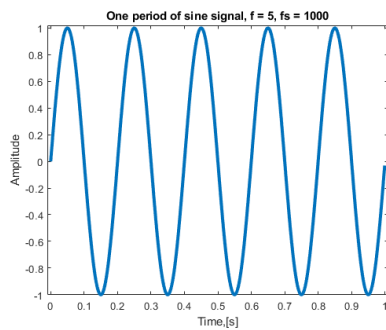


Figure 2.5: Sine wave signal

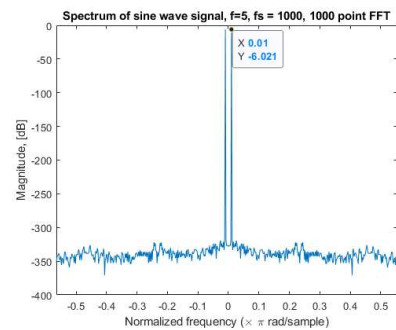


Figure 2.6: Spectrum of sine wave signal

2.1.1. Modulation and complex envelope of bandpass signal

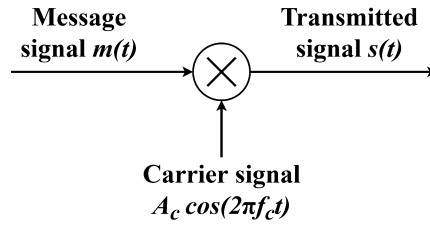


Figure 2.7: Block diagram of modulation

Modulation represented with block diagram Fig. 2.7 is a procedure in which modulation signal $m(t)$ is mixed with carrier signal. Modulation or mixing is mathematically multiplication. Modulation signal is carrying information or message. Carrier signal is RF signal tuned for RF channel frequency. Multiplication of carrier signal with $m(t)$ produces signal $s(t)$ which contains spectrum of $m(t)$ centered at carrier frequency, thus whole information signal spectrum is shifted into RF channel band.

Modulation can affect carrier signal frequency, amplitude or phase. Modulation formats which use phase of carrier signal as information carrier are called phase modulations. Keying is a modulation format in which message signal is pulse coded, for example Amplitude Shift Keying in which pulse can have amplitude of 0 or 1. These pulses are multiplied with carrier signal to produce ASK modulated signal. A passband is the range of frequencies that can pass through a filter. For example, a radio receiver contains a bandpass filter to select the frequency of the desired radio signal out of all the radio waves picked up by its antenna. The passband of a receiver is the range of frequencies it can receive when it is tuned into the desired frequency. A bandpass-filtered signal (that is, a signal with energy only in a passband), is known as a bandpass signal, in contrast to a baseband signal [8]. Baseband is a signal that has a near-zero frequency range. In telecommunications and signal processing, baseband signals are transmitted without modulation, that is, without any shift in the range of frequencies of the signal [9]. Signal $m(t)$ is basically baseband signal.

Direct Conversion modulation is one of the techniques which gives modulated signal using complex mixing or complex modulation. Fig. 2.8 represents basic block diagram of direct conversion modulator. Signal $u_{LP}(t)$ is a complex signal consisting of two signals. Real part of $u_{LP}(t)$ is signal $u_{LP_I}(t)$ or In-phase signal. Imaginary part of $u_{LP}(t)$ signal is $u_{LP_Q}(t)$ or Quadrature signal.

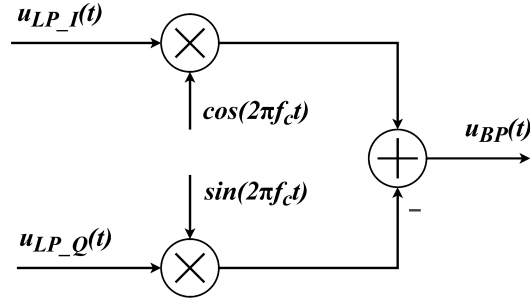


Figure 2.8: Basic block diagram of direct conversion modulator

$$u_{BP}(t) = \text{Re}[u_{LP}(t) \cdot e^{j\omega_0 t}] \quad (2.12)$$

$$u_{BP}(t) = \text{Re}[u_{LP_I}(t) + ju_{LP_Q}(t)] \cdot [(cos(\omega_0 t) + jsin(\omega_0 t)] \quad (2.13)$$

$$u_{BP}(t) = u_{LP_I}(t)cos(\omega_0 t) - u_{LP_Q}(t)sin(\omega_0 t) \quad (2.14)$$

Equation Eq. (2.14) is showing that any modulated signal can be achieved by using adequate I and Q signals mixed with complex exponential, that is cosine and sine signals with carrier frequency. Summation of I branch signal with negative Q branch signal results in modulated bandpass signal. Signal $u_{LP}(t) = u_{LP_I}(t) + ju_{LP_Q}(t)$ is a complex envelope of signal $u_{LP}(t)e^{j\omega_0 t}$.

2.2. Digital modulation formats

Among many keying formats this Thesis focuses on M-ary Phase Shift Keying (M-PSK). M in M-PSK represents order of modulation. For M=2 modulation is 2-PSK or binary PSK (BPSK), and for M=4 modulation is Quadriphase PSK or QPSK.

Modulation states are represented with symbols instead of bits and symbols are actual information, that is signals which are modulated on carrier signal. Depending on order of modulation, one or more bits are packed into one symbol. This is done by mapping bits into symbols and representing symbols using vector space signals. Parameter $K = \log_2(M)$ defines how much message bits is packed into one M-ary modulation symbol. For example, 8-PSK, with M=8 transmits or modulates K=3 bits per symbol.

Assigning signals to symbols.

Functions $\phi_j(t)$ form the orthonormal basis of vector space. Signals $s_i(t)$ can be shown

as linear combination of $\phi_j(t), j = 1, \dots, M$.

$$s_i(t) = \sum_{j=1}^N s_{ij} \phi_j(t), 0 \leq t \leq T_b \quad (2.15)$$

Equation Eq. (2.15) for $N = 2$ resembles the direct conversion modulator in which basis functions are $\phi_1 = \cos(\omega_0 t)$ and $\phi_2 = \sin(\omega_0 t)$. Signals representing symbols are $s_{i1} = u_{LP_I}$ and $s_{i2} = u_{LP_Q}$.

Basis functions are chosen using equations Eq. (2.16), Eq. (2.15), and the facts that basis functions must be orthogonal and signals $s_i(t)$ are linear combination of products of multiplication basis functions with coefficients s_{ij} .

$$\phi_i(t) = \frac{g_i(t)}{\sqrt{\int_0^{T_b} g_i^2(t) dt}} \quad (2.16)$$

2.2.1. BPSK modulation

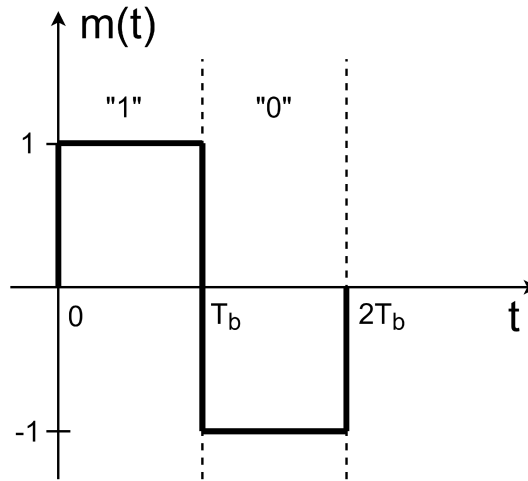


Figure 2.9: Message signal pulses representing binary "1" and "0"

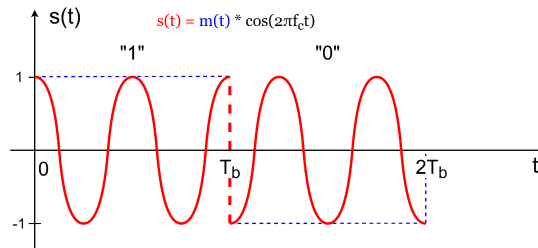


Figure 2.10: BPSK - modulated symbols "1" and "0"

Coherent BPSK is modulation with $M = 2$, thus two signals are used, s_1 and s_2 . Signals are given with the equations Eq. (2.17) and Eq. (2.18).

$$T_b = \frac{1}{SR}$$

T_b is time duration of the symbol and SR is symbol rate at which symbols are transmitted.

$$s_1 = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_0 t) \quad (2.17)$$

$$s_2 = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_0 t + \pi) = -\sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_0 t) \quad (2.18)$$

BPSK have one basis function $\phi_1(t)$ thus signal vector space is one-dimensional.

$$\phi_1(t) = \sqrt{\frac{2}{T_b}} \cos(2\pi f_0 t) \quad (2.19)$$

Basis function amplitude $\sqrt{\frac{2}{T_b}}$ ensures that basis is orthonormal. ~

Vector which describes first signal:

$$s_1 = s_{11} = \int_0^{T_b} s_1(t) \phi_1(t) dt = \sqrt{E_b}$$

Vector which describes second signal:

$$s_2 = s_{21} = \int_0^{T_b} s_2(t) \phi_1(t) dt = -\sqrt{E_b}$$

Signal space or constellation of BPSK modulation is shown in Fig. 2.11. Probability

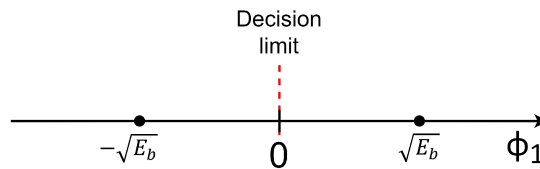


Figure 2.11: BPSK signal space

of transfer error for Additive White Gaussian Noise (AWGN) communication channel:

$$P_{e,BPSK} = \frac{1}{2} \text{erfc}\left(\sqrt{\frac{E_s}{N_0}}\right)$$

where E_s is symbol energy and N_0 is channel noise energy. BPSK bit to symbol mapping maps one bit of message into one symbol, that is number of bits $K_{BPSK} = 1$. In case of BPSK modulation symbol energy and symbol duration is same as bit energy and bit duration, $E_b = E_s, T_b = T_s$.

2.2.2. QPSK modulation

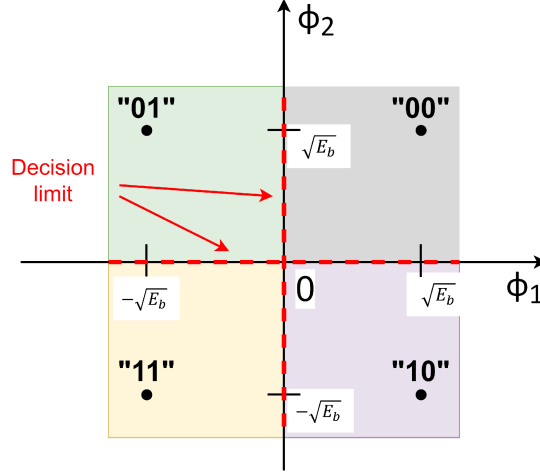


Figure 2.12: QPSK signal space

Coherent QPSK is modulation with $M = 4$, thus four signals are used, s_1, s_2, s_3 , and s_4 . Signals are given with the equations Eq. (2.20) - Eq. (2.23).

$$s_1 = \sqrt{\frac{2E_s}{T_s}} \cos(2\pi f_0 t + \frac{1\pi}{4}) \quad (2.20)$$

$$s_2 = \sqrt{\frac{2E_s}{T_s}} \cos(2\pi f_0 t + \frac{3\pi}{4}) \quad (2.21)$$

$$s_3 = \sqrt{\frac{2E_s}{T_s}} \cos(2\pi f_0 t + \frac{5\pi}{4}) \quad (2.22)$$

$$s_4 = \sqrt{\frac{2E_s}{T_s}} \cos(2\pi f_0 t + \frac{7\pi}{4}) \quad (2.23)$$

$$T_s = \frac{1}{SR}$$

T_s is time duration of the symbol and SR is symbol rate at which symbols are transmitted.

QPSK have two basis function $\phi_1(t)$ and $\phi_2(t)$ thus signal vector space is two-dimensional.

$$\phi_1(t) = \sqrt{\frac{2}{T_s}} \cos(2\pi f_0 t) \quad (2.24)$$

$$\phi_2(t) = \sqrt{\frac{2}{T_s}} \sin(2\pi f_0 t) \quad (2.25)$$

Signal space is shown in Fig. 2.12. Adjacent symbols have Hamming distance of 1, which means that they are different in 1 bit. Different constellation mappings are possible. It is beneficial for reducing transfer error probability that signals which are close have small Hamming distance.

BPSK and QPSK have same probability of transfer error, P_e . This is because QPSK modulation uses one bit of message to modulate In-Phase signal and second bit of message to modulate Quadrature signal. These bits, and these quadrature component modulations are independent. For that reason QPSK is basically two independent BPSK modulations thus P_e is the same for both modulations. QPSK modulation sends two bits of message at the same time and uses twice the energy of BPSK to maintain same Signal To Noise ratio.

Rectangular signals $u_{LP_I}(t)$ and $u_{LP_Q}(t)$ from Fig. 2.8 represent coordinates in signal space, signal $u_{BP}(t)$ is a QPSK modulated signal.

2.2.3. Offset QPSK (O-QPSK) modulation

QPSK modulation modulates In-Phase and Quadrature signals with message bits at the same instant of time. Symbol transition from one symbol to another passes through IQ plane origin, that is zero amplitude. This variation of amplitude produces unwanted effects in RF power amplifier and reduces amplifier efficiency.

O-QPSK is a variant of QPSK which modulates In-Phase and Quadrature signals with time delay, or offset, of half a symbol duration, $\frac{T_s}{2}$. This procedure eliminates symbol transitions through zero amplitude.

2.3. Nyquist filter and pulse shaping

Every real communication channel is band-limited. For example, FERSat expected channel bandwidth is 5 to 10 MHz. If we consider mixer input signals u_{LP_I} and u_{LP_Q} as rectangular pulses, for example as in Fig. 2.3 then magnitude spectrum of that signal has shape as in Fig. 2.4. This spectrum is unlimited with side lobes which

decays slowly (sinc function). Besides bandwidth issues with rectangular pulses, impulse response of the channel causes a transmitted symbol to be spread in time domain. This causes Inter Symbol Interference (ISI) where the previously transmitted symbols affects the currently received symbol, thus reducing tolerance for noise.

H. Nyquist came up with criterion for distortionless transmission which states that height (amplitude) of the middle of each signal element (symbol) should be undistorted. This means that each pulse $p(t)$ should be zero-valued in every sampling time instance T_s except for time instance $t = 0$, that is Nyquist criterion for zero-ISI:

$$p(t) = \begin{cases} 1; & t = 0 \\ 0; & t = nT_s \ (n \neq 0) \end{cases} \quad (2.26)$$

and

$$\sum_{n=-\infty}^{\infty} P\left(f - \frac{n}{T_s}\right) = T_s \quad (2.27)$$

A minimum bandwidth system has a rectangular shape from $-\frac{1}{2}T$ to $\frac{1}{2}T$. Where $\frac{1}{2}T$ is Nyquist frequency. Any filter that has excess bandwidth with odd symmetry around Nyquist frequency and even symmetry around zero frequency also satisfies zero-ISI criterion. Any such filter is called Nyquist filter.

Brick-wall filter or sync filter is ideal Nyquist filter. Frequency response and impulse response of the filter are illustrated in Fig. 2.13 and Fig. 2.14. Impulse response of the filter

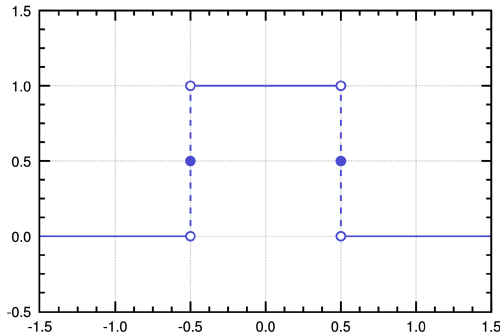


Figure 2.13: Rectangular function - frequency response of a brick-wall filter

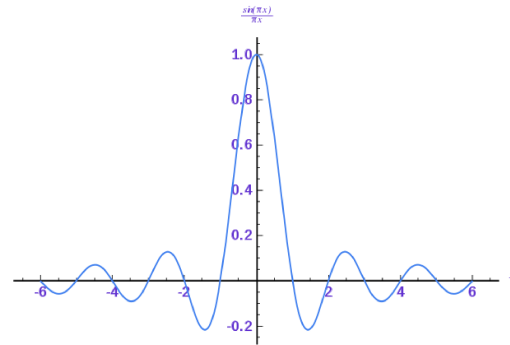


Figure 2.14: Graph of the normalized sinc function - impulse response of a brick wall filter

brick-wall filter is a Sinc function. It is slowly decaying signals which lasts from $t = -\infty$ to ∞ and because it decays slowly, time shifting and time-limiting such signal produces significant errors in spectrum. Therefore, it is practically unachievable. For

that reason we would like to use filter with different frequency response shape and fast decaying impulse response which could be approximated.

Raised Cosine filter is popular Nyquist filter. It's pulse has signal bandwidth $(\frac{1}{2}T_s)(1 + \beta)$. Frequency response of Raised Cosine filter [10] is given in Eq. (2.28) and plotted in Fig. 2.15.

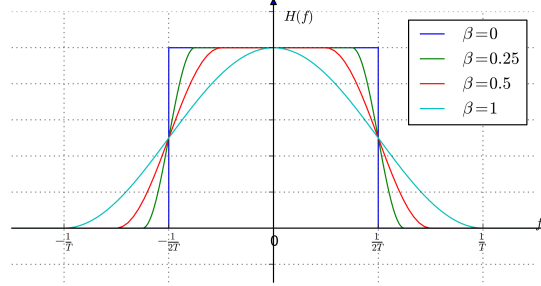


Figure 2.15: Frequency response of raised-cosine filter with various roll-off factors β [1]

$$H(f) = \begin{cases} 1, & |f| \leq \frac{1-\beta}{2T} \\ \frac{1}{2} \left[1 + \cos \left(\frac{\pi T}{\beta} \left[|f| - \frac{1-\beta}{2T} \right] \right) \right], & \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \\ 0, & \text{otherwise} \end{cases} \quad (2.28)$$

For each impulse at filter input filter output is one impulse response. This procedure is called pulse shaping. Using Raised Cosine (RC) shaped pulses, zero-ISI is achievable. Impulse response of RC filter decays quickly than Sinc pulses, and could be much better approximated. This type of filter is most widely in use as Nyquist transmit filter. Impulse response of Raised Cosine filter is given in Eq. (2.29) and plotted in Fig. 2.16 for various roll-off factors. There is an trade-off between spectral bandwidth and decaying slope.

$$h(t) = \begin{cases} \frac{\pi}{4T} \text{sinc} \left(\frac{t}{2\beta} \right), & t = \pm \frac{T}{2\beta} \\ \frac{1}{T} \text{sinc} \left(\frac{t}{T} \right) \frac{\cos \left(\frac{\pi \beta t}{T} \right)}{1 - \left(\frac{2\beta t}{T} \right)^2}, & \text{otherwise} \end{cases} \quad (2.29)$$

2.4. Peak to Average Power Ratio (PAPR)

Theoretical modulation use rectangular impulses as symbols. Perfect rectangular impulses have instantaneous level transitions and maximally sharp edges. Using theoretical modulation, modulated signal wave would have shape as in Fig. 2.10. If we

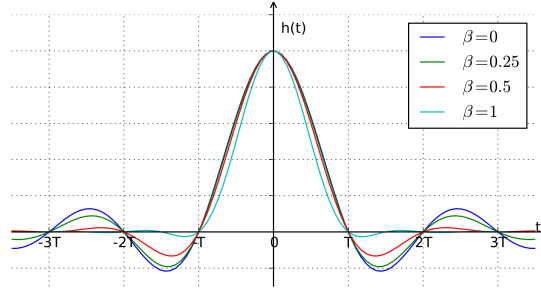


Figure 2.16: Impulse response of raised-cosine filter with various roll-off factors β [1]

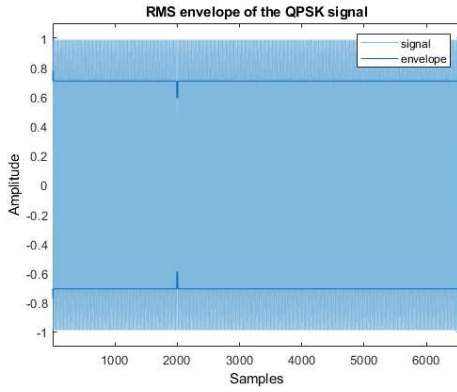


Figure 2.17: Envelope of QPSK modulation with ideal rectangular pulses

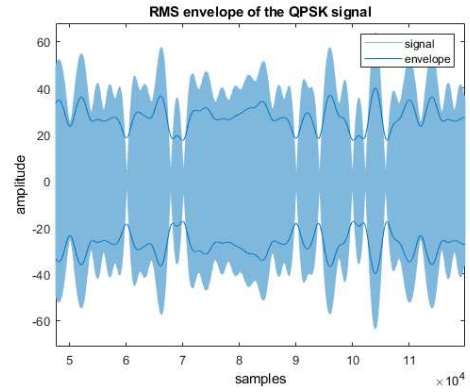


Figure 2.18: Envelope of QPSK modulation with RC shaped pulses

plot envelope of that signal Fig. 2.17, it can be seen that peak power to rms or average power is constant. When RC filtered pulses are used as symbols then symbol transition are not instantaneous. Sum of impulses produces envelope with peaks and drops. In that case PAPR is lower than ideal. Such transients behavior of signal envelope is a problem for linear RF power amplifier [11] because amplifier must have enough dynamic range to convey envelope peaks, but most of the time it operates at power lower than optimal. Envelope with peaks and drops produces unwanted nonlinear effects in the amplifier and reduces power efficiency of linear RF amplifier. Power amplifier efficiency is a trade-off for spectral efficiency of higher-order modulation formats. O-QPSK modulation format is variant of Q-PSK in which delay of $\frac{T_s}{2}$ between I and Q transitions produces modulated signal envelope which never drops to zero, that is signal transition in signal space never pass trough origin and envelope is more flat than compared with ordinary QPSK modulated envelope.

3. Proposed system architecture

3.1. System requirements

The goal is to have satellite communicating in Amateur-Radio X-band 10.450 - 10.500 MHz with dedicated channel bandwidth of 10 MHz. For that type of RF communication channel, International Telecommunication Union documentation was studied under FERSat project. Resulting maximum allowed spectral emissions are illustrated on the spectral mask given in Appendix A. This thesis is focused on more stringent spectral requirement of -40 dBc.

Analog part of transmitter is already defined. Mixer that will be used is Analog Devices's HMC1056LP4BE. This mixer receives analog IF input signals (In-phase and Quadrature component) in frequency range from DC up to 4 GHz. It's reasonable to have stringent requirements for digital processing part until transmitter prototype is field tested because of unknown parasitic parameters in analog part of transmitter, which can affect resulting unwanted spectral emissions.

System should transmit using one or all of modulation formats: BPSK, QPSK and O-QPSK. Line transmission rate need to be up to 16 Mbit per second.

3.2. Digital modulator architecture

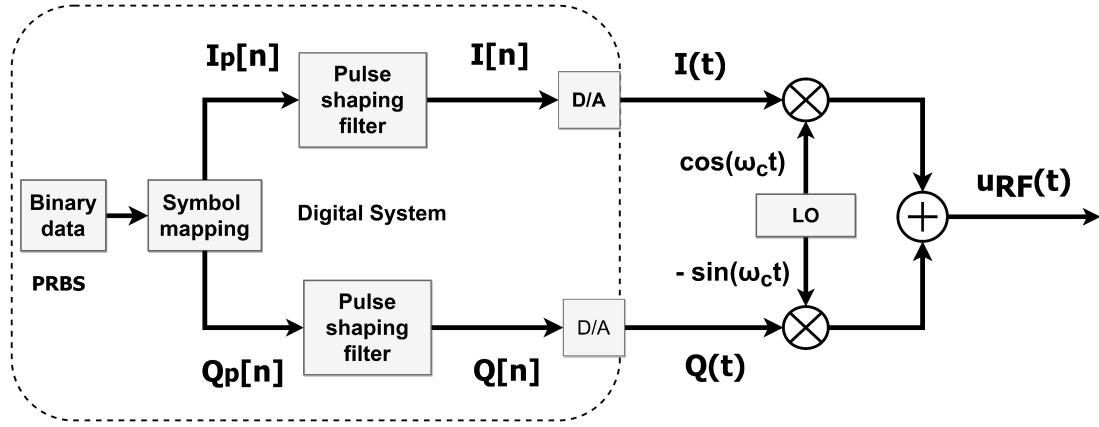


Figure 3.1: Block diagram of digital system for baseband signal processing

Fig. 3.1 illustrates proposed Direct Frequency Conversion mixer with complex mixing. Modulator is consisting of symbol mapper, transmit filters and digital to analog converters (DAC). Output of DACs are two continuous-time analog signals, $I(t)$ and $Q(t)$ which together makes complex envelope signal.

Pseudo-random Bit-Stream (PRBS) is not part of modulator. PRBS is used for testing and simulation purposes, which will be explained in the next chapter.

4. Matlab simulation

4.1. Pseudo-random Bit-Stream generator

PRBS generator is based on Linear Feedback Shift Register architecture. Basically it is a shift register with some flip-flop outputs (taps) connected to XOR gate. Output of the XOR gate is connected back to the input of the first flip-flop in the shift register [2]. Table in Fig. 4.2 contains listed polynomials for maximum length LFSR counters. For example, if PRBS-9 is used then potentions of x in the polynomial for the PRBS-9 defines which taps of LFSR are connected to XOR gate. Output of the PRBS is a sequence of random bits, that is sequence of the output from the last flip-flop in the LFSR. During first steps of writing simulation algorithm, it would be beneficial if

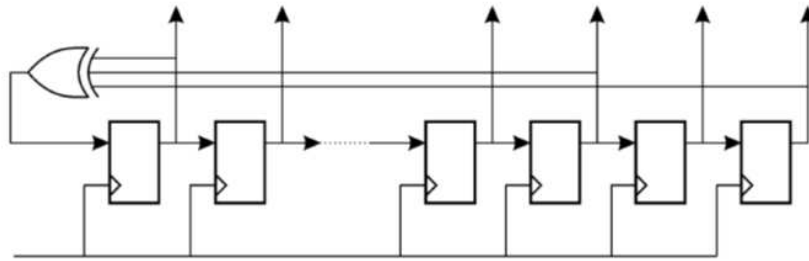


Figure 4.1: LFSR architecture [2]

resulting FFT spectrum of complex envelope would be clean and flat as possible. DFT expects that input sequence contains exactly k periods in the periodic signal, where $k \in \mathbb{N}$. To minimize spectral estimation errors due to non-integer number of periods, periodic sequence of data is used. PRBS generator of order n generates random bits with repetition cycle of $2^n - 1$ bits. In other word, PRBS- n is cyclic in $2^n - 1$ bits. This bits are mapped into symbols, thus period sequence of bits becomes periodic sequence of symbols. Because every symbol is represented with L samples, if FFT is calculated on a sequence of length $2^n - 1$ samples it will always be periodic sequence which will produce clean spectrum. This way it's easier to spot changes in spectrum

- Primitive polynomials with minimum # of XORs

Degree (n)	Polynomial
2,3,4,6,7,15,22	$x^n + x + 1$
5,11,21,29	$x^n + x^2 + 1$
8,19	$x^n + x^6 + x^5 + x + 1$
9	$x^n + x^4 + 1$
10,17,20,25,28	$x^n + x^3 + 1$
12	$x^n + x^7 + x^4 + x^3 + 1$
13,24	$x^n + x^4 + x^3 + x + 1$
14	$x^n + x^{12} + x^{11} + x + 1$
16	$x^n + x^5 + x^3 + x^2 + 1$
18	$x^n + x^7 + 1$
23	$x^n + x^5 + 1$
26,27	$x^n + x^8 + x^7 + x + 1$
30	$x^n + x^{16} + x^{15} + x + 1$

C. Stroud, Dept. of ECE, Auburn Univ. 10/04

Figure 4.2: LFSR polynomials [3]

due to simulated system parameter change. The concept of creating periodic sequence illustrated in Fig. 4.3. Depending on modulation type, more than one bit is packed into symbol and PRBS-n doesn't produce even-length sequence. If just one period of PRBS is used as sequence of data bits then last symbol would not be complete for modulation with order $M > 2$. This would break the periodicity of FFT sequence. Solution is to use at least two periods of PRBS as data sequence. Then, it is possible calculate FFT on a sequence of symbols with length equal to PRBS period. This sequence can be shifted left and right for any number of symbols. FFT sequence will be periodic, no matter the shift. This shift is used to avoid calculating FFT from zero-valued samples which exist at the beginning of simulation due to filter delay.

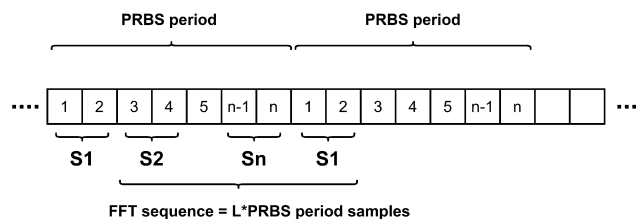


Figure 4.3: PRBS and sequence periodicity [1]

4.2. PRBS program implementation

PRBS generator is called in simulation by calling function
[symbol, nextseed] = f_prbs(seed, bits).

Function will determine order of PRBS-n from length of the parameter *seed*. Second paramter *bits* defines how many bits function will return as value of *symbol*. Next block of code will generate periodic pseudo-random message bits stored in variable *data* .

```

1 K = log2(M); % number of bits per symbol
2 % ...
3 seed = '100000000';
4 bits = length(seed);
5 prbs_period = int16(2^bits)-1; % LFSR RNG periodicity
6 sequence_period = lcm(int16(K), prbs_period); % symbols to
    repetition: symbols for cyclical sequence
7 m = 2*sequence_period; % need to have enough data for cyclic fft
    capture
8
9 for k=1:m
10 [symbol,seed] = f_prbs(seed,K);
11 switch symbol
12 %QPSK
13 case '00'
14 data = [data,0,0];
15 case '01'
16 data = [data,0,1];
17 case '10'
18 data = [data,1,0];
19 case '11'
20 data = [data,1,1];
21 %BPSK
22 case '0'
23 data = [data,0];
24 case '1'
25 data = [data,1];
26
27 otherwise
28 warning('Unsupported symbol type. ');
29 end
30 end

```

4.3. Symbol mapping

Symbol mapper is implemented inside function *f_mapsym_ms*. Functions is called from main simulation script *mpsk_sim_ms.m* within next block of code:

```
1 % #####
2 % SYMBOL MAPPING
3 % #####
4
5 [symbols, I, Q, SYMmapped] = f_mapsym_ms(modulation, data);
```

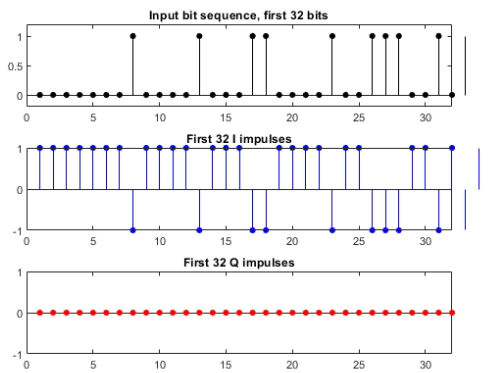


Figure 4.4: BPSK mapped symbol pulses

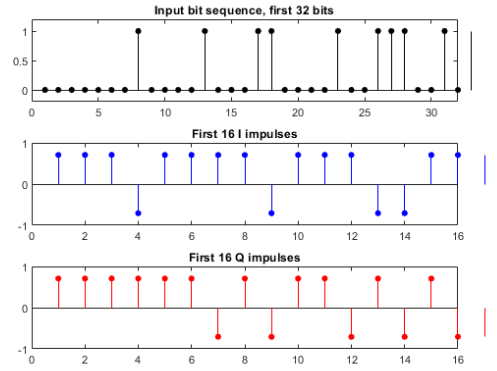


Figure 4.5: QPSK and O-QPSK mapped symbol pulses

Symbol mapping is implemented in function `[symbols, I, Q, SYMmapped] = f_mapsym_ms ( modulation, bitstream )`. Parameter *modulation* defines modulation format (BPSK, QPSK, O-QPSK). Function reads sets of K bits from input message (array of bits) and maps them into output vectors I and Q which contains impulses (one sample is one impulse) representing coordinates of symbols in signal space. Every k -th symbol $S(k)$ have coordinates in vector space $S(k) = (I(k), Q(k))$. Output array *SYMmapped* is array of k complex numbers where each complex number is representing one symbol $S(k)$ where $Re[S(k)] = I(k)$ and $Im[S(k)] = Q(k)$. Function also returns array *symbols* which is array of numbers (bits) with columns length of K numbers. Each column is representing K bits, that is, one symbol. Fig. 4.4 and Fig. 4.5 shows first 32 bits of message sequence and corresponding mapped symbol pulses. The next block of code is part of function *f_mapsym_ms.m*. Constellation lookup tables and I, Q mapping is listed.

```
1 %% Constellations (Look-up Tables)
2 % vector-row format!!!
3
```

```

4 % BPSK constellation
5 constell_BPSK = [ 1 , ... % 0
6 -1 ]; % 1
7
8 % QPSK constellation, Gray code, Hamming distance = 1
9 constell_QPSK = [ exp( 1i * pi/4 ), ... % 00 = 0d
10 exp( 1i * 3*pi/4 ), ... % 01 = 1d
11 exp( 1i * 7*pi/4 ), ... % 10 = 3d
12 exp( 1i * 5*pi/4 ) ]; % 11 = 2d
13
14 % Other constellations
15
16 % ...
17 % My own reshape and mapping
18 SYMmapped = zeros(1,len/K); % memory allocation for mapped
    symbols
19 symbols = zeros(K,len/K); % symbols, binary
20 symbol = zeros(1,K); % one symbol, binary
21 addr = zeros(1,len/K); % memory allocation for symbol
    addressing
22 col = 0;
23 idx = 0;
24 for ii = 1: K : len
25     col = col +1;
26     idx = idx+1;
27     for kk = 0 : K-1
28         row = kk+1;
29         symbols(row,col) = bitstream(ii+kk); % All symbols
30         symbol(kk+1) = bitstream(ii+kk); % One symbol
31         addr(idx) = addr(idx) + ((2^(K-1-kk)) * bitstream(ii+kk));
32         SYMmapped(idx) = constell (addr(idx)+1);
33     end
34 end
35
36 % ...
37 I = real(SYMmapped);
38 Q = imag(SYMmapped);

```

4.4. Raised Cosine filter

Raised Cosine filter is a family of Nyquist filters with frequency response shape of raised cosine function with flat top. It is used for zero-ISI transmission.

Filter is designed using function `h = f_raisedcosine_ms( alpha, span, sps, shape, implementation )` as in the next code block:

```
1 % h_rc is a impulse response coefficients of FIR nyquist filter
2 % h = f_raisedcosine_ms( alpha, span, sps, shape, implementation )
3
4 h_rc = f_raisedcosine_ms( rolloff, RCF_span, L, RCF_shape, 'matlab' )
5 %
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
26
```

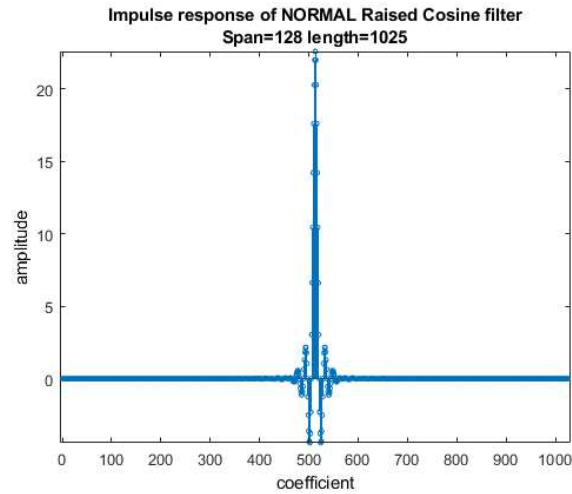


Figure 4.6: Impulse response samples of the designed RC filter

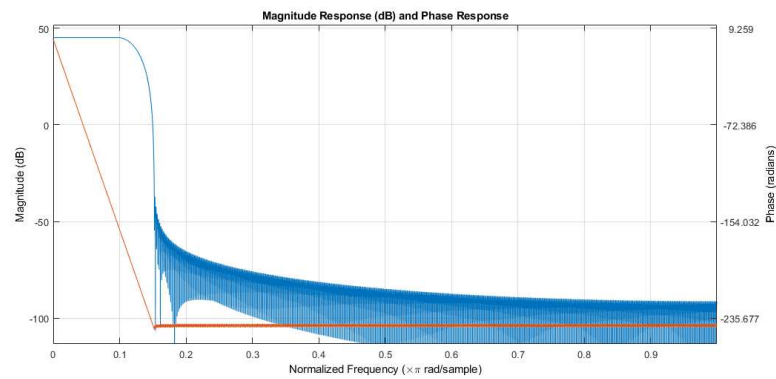


Figure 4.7: Magnitude and Phase response of the designed RC filter

4.5. Pulse shaping

Pulse shaping is a procedure in which pulse are 'shaped' by the Nyquist filter, that is RC filter. Each pulse is a impulse response of the filter caused by input stimulus. In order to produce correct impulse response, input stimulus cannot be mapped impulses but the need to be upsampled for a factor of 2, at least. Basically, filter need enough samples (discrete time) to produce response and enough samples containing information which describe each pulse. Upsampling by factor L is done by inserting $L - 1$ zeros between original impulses. Next block of code produces upsampled I and Q signals (I_up and Q_up):

```
1 % ### UPSAMPLE
2 I_up = upsample(I, L);
3 Q_up = upsample(Q, L);
```

Pulse shaping is a filtering process and it is done using Matlab function filter by calling:

```

1 % Signals are concatenated with zeros at the end to allow filter
  response
2 % to decay
3 I_up = [I_up, zeros(1,L*(RCF_span/2))];
4 Q_up = [Q_up, zeros(1,L*(RCF_span/2))];
5
6 I_f_state = zeros(1,length(h_rc)-1); % I filter state variable
7 Q_f_state = zeros(1,length(h_rc)-1); % Q filter state variable
8
9 [I_filt, I_f_state] = filter(h_rc, 1, I_up, I_f_state); % In-
  phase branch filtration
10 [Q_filt, Q_f_state] = filter(h_rc, 1, Q_up, Q_f_state); %
  Qadrature branch filtration

```

Impulse response to each impulse is affected by impulse response of previous $\frac{span}{2}$ impulse responses. For this reason filter state is kept in a _state variable. At the beginning state variable is empty (all state are zero). Output signal delay is half the filter span. This is why the signal is concatenated with zeros at the end, to allow the filter to settle down after all data symbols are filtered.

Resulting upsampled signals for BPSK and QPSK are shown in Fig. 4.8 and Fig. 4.9. When using O-QPSK modulation then phase (time) delay of $\frac{T_s}{2}$ is inserted between I

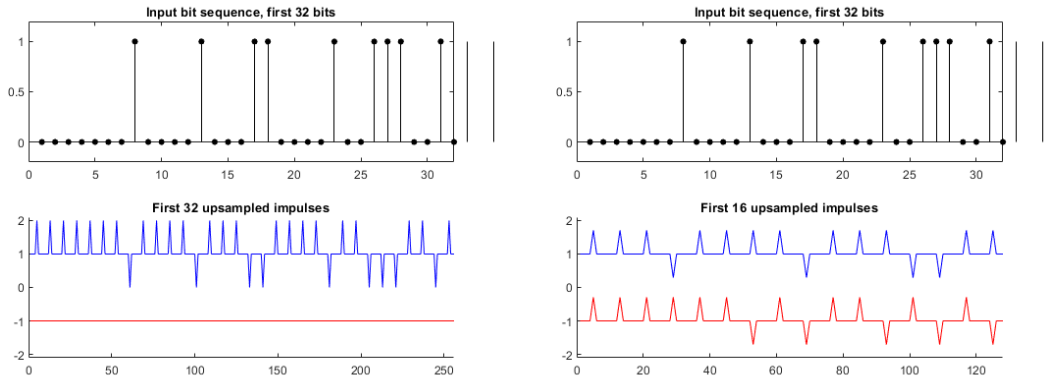


Figure 4.8: BPSK upsampled symbol pulses **Figure 4.9:** QPSK upsampled symbol pulses

and Q impulses. This is shown in Fig. 4.10. Effect of upsampling factor L is signal shift in discrete frequency band. Increasing L pushes signal spectrum closer to 0 frequency. It is easier to filter signal which is more 'packed' around low frequencies, thus it looks that it is better to have larger upsampling factor L . There is a trade off in increased data rate and need for increased processing power.

Another question is delay of half the symbol time T_s when O-QPSK modulation is used. If symbol is sampled with even number of samples, for example 8, it is not pos-

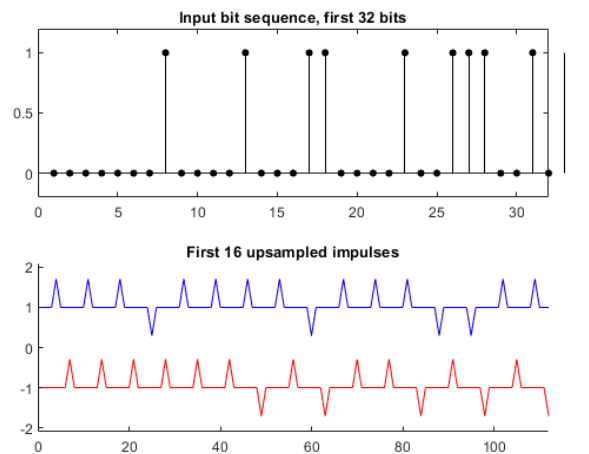


Figure 4.10: O-QPSK upsampled symbol pulses

sible to delay impulses for a exactly half the symbol time. It can be delayed for a $\text{floor}[\frac{L-1}{2}]$ samples. In case of $L = 8$ that would be 3 samples. In case of even number of samples per symbol, L, to make correct delay of impulses interpolation would be needed. To avoid interpolation, thus reducing processing power, it is favorable to chose odd number of samples per symbol. Resulting shaped impulses are shown in

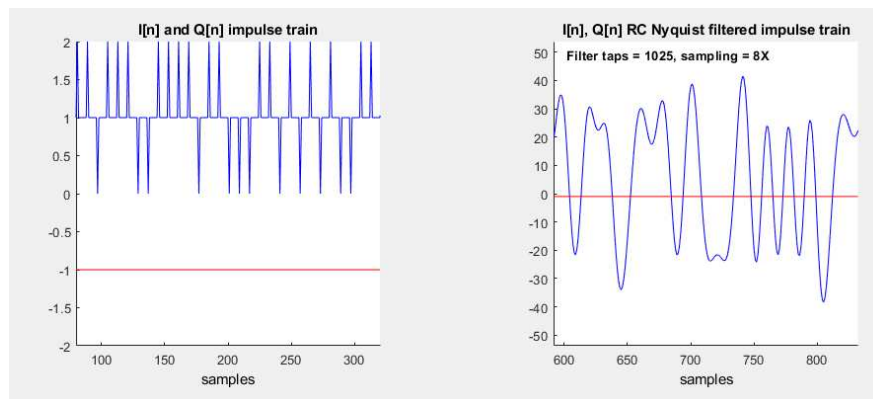


Figure 4.11: BPSK filtered and shaped I and Q impulses

Fig. 4.11, Fig. 4.12 and Fig. 4.13.

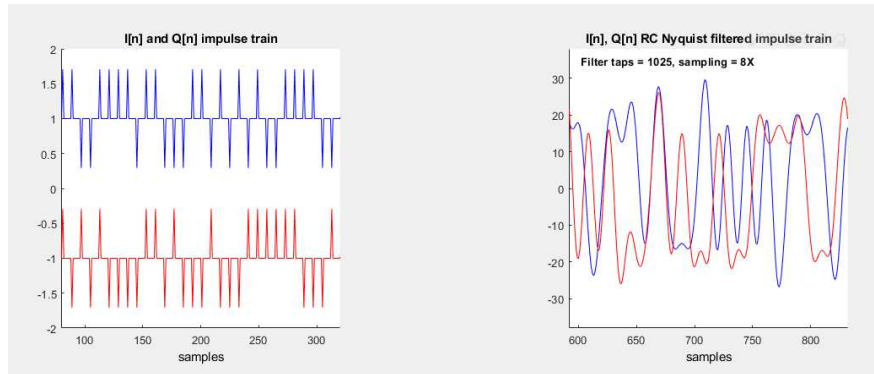


Figure 4.12: QPSK filtered and shaped I and Q impulses

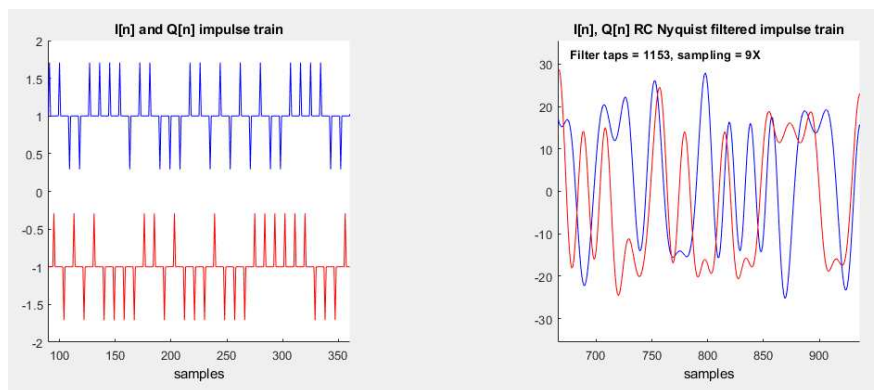


Figure 4.13: O-QPSK filtered and shaped I and Q impulses

5. Simulation results

5.1. Long RC filter span results

Using filter span 128 symbols and sampling factor $L=8$ next signal transition in vector space and resulting spectra are simulated. Shown in next sections.

5.1.1. BPSK results

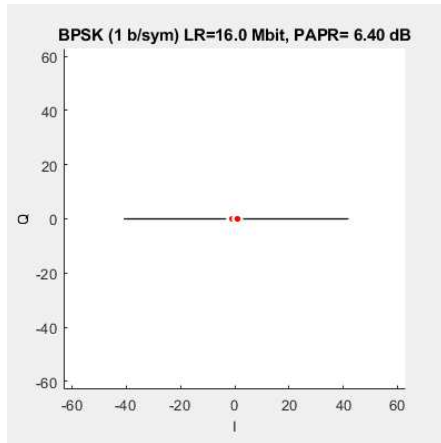


Figure 5.1: BPSK signal transition in signal space

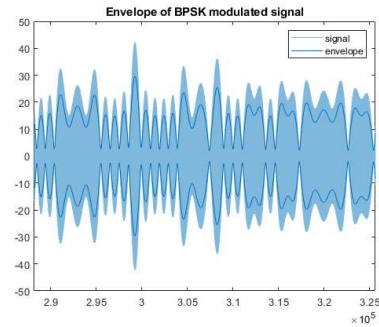


Figure 5.2: Envelope of the BPSK signal

Fig. 5.1 shows that BPSK modulation have one-dimensional signal space. Envelope drops to zero in each symbol state change. Bandwidth of BPSK signal from Fig. 5.3 is wider than symbol rate bandwidth and conforms to rule for RC filter which is $SR(1 + \alpha) = 16 \times 10^6 \times (1 + 0.22) = 19.52 \text{ MHz}$

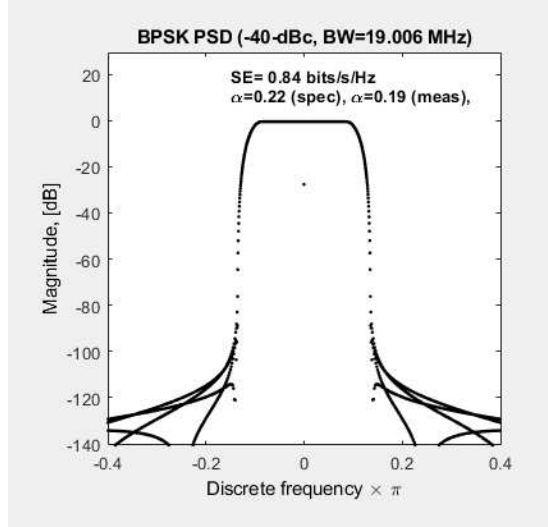


Figure 5.3: BPSK signal spectrum

5.1.2. QPSK results

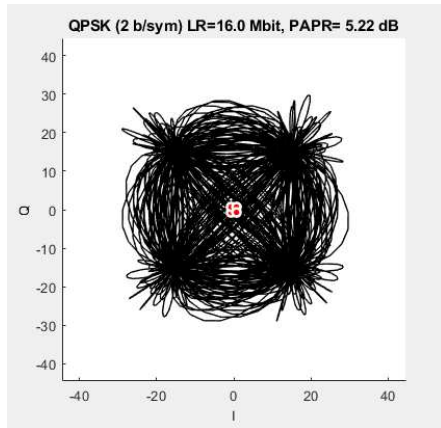


Figure 5.4: QPSK signal transition in signal space

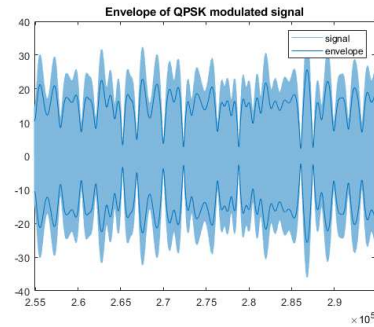


Figure 5.5: Envelope of the QPSK signal

Fig. 5.4 shows that QPSK modulation have two-dimensional signal space. Envelope drops to zero, but not as in BPSK modulation. QPSK modulation produce signal with better Peak to Average Power Ratio. This modulation conforms the channel bandwidth requirement.

5.1.3. O-QPSK results

Fig. 5.7 shows that O-QPSK modulation have two-dimensional signal space. Envelope does not drop to zero amplitude and PAPR is beter than PAPR from QPSK modulation. Bandwidth is the sam as for QPSK modulation.

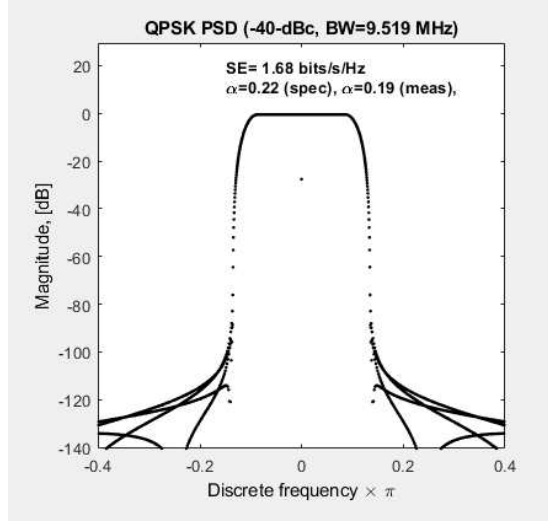


Figure 5.6: QPSK signal spectrum

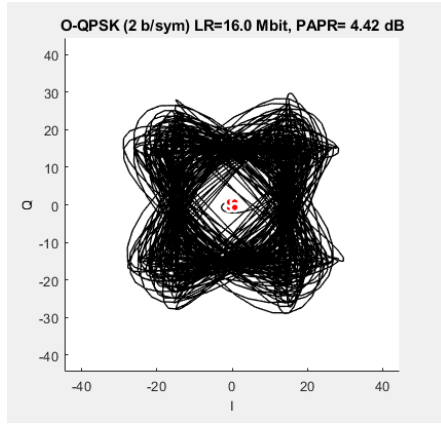


Figure 5.7: O-QPSK signal transition in signal space

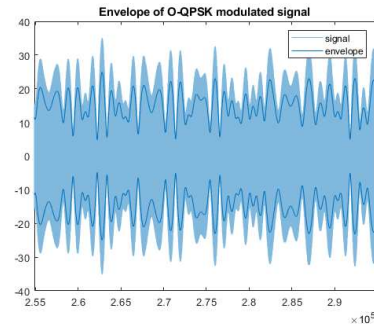


Figure 5.8: Envelope of the O-QPSK signal

5.2. Shortening the RC filter span

When filter have smaller number of taps it's have greater error. Result is spectrum which stop band is not attenuated as much as with the longer filter. This simulation is concentrated on BPSK and QPSK modulations.

Using the same rolloff factor $\alpha = 0.22$ as in previous simulations, but shortening the filter span to 16 symbols and increasing the sampling factor to $L=16$ result with the filter response, as in Fig. 5.10.

Changing the rolloff factor from $\alpha=0.22$ to $\alpha=0.30$ widens passband bandwidth but stop-band attenuation is increased. Magnitude response of the RC filter with 257 tap, 16symbol span, 16 samples per symbol and rolloff=0.30 is shown in Fig. 5.11. Resulting spectra for BPSK and QPSK are shown in Fig. 5.12 and Fig. 5.13. From the

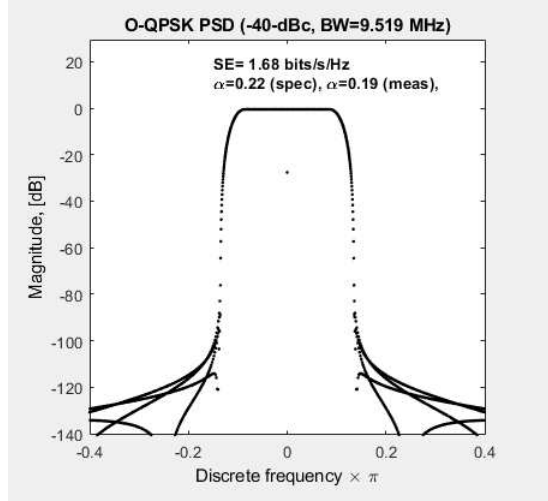


Figure 5.9: O-QPSK signal spectrum

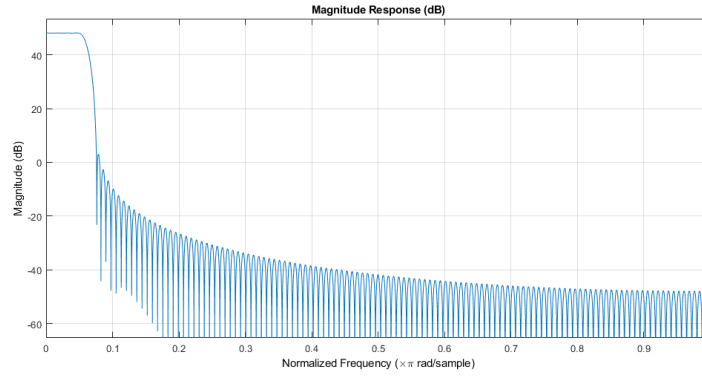


Figure 5.10: Magnitude response of 257 tap RC filter, span 16 symbols, $L=16$, $\alpha=0.22$

simulation results it is clear that RC filter rolloff factor will make a trade off between excess bandwidth in passband and attenuation in stop-band. BPSK modulated signal violates channel bandwidth, thus using BPSK maximum attainable line rate is up to 8 Mbps. With this filter design, unwanted spectra is still under -40 dBc.

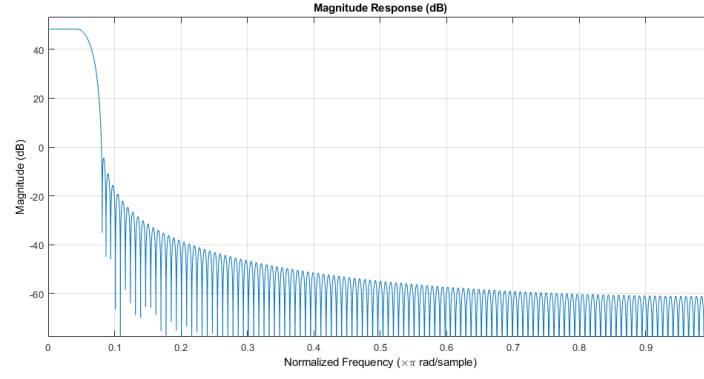


Figure 5.11: Magnitude response of 257 tap RC filter, span 16 symbols, $L=16$, $\alpha=0.30$

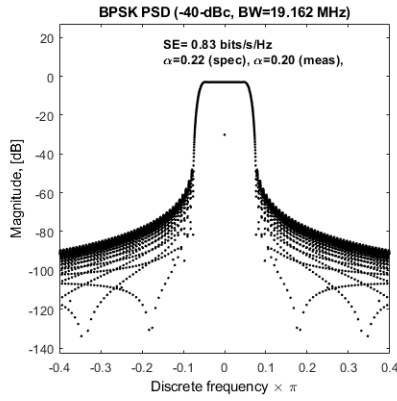


Figure 5.12: BPSK spectrum using 257 tap RC filter

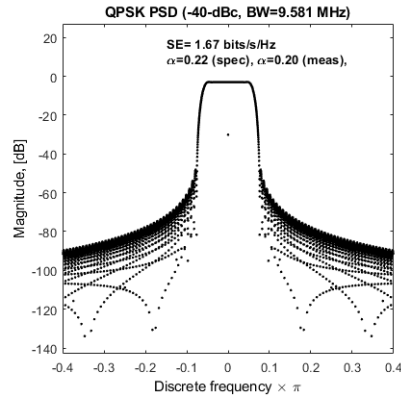


Figure 5.13: QPSK spectrum using 257 tap RC filter

5.3. Further shortening the RC filter span

Using the same rolloff factor $\alpha = 0.22$ as in previous simulations, but shortening the filter span to 16 symbols and *decreasing the sampling factor to $L=8$* result with the spectra as in Fig. 5.14 and Fig. 5.15. From the simulation it stands out that the impact on stop band attenuation and first side lobe magnitude is greater by the filter span than by the sampling factor L . Minimum passband width is determined by the line rate, that is, symbol rate. If the only criterion would be first side lobe in spectrum below -40 dBc than filter span 16 symbols with 5 time oversampling would be marginally enough. But for better results, oversampling by a factor of 8 is needed. Thus resulting filter span using $L=8$ is 129 taps.

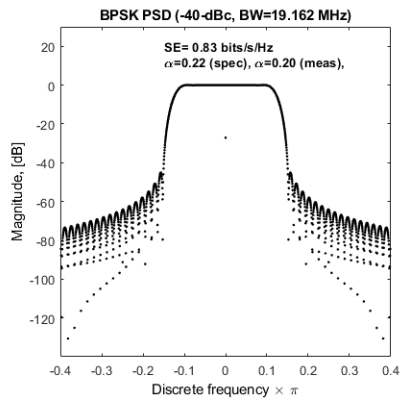


Figure 5.14: BPSK spectrum using 129 tap RC filter

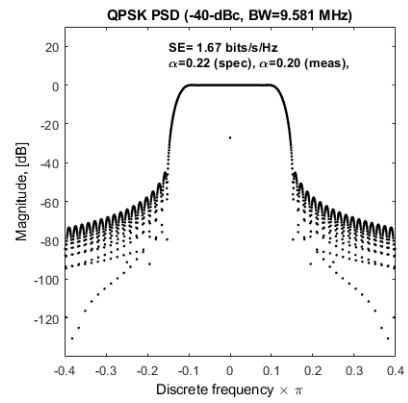


Figure 5.15: QPSK spectrum using 129 tap RC filter

6. Conclusion

This Thesis focus was on analysis digital signal processing steps for a baseband signals and how complex envelope signal is generated.

It is simulated how Nyquist filter affects complex envelope signal bandwidth. What are effects of rolloff factor, that is excess bandwidth and how filter impulse response is affecting spectral content of complex envelope. Trough multiple runs of simulation it is come to conclusion that to satisfy 10 MHz channel bandwidth using BPSK modulation maximum line rate is up to 8 MHz and spectral efficiency of BPSK is 0.84 bits/s/Hz. QPSK modulation has the spectral efficiency twice the BPSK modulation. To satisfy requirement of -40 dBc filter with at least 16 symbol span is needed. Oversampling factor from $L=5$ up is producing satisfying results. Thus resulting minimum filter span is 81 taps.

Whole processing system need to have two such filters. System output in case of $L=5$ oversampling would be 25 Msps for each of I and Q branches.

Next steps would be to consider decimating filter for lowering data throughput and analysis of digital to analog converter amplitude resolution effects.

BIBLIOGRAPHY

- [1] User:Krishnavedala. (2011) Frequency response of raised-cosine filter with various roll-off factors. [Online; accessed 18-September-2020]. [Online]. Available: https://commons.wikimedia.org/wiki/File:Raised-cosine_filter.svg
- [2] M. Vučić. (2020) Bilješke s predavanja iz predmeta obrada signala u komunikacijama. [Online; accessed 18-September-2020]. [Online]. Available: <https://www.fer.unizg.hr/predmet/osuk/predavanja>
- [3] C. E. Stroud. (2011) Linear feedback shift registers. [Online; accessed 18-September-2020]. [Online]. Available: <http://www.eng.auburn.edu/~strouce/class/elec6250/LFSRs.pdf>
- [4] H. Riebeek and R. Simmon, “Catalog of earth satellite orbits,” Sep 2009. [Online]. Available: <https://earthobservatory.nasa.gov/features/OrbitsCatalog>
- [5] “Pregled formula iz digitalne obradbe signala,” 2019. [Online]. Available: <http://www.fer.hr/predmet/dos/ispiti>
- [6] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.
- [7] D. Petrinović, *Ekvivalencija vremenski kontinuiranih i diskretnih signala i sustava*, Fakultet elektrotehnike i računarstva, 2008.
- [8] Wikipedia contributors, “Passband — Wikipedia, the free encyclopedia,” <https://en.wikipedia.org/w/index.php?title=Passband&oldid=972761523>, 2020, [Online; accessed 20-September-2020].
- [9] —, “Baseband — Wikipedia, the free encyclopedia,” <https://en.wikipedia.org/w/index.php?title=Baseband&oldid=972377414>, 2020, [Online; accessed 20-September-2020].

- [10] —, “Raised-cosine filter — Wikipedia, the free encyclopedia,” https://en.wikipedia.org/w/index.php?title=Raised-cosine_filter&oldid=948688776, 2020, [Online; accessed 18-September-2020].
- [11] E. McCune, *Practical Digital Wireless Signals*, ser. The Cambridge RF and Microwave Engineering Series. Cambridge University Press, 2010.

Baseband Signal processing Chain for a Satellite Digital Transmitter

Abstract

Baseband signal processing chain for a Satellite Digital Transmitter is Master Thesis at the faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. Digital processing procedures and steps for the baseband signal were analyzed. System for processing complex envelope signal was simulated. Minimal parameters of the digital processing system which satisfies assigned parameters of signal quality were specified. Simulations were made for digital modulation formats BPSK, QPSK and O-QPSK.

Keywords: BPSK, QPSK, O-QPSK, DSP, COMPLEX, MIXING, MODULATOR

Lanac za obradu signala u osnovnom pojasu u satelitskom digitalnom predajniku

Sažetak

Lanac za obradu signala u osnovnom pojasu u satelitskom digitalnom predajniku je Diplomski rad na Fakultetu elektrotehnike i računarstva u Zagrebu. Analizirani su postupci digitalne obrade signala u osnovnom pojasu. Simuliran je sustav za obradu signala kompleksne ovojnice i određeni su minimalni parametri digitalnog sustava koji zadovoljavaju zadane kriterije kvalitete signala. Simulacije su provedene za tipove digitalnih modulacija BPSK, QPSK i O-QPSK.

Ključne riječi: BPSK, QPSK, O-QPSK, DSP, MODULATOR, KOMPLEKSNO, MIJEŠANJE

Appendix A

Spectral emission limits for Radio-Amater Band in X-Band radio frequencies

UTU Emission designator for 10 MHz bandwidth channel, digital phase modulation with single carrier (single channel communication) is 10M0 G1D.

Emission power

f [GHz]

Assigned band

10 MHz

BN

0 dBc **P = 2 W** **0 dBsd (PSD)**

MASK from SM.1541-6, p.58

-23 dBc**

Occupied bandwidth *

-10 dBc, RBW = 1 kHz

spurious domain

OOB domain

Amateur-satellite band

10.4825 **10.490** **10.495** **10.50** **10.507** **10.5175** **10.520**

-2.5BN **-1.2BN** **-0.5BN** **f_c** **+0.5BN** **+1.2BN** **+2.25BN** **+2.5BN**

- (31 + 10 log P) = -34 dBc (RBW = kHz)

- (38 + 10 log P) = -41 dBc, RBW = 10 kHz

- (43 + 10 log P) = -46 dBc or -13 dBm RBW = 4 kHz #

consultation needed

-163 dBW # = -133 dBm

-28 dBm with antenna angle 20° at 600 km altitude

(**) 99% power occupied bandwidth permitted by a particular emission mask can be determined by calculating the frequency difference between the 23 dB attenuation levels

(***) Ask J. Vuković - How is necessary bandwidth defined, depending on Link Budget or something else?

(***) Assigned bandwidth $\neq$ Necessary bandwidth $\neq$ Occupied bandwidth

(#) ITU-R SM.329-10, Table 9, p. 31

(##) ITU-R SM.329-10, Table 10, p. 34

38

Appendix B

Source code

B.1. Source code of main Matlab simulation script

```
1 % File: MPSK_SIM_MS.m
2 %
3 % Master Thesis
4 %
5 % Work in progres
6 % Simulating an 4-PSK (QPSK) modulation
7 %
8 % Version:      0.200909-0
9 % Date:         September 9, 2020
10 % Student:      Mario Simunic
11 %   email:      mario.simunic@fer.hr
12 %   email1:     mario.simunic@gmail.com
13
14 % close all
15 % clear all
16 clc
17 colordef white
18
19 % #####
20 % SIMUATION PARAMETERS
21 % #####
22
23 use_prbs = 1;
24 % use_prbs = 0;
25
26 % -----
27 % MODULATION
28 % -----
```

```

29 modulation = 'BPSK';
30 M=2; % modulation order
31 % modulation = 'QPSK';
32 % M = 4; % modulation order
33 o_qpsk = 0;
34
35 % modulation = 'O-QPSK';
36 % M = 4; % modulation order
37 % o_qpsk = 1;
38
39
40 K = log2(M); % number of bits per symbol
41
42 % -----
43 % TRANSMIT AND SAMPLE PARAMETERS
44 % -----
45 LR = 8*10^6; % LR = line rate, Transfer speed, bits/
second
46 SR = LR/K; % SR = symbol rate, symbols/second
47
48 Tsymbol = 1/SR; % Symbol period, from link speed, br (
bits per second)
49
50
51 % -----
52 % TRANSMIT FILTER (Nyquist) PARAMETERS - Spectral Raised Cosine
filter
53 % -----
54 % RCF_shape = 'sqrt'; % Square-Root Raised Cosine filter
55 RCF_shape = 'normal'; % Square-Root Raised Cosine filter
56 RCF_span =16; % Filter impulse response span
57 rolloff = 0.22; % beta factor, spectrum rolloff
58 % ##### SYMBOL SAMPLING
59 L = 3; % symbol upsampling factor, samples per
symbol
60
61 % ===== print parameters
62 starttime = char(datetime('now','Format','yyyy-MM-dd''_''HHmmss'));
63 log_filename = ['M-PSK_sim_',starttime,'.log'];
64 diary(log_filename); % diary on
65
66 fprintf("\nM-PSK SIMULATION by Mario Simunic\n");
67

```

```

68 fprintf("\n\nLog filename:\t\t\t M-PSK_sim_%s.log\n", starttime);
69
70 fprintf("\n\nSimulation parameters:\n");
71
72 fprintf("\nModulation:\t\t %s (M = %d)\n", modulation, M);
73
74 fprintf("\nBit rate:\t\t\t\t\t %d,\n", LR);
75 fprintf("\nSymbol rate:\t\t\t\t\t %d,\n", SR);
76 fprintf("\nSamples per symbol:\t\t\t\t %d,\n", L);
77 fprintf("\nSample rate:\t\t\t\t\t %d,\n", L*SR);
78 fprintf("\nTransmit filter shape:\t\t\t %s,\n", RCF_shape);
79 fprintf("\nTransmit filter span:\t\t\t %d symbols,\n", RCF_span);
80 fprintf("\nTransmit filter alpha:\t %1.2f\n", rolloff);
81
82 % -----
83
84
85
86
87
88
89 % #####
90 % BINARY DATA - pseudorandom bit vector
91 % #####
92 % This binary data vector represents a synchronization header (ie.
   Barker
93 % code) plus payload data with protective coding. All together
   scrambled
94 % for uniform distribution of zeros and ones. Uniform distribution
   of zeros
95 % and ones (without contonous blocks) helps to estimate carrier
96 % signal at receiver.
97
98 fprintf('\n>BEGIN ===== f_prbs.m
   =====>> \n\n');
99
100 if use_prbs == 1
101 %
   ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
102 % Generate Ciclic repetitive pseudo-random sequence - for a nice FFT
   :-)

```

```

103 %
      .....
104
105 seed = '100000000';      % if it has nine digits, its PRBS9, if 11
      its PRBS11.
106 % seed = '10000';        % 5 bits
107 % It can also do 15, 21, 31, but those will be too long for the
      memory
108 bits = length(seed);
109 prbs_period = int16(2^bits)-1;      % LFSR RNG periodicity
110 sequence_period = lcm(int16(K), prbs_period); % symbols to
      repetition: symbols for cyclical sequence
111 m = 2*sequence_period; % need to have enough data for cyclic fft
      capture
112
113 fprintf(' Generating %d bits of random data \n', m);
114 data = [];
115 for k=1:m
116     [symbol,seed] = f_prbs(seed,K);
117     switch symbol
118         %QPSK
119         case '00'
120             data = [data,0,0];
121         case '01'
122             data = [data,0,1];
123         case '10'
124             data = [data,1,0];
125         case '11'
126             data = [data,1,1];
127         %BPSK
128         case '0'
129             data = [data,0];
130         case '1'
131             data = [data,1];
132
133         otherwise
134             warning('Unsupported symbol type.');
```

```

140
141 end % end if
142
143 % Print data sequence
144 fprintf('\n data = \n');
145 for k=0:ceil(length(data)/40)-1
146     for(n=1:40)
147         if k*40+n <= length(data)
148             fprintf(" %d", data(k*40+n));
149         end
150     end
151     fprintf("\n");
152 end
153 fprintf('\n\n');
154 fprintf('>==== f_prbs_ms.m =====
        END\n\n');
155
156 % -----
157
158
159
160
161
162
163 % #####
164 % SYMBOL MAPPING
165 % #####
166
167 [symbols, I, Q, SYMmapped] = f_mappsym_ms(modulation, data);
168
169 fig = figure('Name', ['I and Q impulses of ', modulation, ' modulation
        ']);
170 subplot(3,1,1);
171 stem(data, 'k', 'MarkerSize', 4, 'MarkerFaceColor', 'auto');
172 axis([0, length(data)+1, -0.2, 1.2]);
173 title(['Input bit sequence']);
174
175 subplot(3,1,2);
176 stem(I, 'b', 'MarkerSize', 4, 'MarkerFaceColor', 'auto');
177 axis([0, length(I)+1, -1.2, 1.2]);
178 title(['In-phase (I) impulses']);
179
180 subplot(3,1,3);

```

```

181     stem(Q, 'r', 'MarkerSize', 4, 'MarkerFaceColor', 'auto');
182     title(['Quadrature (Q) impulses']);
183     axis([0, length(Q)+1, -1.2, 1.2]);
184
185 fprintf("\nFigure %d: I and Q impulses\n", fig.Number);
186
187
188
189 if length(data) > 32
190     N_disp = 32;
191 else N_disp = length (data);
192 end
193
194 N_IQ_disp = N_disp/log2(M);
195
196 fig = figure('Name', ['First ', num2str(N_disp), ' I and Q impulses of
    ', modulation, ' modulation']);
197 subplot(3,1,1);
198 stem(data, 'k', 'MarkerSize', 4, 'MarkerFaceColor', 'auto');
199 axis([0, N_disp, -0.2, 1.2]);
200 title(['Input bit sequence, first ', num2str(N_disp), ' bits']);
201
202 subplot(3,1,2);
203 stem(I, 'b', 'MarkerSize', 4, 'MarkerFaceColor', 'auto');
204 axis([0, N_IQ_disp, -1, 1]);
205 title(['First ', num2str(N_IQ_disp), ' I impulses']);
206
207 subplot(3,1,3);
208 stem(Q, 'r', 'MarkerSize', 4, 'MarkerFaceColor', 'auto');
209 title(['First ', num2str(N_IQ_disp), ' Q impulses']);
210 axis([0, N_IQ_disp, -1, 1]);
211
212 fprintf("\nFigure %d: First %d I and Q impulses\n", fig.Number,
    N_disp);
213
214 %-----
215
216
217
218
219
220 % #####
221 % SYMBOL FILTERING

```

```

222 % #####
223
224 % ### UPSAMPLE
225 I_up = upsample(I, L);
226 Q_up = upsample(Q, L);
227
228 if o_qpsk == 1 % ##### Using O-QPSK
229     Q_up = circshift(Q_up, floor(L/2)); % Equivalent to time
        delay
230                                     % of half a symbol
231 end
232
233 fig = figure('Name', ['Upsampled I and Q signals']);
234     subplot(2,1,1);
235     stem(data, 'k', 'MarkerSize', 4, 'MarkerFaceColor', 'auto');
236     axis([0, length(data), -0.2, 1.2]);
237     title(['Input bit sequence']);
238
239     subplot(2,1,2);
240     hold on
241     plot(1 + I_up, 'b');
242     plot(-1 + Q_up, 'r');
243     hold off
244     axis([0, length(I_up), -2.1, 2.1]);
245     title(['Upsampled impulses']);
246
247 fprintf("\nFigure %d: %s input bit stream, I and Q signals upsampled
        with %d samples per symbol.\n", fig.Number, modulation, L);
248
249 if length(data) > 32
250     N_disp = 32;
251 else
252     N_disp = length(data);
253 end
254 N_IQ_disp = N_disp/K;
255
256
257 fig = figure('Name', [modulation, ' I and Q upsampled signals']);
258     subplot(2,1,1);
259     stem(data, 'k', 'MarkerSize', 4, 'MarkerFaceColor', 'auto');
260     axis([0, N_disp, -0.2, 1.2]);
261     title(['Input bit sequence, first ', num2str(N_disp), ' bits']);
262

```

```

263     subplot(2,1,2);
264     hold on
265     plot(1 + [zeros(1,floor(L/2)),I_up(1:end-floor(L/2))],'b');
        % [zeros(1,L),I_up] inserted zeros are for plot alignment of
        impulses with stem(data)
266     plot(-1 + [zeros(1,floor(L/2)),Q_up(1:end-floor(L/2))],'r');
267     hold off
268     axis([0, N_IQ_disp*L, -2.1, 2.1]);
269     title(['First ',num2str(N_IQ_disp),' upsampled impulses']);
270
271 fprintf("\nFigure %d: %s input bit stream, I and Q signals upsampled
        with %d samples per symbol.\n", fig.Number, modulation, L);
272
273
274
275 % ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
276 % NYQUIST FILTER - design using L samples per symbol
277 % ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
278
279 % simulation parametrs - at the begining
280
281 % #### Ideal Nyquist filter is brick-wall filter or rectangular
        spectral
282 % pulse. Problems with ideal filter - Impulse response is decaying
        slowly,
283 % it is non-causal, can't be realized, etc.
284 % Paramters of Nyquist filter ----
285 % We will be using Raised Cosine and/or Square-Root Raised Cosine
        filter
286
287 % h_rc is a impulse response coefficients of FIR nyquist filter
288 h_rc = f_raisedcosine_ms( rolloff, RCF_span, L, RCF_shape, 'matlab')
        ;
289 %
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
290 % Filter Gain, [dB]
291 FG = 40-4.139; % To get PSD at 0 dB
292 % rcosdesign and f_raisedcosine_ms returns filter coefficients with
293 % energy = 1, ie. h = b2/sqrt(sum(b2.^2));
294 % It would be beneficial to use filter with heigher gain, eg. +20db,
        but
295 % not to exceed dynamic range of variable (register).

```

```

296 G = 10^(FG/20);
297     % In case of using R bit wide register and fixed
    point
298     % arithmetic. Number in 2's complement format 1.(R
    -1) can
299     % have maximum positive value  $1 - 2^{-(R-1)}$ .
300     % To avoid saturation use gain  $K < 1.0$ 
301     % For 13 bit register and number in fractional 1.12
    format, maximum postive
302     % number is 0.99975586
303
304 % impulse reponse with gain
305 h_rc = G * h_rc;
306 % -----
307
308 % plot RC filter
309
310 fig = figure('Name','f_raisedcosine_ms: Designed Nyquist filter');
311     stem(h_rc,'MarkerSize',3,'markerFaceColor','none');
312     axis([-5, length(h_rc)+6, -0.1 0.3]);
313     title(['Impulse response of ',upper(RCF_shape),' Raised Cosine
    filter'], ['Span=',num2str(RCF_span),' length=',num2str(length(
    h_rc))]);
314     xlabel('coefficient');
315     ylabel('amplitude');
316 fprintf("\nFigure %d: Impulse response of %s Raised Cosine filter,
    span = %d symbols with %d samples per symbol.\n",...
317     fig.Number, RCF_shape, RCF_span, L);
318
319 % Magnitude frequency characteristics
320 H_rc = 20*log10((1/length(h_rc))*abs(fftshift(fft(h_rc))));
321 f_step = 2/(length(H_rc)-1);           %  $2\pi/\pi = 2$ ;
322 f_axis = -1:f_step:1;                  % f_axis =  $-1\pi/\pi:f\_step/\pi$ 
    :1 $\pi/\pi$ ;
323
324 fig = figure('Name','Magnitude frequency response of nyquist filter'
    );
325     plot(f_axis,H_rc);
326     axis([-1.05 1.05 max(H_rc)-120 max(H_rc)+10]);
327     title(['Magnitude frequency response of ',upper(RCF_shape),'
    Raised Cosine filter']);
328     xlabel('discrete frequency \times \pi');
329     ylabel('magnitude, [dB]');

```

```

330 fprintf("\nFigure %d: Frequency response of %s Raised Cosine filter,
      span = %d symbols with %d samples per symbol.\n",...
331       fig.Number, RCF_shape, RCF_span, L);
332 % -----
333
334
335
336 % ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
337 % PULSE SHAPING (filtration) using Raised Cosine Nyquist filter
338 % ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
339
340 [D, state_D] = f_set_fir_delay( floor(L/2)); % used to prepare the
      FIR for delaying one half of the symbol
341 [BI, state_BI] = f_set_fir_delay( 1); % used for FIR for syncing the
      response with the inpyut impulses
342 [BQ, state_BQ] = f_set_fir_delay( 1); % used for FIR for delaying
      one half of the symbol
343
344 % Signals are concatenated with zeros at the end to allow filter
      response
345 % to decay
346 I_up = [I_up, zeros(1,L*(RCF_span/2))];
347 Q_up = [Q_up, zeros(1,L*(RCF_span/2))];
348
349 I_f_state = zeros(1,length(h_rc)-1); % I filter state variable
350 Q_f_state = zeros(1,length(h_rc)-1); % Q filter state variable
351
352 [I_filt, I_f_state] = filter(h_rc, 1, I_up, I_f_state); % In-
      phase branch filtration
353 [Q_filt, Q_f_state] = filter(h_rc, 1, Q_up, Q_f_state); %
      Qadrature branch filtration
354
355
356 fig = figure('Name',[modulation,' - filtered I and Q signals']);
357 hold on
358 plot(0.5 + I_filt,'b');
359 plot(-0.5 + Q_filt,'r');
360 hold off
361 axis([0, length(I_filt), -(max(I_filt)+0.5)*1.1, (max(I_filt)
      +0.5)*1.1]);
362 title(['Filtered impulses']);
363 xlabel('sample');
364 legend('I signal','Q signal','location','east');

```

```

365
366 fprintf("\nFigure %d: %s quadrature signals filtered using %s Raised
      Cosine filter, span = %d symbols with %d samples per symbol.\n
      ", ...
367       fig.Number, modulation, RCF_shape, RCF_span, L);
368 % -----
369
370
371
372
373
374
375
376 % #####
377 % SPECTRAL ANALYSIS
378 % #####
379
380 % ### FFT
381 ss_period = L*sequence_period;
382 ssp = L*(RCF_span); % we will make FFT somewhere in
      signal. This beginng is SSP
383 Nfft = double(ss_period); % ss_period == speriod
384 % -----
385
386 I_fft = 1/Nfft * fftshift(fft( I_filt( ssp : ssp+Nfft ), Nfft)); %
      FFT of filtered I signal
387 Q_fft = 1/Nfft * fftshift(fft( Q_filt( ssp : ssp+Nfft ), Nfft)); %
      FFT of filtered Q signal
388
389 % P_fft = 10*log10(abs( I_fft.^2 + Q_fft.^2)); % PSD - here lies
      a problem - sidelobes in spectrum
390 P_fft = 10*log10(abs( I_fft .* conj(I_fft) + Q_fft .* conj(Q_fft)));
      % PSD - here lies a problem
391 I_fft = 20*log10(abs(I_fft));
392
393 fstep = 2/Nfft;
394 f_axis = -1:fstep:1-fstep;
395
396 %find max magnitude
397 mag0dBc = max(P_fft); % Carrier magnitude or for M
      -PSK it's max magnitude in passband
398 % mag0dBc = max(I_fft);
399 mag_limit_dBc = -40; % 60 dB below carrier

```

```

400 bw_limit = mag0dBc + mag_limit_dBc;      % Bandwidth limit - I choose
      to set at 60 dB below carrier (-60 dBc)
401 for k = Nfft:-1:floor(Nfft/2)           % search index where FFT is
      gretaer than magnitude given in bw_limit
402 %     if P_fft(k) > bw_limit
403     if P_fft(k) > bw_limit
404         leftk = k;
405         break;
406     end
407 end
408
409 % #### bandwidth
410 bw_desc = 2*f_axis(leftk);               % descrete domain bandwidth
411 fsample = SR*L;                          % symbol sample rate
412 bw_cont = bw_desc * fsample/2;           % continous domain bandwidth
413
414 % Efeffective bandwidth = SPECTRAL EFFICIENCY
415 % bweff = SR*K/BW_cont;                  % LR = SR*K
416 bweff = LR/bw_cont;                      % spectral efficiency, [bits/s/Hz]
417
418
419 % #### PAPR
420 E = I_filt.^2 + Q_filt.^2;
421 P = sum(E)/length(E);
422 Pavg = 10*log10(P);
423
424 Ppeak = 10*log10(max(E));
425
426 PAPR = Ppeak - Pavg; %It is Ppeak/Pavg, but this is logarithmic
427 %-----
428
429
430
431
432
433
434 % #####
435 % RESULTS PLOTING
436 % #####
437 % plot labels
438 textlabel2 = sprintf('Filter taps = %d, sampling = %dX', RCF_span*L
      +1, L);
439

```

```

440 textlabel3 = sprintf('%s (%d b/sym) LR=%3.1f Mbit, PAPR=%5.2f dB',
    modulation, K, LR*1.0e-6, PAPR);
441
442 bwlabel1 = sprintf(' PSD (%d-dBc, BW=%5.3f MHz)', mag_limit_dBc,
    bw_cont*1.0e-6 );
443
444 bwlabel2 = sprintf('SE=%5.2f bits/s/Hz', bw_eff );
445 bwlabel2_pos = [-0.15, mag0dBc+20];
446
447 bwlabel3 = sprintf('\alpha=%4.2f (spec), \alpha=%4.2f (meas), ',
    rolloff, bw_cont/SR-1 );
448 bwlabel3_pos = [-0.15, mag0dBc+12];
449
450
451 % plot all
452 fig = figure('Name',[modulation,' simulation results'], 'Position'
    ,[50 50 1100 900]);
453     subplot(2,2,1)
454     hold on
455     plot(1 + I_up,'b');
456     plot(-1 + Q_up,'r');
457     hold off
458     axis([10*L 40*L -2 2]);
459     axis square;
460     title(['I[n] and Q[n] impulse train'],'FontSize', 11);
461     xlabel('samples');
462
463     subplot(2,2,2)
464     hold on
465     plot(1 + I_filt,'b');
466     plot(-1 + Q_filt,'r');
467     hold off
468     axis([L*(10+RCF_span/2) L*(40+RCF_span/2) -max(I_filt)*1.2 max(
    I_filt)*1.2]);
469     axis square;
470     % Kako je prof. Babic dobio vecu (istu) amplitudu na izlazu
    filtra??? -
471     % Pogledaj energiju filtra - skaliranje koeficijenata
472     title(['I[n], Q[n] RC Nyquist filtered impulse train'],'FontSize'
    ', 11);
473     xlabel('samples');
474     text(L*(10+RCF_span/2)+10,max(I_filt)*1.1,textlabel2,'FontSize',
    10,'FontWeight','bold');

```

```

475
476     ax = max(I_filt);
477     subplot(2,2,3)
478     hold on;
479     plot( I_filt, Q_filt, 'k-', 'LineWidth', 1);
480     plot( I, Q, 'wo', 'LineWidth', 2, 'MarkerFaceColor', 'r');
481     hold off
482     axis([-1.5*ax 1.5*ax -1.5*ax 1.5*ax]);
483     xlabel('I', 'FontSize', 10 );
484     ylabel('Q', 'FontSize', 10 );
485     title(textlabel3, 'FontSize', 11 );
486     axis square;
487
488     subplot(2,2,4)
489     plot(f_axis, P_fft, 'k. ');
490     axis([-0.4 0.4 max(P_fft)-140 max(P_fft)+30]);
491     axis square;
492     title([modulation, bwlabel1], 'FontSize', 11);
493     xlabel('Discrete frequency \times \pi');
494     ylabel('Magnitude, [dB]');
495     text(bwlabel2_pos(1), bwlabel2_pos(2), bwlabel2, 'FontSize', 10, '
FontWeight', 'bold' );
496     text(bwlabel3_pos(1), bwlabel3_pos(2), bwlabel3, 'FontSize', 10, '
FontWeight', 'bold' );
497
498     fprintf("\nFigure %d: %s simulation results.\n", ...
499         fig.Number, modulation);
500
501     fprintf("\n\nRESULTS:\n");
502     fprintf('%3d-dBc BW = %2.3f MHz, SE = %2.2f bits/s/Hz\n', floor(
        mag_limit_dBc), bw_cont*1.0e-6, bweff );
503     fprintf('\alpha=%4.2f (spec), \alpha=%4.2f (meas), PAPR=%5.2f dB\n
        ', rolloff, bw_cont/SR-1, PAPR );    % PAPR - to explain !!!
504
505     % Just spectrum figure
506     figure('Name', [modulation, ' - resulting spectrum'])
507     plot(f_axis, P_fft, 'k. ');
508     axis([-0.4 0.4 max(P_fft)-140 max(P_fft)+30]);
509     axis square;
510     title([modulation, bwlabel1], 'FontSize', 11);
511     xlabel('Discrete frequency \times \pi');
512     ylabel('Magnitude, [dB]');

```

```

513     text(bwlabel2_pos(1), bwlabel2_pos(2), bwlabel2,'FontSize', 10,'
        FontWeight', 'bold' );
514     text(bwlabel3_pos(1), bwlabel3_pos(2), bwlabel3,'FontSize', 10,'
        FontWeight', 'bold' );
515
516
517 % Envelope of the signal
518 %
519 fs_RF = 100*SR*L;
520 Tc_RF = 1/fs_RF;
521 fc = fs_RF/20;
522
523 % take piece of filtered pulses and resample the to RF sampling rate
524 % (simulation)
525
526 I_filt_block = I_filt(1: L*RCF_span*5);
527 Q_filt_block = Q_filt(1: L*RCF_span*5);
528 I_rf = resample(I_filt_block,100,1);
529 Q_rf = resample(Q_filt_block,100,1);
530
531 t_step = 1/fs_RF;
532 tc = 0:t_step:(length(I_rf)-1)*t_step;
533
534 uc_sine = -sin(2*pi*fc*tc); % cosine signal, frequency fc, duration
    = I_rf
535 uc_cosine = cos(2*pi*fc*tc);
536
537 I_t = uc_cosine .* I_rf;
538 Q_t = uc_sine .* Q_rf;
539
540 u_rf = I_t + Q_t;
541
542 figure();
543 envelope(u_rf,20*L,'rms');
544 title(['Envelope of ',modulation,' modulated signal']);
545
546
547
548 fprintf("\n\n===== END
    =====");
549 fprintf("\n\n");
550 diary off

```

B.2. Source code of PRBS generator

```
1 function [symbol, nextseed] = f_prbs(seed,bits)
2 % seed in string form, the length of the seed determines n
3 % output is a bit for bits=1, but you can get any number of bits out
  with
4 % bits
5
6 n = numel(seed);
7 switch n
8     case 1
9         symbol = '';
10        for k=1:bits
11            if rand > 0.5
12                symbol = strcat(symbol, '1');
13            else
14                symbol = strcat(symbol, '0' );
15            end
16        end
17        nextseed = '0';
18        return;
19    case 2
20        tap1 = 2;
21        tap2 = 1;
22    case 3
23        tap1 = 3;
24        tap2 = 1;
25    case 4
26        tap1 = 4;
27        tap2 = 1;
28    case 5
29        tap1 = 5;
30        tap2 = 2;
31    case 6
32        tap1 = 6;
33        tap2 = 1;
34    case 7
35        tap1 = 6;
36        tap2 = 7;
37    case 9
38        tap1 = 5; % 4
39        tap2 = 9;
40    case 10
```

```

41         tap1 = 10;
42         tap2 = 3;
43     case 11
44         tap1 = 9;
45         tap2 = 11;
46     case 15
47         tap1 = 14;
48         tap2 = 15;
49     case 17
50         tap1 = 17;
51         tap2 = 3;
52     case 18
53         tap1 = 18;
54         tap2 = 7;
55     case 20
56         tap1 = 20;
57         tap2 = 3;
58     case 21
59         tap1 = 21;
60         tap2 = 2;
61     case 22
62         tap1 = 22;
63         tap2 = 1;
64     case 23
65         tap1 = 18;
66         tap2 = 23;
67     case 25
68         tap1 = 3;
69         tap2 = 25;
70     case 28
71         tap1 = 28;
72         tap2 = 3;
73     case 29
74         tap1 = 29;
75         tap2 = 2;
76     case 31
77         tap1 = 28;
78         tap2 = 31;
79     otherwise
80         fprintf('prbs unrecognized... exit\n');
81         return;
82 end
83 verb = 0;

```

```

84 for k=1:n
85     rg(k) = str2num(seed(k));
86     if verb
87         fprintf('%d', rg(k));
88     end
89 end
90 if verb
91     fprintf('\n');
92 end
93 symbol = '';
94 for k=1:bits
95     if (rg(tap1) == 1) && (rg(tap2) == 0)
96         dummy = 1;
97     elseif (rg(tap1) == 0) && (rg(tap2) == 1)
98         dummy = 1;
99     else
100         dummy = 0;
101     end
102     for ii=n:-1:2
103         rg(ii) = rg(ii-1);
104     end
105     rg(1) = dummy;
106     dum = num2str(rg(n));
107     symbol = strcat( symbol, dum );
108 end
109 nextseed = '';
110 for k=1:n
111     nextseed = strcat( nextseed, num2str(rg(k)) );
112 end

```

B.3. Source code of symbol mapper

```
1 % F_MAPPSYM_MS() will mapp input data vector of bits (bitstream)
   to
2 % M - point (M-ary) constellation based on Gray code.
3 % modulation - type of modulation. Supported types are 'BPSK', '
   QPSK'.
4 % bitstream - input vector (length power-of-2) of integers or
   boolean
5 % representing bits.
6 %
7 % f_mappsym_ms(modulation, bitstream) outputs binary symbols in
   form of
8 % K x N matrix, where K is number of bits in constellation symbol
   and N
9 % is a length(bitstream)/K constellation points.
10 %
11 % [symbols, I, Q, SYMmapped] = f_mappsym_ms(modulation, bitstream)
12 % outputs binary symbols, mapped constellation points in vector of
13 % In-Phase componenets, I, and quadrature components, Q, and the
   vector
14 % of complex points, SYMmapped, consisting of I + 1i*Q points.
15 %
16 %
17 % Version:      0.200909-0
18 % Date:         September 9, 2020.
19 % Student:      Mario Simunic
20 % email:        mario.simunic@fer.hr
21 % email1:       mario.simunic@gmail.com
22
23 function [symbols, I, Q, SYMmapped] = f_mappsym_ms ( modulation,
   bitstream )
24 version = '0.2009090-0'; % mod(2006050,11) = 2
25 filename = 'f_mappsym_ms.m';
26 file = dir(filename);
27
28 fprintf('\n\n>BEGIN ===== f_mappsym_ms.m ===== ver. %s
   =====>\n',version);
29 if isempty(file) == 1
30     warning('File f_mappsym_ms.m is not in the current folder.');
```

```
31 else
32 %     fprintf("%s, file version: %s, last change: %s\n", file.name,
   version, file.date);
```

```

33 end
34
35
36 %% Constellations (Look-up Tables)
37 % vector-row format!!!
38
39 % BPSK constellation
40 constell_BPSK = [ 1 , ... % 0
41                 -1 ]; % 1
42
43 % QPSK constellation, Gray code, Hamming distance = 1
44 constell_QPSK = [ exp( 1i * pi/4 ), ... % 00 = 0d
45                 exp( 1i * 3*pi/4 ), ... % 01 = 1d
46                 exp( 1i * 7*pi/4 ), ... % 10 = 3d
47                 exp( 1i * 5*pi/4 ) ]; % 11 = 2d
48
49 % Other constellations
50
51
52
53 %% Symbol Mapping
54 % M = modulation order
55 % K = number of bits per symbol
56
57 % convert type just for the logic operations (if, switch, etc.)
58 bitstream = boolean(bitstream);
59
60 switch modulation
61
62     case 'BPSK'
63         constell = constell_BPSK;
64         M = 2;
65         K = log2(M);
66
67     case 'QPSK'
68         constell = constell_QPSK;
69         M = 4;
70         K = log2(M);
71
72     case 'O-QPSK'
73         constell = constell_QPSK;
74         M = 4;
75         K = log2(M);

```

```

76
77     otherwise
78         error('Unsupported modulation type.');
```

79 end

```

80
81 % Check in bistream length is correct for the modulation type.
82 % len need to be divisible by the K
83 len = length (bistream);
84 if (mod(len,K) ~= 0)
85     error('Unsupported length of input data vector.\n');
```

86 end

```

87
88 fprintf("\n\nUsing modulation: %s ( M = %d)\n", modulation, M);
89
90
91 % symbol mapping using matrix operation and block data
92 % syms = reshape(data,K,L/K);      % Vector-row convert to K-
elemnt columns.
93                                     % Each column is a symbol.
94 % addr = (2.^(K-1:-1:0))*syms); % Symbols converted to look-up
table
95                                     % address.
96 %
97 % Ss = constell (addr+1);          % Symbol stream
98
99
100 % Symbol mapping using elemnt operation and elemnt-by-element
operation
101 %     Parameters for testing
102 %     data = [1 0 1 1 0 0 0 1];
103 %     K = 2;
104 %     L = 8;
105
106 % My own reshape and mapping
107 SYMmapped = zeros(1,len/K);          % memory allocation for mapped
symbols
108 symbols = zeros(K,len/K);            % symbols, binary
109 symbol = zeros(1,K);                 % one symbol, binary
110 addr = zeros(1,len/K);               % memory allocation for symbol
addressing
111 col = 0;
112 idx = 0;
113 for ii = 1: K : len
```

```

114     col = col +1;
115     idx = idx+1;
116     for kk = 0 : K-1
117         row = kk+1;
118         symbols(row,col) = bitstream(ii+kk);    % All symbols
119
120         symbol(kk+1) = bitstream(ii+kk);        % One symbol
121
122         addr(idx) = addr(idx) + ((2^(K-1-kk)) * bitstream(ii+kk));
123
124         SYMmapped(idx) = constell (addr(idx)+1);
125
126     end
127 end
128
129
130
131 %% plot constellation points
132 fig = figure('Name','f_mappsym_ms: Symbols mapped to constellation')
133     ;
134 hold on;
135     % unit circle definition
136     r = 1;
137     theta = 0:2*pi/50:2*pi;
138     x = r*cos(theta);
139     y = r*sin(theta);
140     plot(x,y,'b. ');    % Unit circle
141
142 % constellation points from lookup table
143 plot(real(constell),imag(constell),'ro', 'MarkerFaceColor','r');
144 axis([-1.5 1.5 -1.5 1.5]);
145 % axis equal
146 axis square
147 grid on
148
149 % constellation point labels
150 if(0)
151     for k = 1:length(constell)
152         text(real(constell(k))-0.2, imag(constell(k))+0.2, dec2bin(k
153             -1, K),...
154             'Color','red','FontSize',13);
155     end
156 end

```

```

155
156 % plot symbol point labels with number of counted symbols from data
    stream
157 if (1)
158     % count number of occurrence of each symbol
159     PointCnt = zeros(1,M);
160     for k = 1:length(addr)
161         PointCnt(addr(k)+1) = PointCnt(addr(k)+1) + 1;
162     end
163     % For each constellation point print number of occurrences
164     for k = 1:length(constell)
165         msg = [dec2bin((k-1),K), ' \times ', int2str(PointCnt(k))];
166         pos_x = real(constell(k))-0.2;
167         pos_y = imag(constell(k))+0.2;
168         text(pos_x, pos_y, msg, 'Color', 'red', 'FontSize', 13);
169     end
170 end
171
172 hold off;
173 xlabel('In-Phase'); ylabel('Quadrature');
174 title([modulation, ' constellation with number of occurrences']);
175 % ax = gca;
176 % ax.XAxisLocation = 'origin';
177 % ax.YAxisLocation = 'origin';
178 % set(ax, 'FontSize', 12);
179 fprintf("\nFigure %d: Symbols mapped to constellation\n", fig.Number
    );
180
181 I = real(SYMmapped);
182 Q = imag(SYMmapped);
183
184 % fprintf('\n Input data = \n');
185 % disp(bitstream);
186 %
187 % fprintf('\n Symbols mapped (each column is one symbol):\n');
188 % disp(symbols);
189 %
190 % fprintf('\n Symbols mapped to constellation = \n');
191 % disp(SYMmapped);
192
193 fprintf('\n>===== f_mappsym_ms.m
    ===== END\n\n');
194

```

```
195 end
196
197 %EOF
```

B.4. Source code of filter design

```
1 %F_RAISEDCOSINE_MS Spectral Raised Cosine Filter design.
2 %   h = f_raisedcosine_ms(alpha, span, sps, shape, implementation)
   returns spectral raised
3 %   cosine FIR filter coefficients, h, with rolloff factor alpha. The
   filter
4 %   is truncated to SPAN symbols and each symbol is represented by
   sps
5 %   samples. F_RAISEDCOSINE_MS designs a symetric filter. Therefore,
   the
6 %   filter order which is span*sps, must be even. The filter energy
   is 1.
7 %   mode will define how FIR filter will be designed, 'mtlb' - using
   matlab
8 %   rcosdesing() function, or, 'ms' - using my own implementation.
   This is
9 %   for testing purposes and results should be identical.
10 %
11 %   alpha is rolloff factor,
12 %   span is filter impulse response duration (symbols),
13 %   sps is number of samples per symbol,
14 %   shape defines which spectral shape of the filter will be
   designed. When
15 %       shape is set to 'normal' function will return spectral
   Raised
16 %       Cosine filter coefficients. When shape is set to 'sqrt'
   function
17 %       will return spectral Square-Root Raised Cosine filter
   coefficients,
18 %   implementation is a string value 'matlab' or 'ms' where 'ms'
   represents
19 %       My implementation.
20 %
21 % Version:      0.200909-0
22 % Date:         Spetember 9, 2020
23 % Student:      Mario Simunic
24 %   email:      mario.simunic@fer.hr
25 %   email1:     mario.simunic@gmail.com
26 %
27 function h = f_raisedcosine_ms( alpha, span, sps, shape,
   implementation )
28
```

```

29 version = "0.200909-0";
30 filename = 'f_raisedcosine_ms.m';
31 file = dir(filename);
32
33 fprintf('\n\n>BEGIN ===== f_raisedcosine_ms.m ==== ver. %s
    =====>\n', version);
34 if isempty(file) == 1
35     warning('File f_mappsym_ms.m is not in the current folder.');
```

```

36 else
37     fprintf(" %s, file version: %s, last change: %s\n", file.name,
    version, file.date);
38 end
39
40 % Debugging
41 % fprintf('\n Using implementation = %s\n', implementation);
42
43
44 %% Take One - Using Matlab function rcosdesign()
45 % =====
46 %
47 % Spectral Rised Cosine filter design using rcosdesign()
48 %
49 % b = rcosdesign(beta,span,sps) returns the coefficients, b, that
50 % correspond to a square-root raised cosine FIR filter with
    rolloff
51 % factor specified by beta. The filter is truncated to span
    symbols,
52 % and each symbol period contains sps samples. The order of the
    filter,
53 % sps*span, must be even. The FILTER ENERGY is 1.
54 %
55 % b = rcosdesign(beta,span,sps,shape) returns a square-root raised
    cosine
56 % filter when you set shape to 'sqrt' and a normal raised cosine
    FIR
57 % filter when you set shape to 'normal'.
58
59 if (1 == strcmp(implementation,'matlab'))
60     fprintf('\n Using ''matlab'' implementation for computing
    coeffcients.\n\n');
```

```

61     % Raised cosine filter
62     b1 = rcosdesign(alpha, span, sps, shape); % design impulse
    response

```

```

63
64     h = b1;
65 end
66
67 % =====
68
69
70
71 %% Take Two - My implementation of raised cosine filter impulse
    response
72 % =====
73 %
74 if (1 == strcmp(implementation,'ms'))
75
76     fprintf('\n Using ''ms'' implementation for computing
    coeffcients.\n\n');
77
78     FIRord = span * sps;          %
79     % Check if filter order is even
80     if ( mod(FIRord,2) ~= 0) %
81         error('    Filter order of Raised Cosine filter must be
    even!');
82     end
83
84     FIRlength = FIRord + 1;
85     b2 = zeros(1,FIRlength);      % allocate memory for filter samples
86
87     for k=1 : FIRlength %filter length L = filter_order+1, odd.
88         % irad = (double((k-1)-nfs/2)*(double(fs)/double(nfs)))/2;
89 %         idx = (((k-1) - FIRord/2)*(span/FIRord))/2;
90             % idx represents t/Tsymbol in raised-cosine equation
91             % idx is index in range [-6.0, 6.0]
92             % (span/FIRord) equals (1/L);
93             time = ((k-1) - FIRord/2)*(span/FIRord);
94
95             if time == 0
96                 b2(k) = 1; % sinc for t=0;
97             else
98 %                 b2(k) = sin(2*pi*idx)/(2*pi*idx); % sinc
99                 b2(k) = sin(pi*time)/(pi*time); % sinc
100             end
101
102             d = 2*alpha*time; % part of denominator

```

```

103
104     if abs(d) == 1
105         b2(k) = 0;           % prevent division with zero
106     else
107         b2(k) = b2(k) * (cos(pi*alpha*time)/(1-d*d));
108     end
109
110 end
111
112 % Normalization to unit energy so that FILTER ENERGY would be 1
113 h = b2/sqrt(sum(b2.^2));
114 end
115
116
117 fprintf('>===== f_raisedcosine_ms.m
118         ===== END\n\n');
119 end
120 % EOF

```