(work)Title: Demo for the Communication Course using LabView software and software defined radio
Author: Azik Gabulov Bachelor report Electrical Engineering
Faculty: EEMCS, 2019

Summary

To improve the effectiveness of the material presented during the Communications course, the Telecommunication Engineering group at the University of Twente proposed to make use of the LabVIEW environment together with the accompanying NI USRP hardware to implement a demo.

The demo is to help the students gain an intuitive grasp of the concepts and theories taught during the course, as well as provide them with necessary tools to allow them experiment and apply the knowledge they have gained during the course.

The finalized demonstration tool accurately covers the digital modulation techniques taught during the Communication course by showing both time- and frequency-domain representations of simulated signals.

Although, incorporation of the NI USRP hardware was suggested, during this assignment it was found that NI USRP added very little educational value and instead proved to be an unnecessary complication.

Chapter 1. Introduction

1.1. Motivation

Telecommunication Engineering (TE) group in the University of Twente is responsible for organizing parts of *Signal Processing and Communication (SPC)* module in the bachelor program. One of the constituent courses in this module is *Communications*, which discusses the fundamentals of analog and digital communications; covered topics, among others, span concepts such as analog and digital modulation techniques, transition from analog to digital domains, and effects of noise in the transmission channel on the desired system performance.

Due to the fundamental nature of the knowledge presented in this module, it is vital for students to be able to grasp the contents of the course as much as possible, preferably on an intuitive level. Currently, the study process is confined to the course book *Introduction to Analog and Digital Communications* by Simon Haykin [1]. As a result, this leads to a limited number of examples to help students understand the course materials and a lack of interactivity. To counter this, there is a call for creation of demonstration tools that would enable students to interact with the concepts given in the book and understand the intricacies that can be made evident only upon experimentation.

Simulation and demonstration tools present on the Web or built-in to the various software tools as examples, while useful, are often difficult to use, because it is implied that students will have some prior knowledge about the syntax or GUI in those programs.

Moreover, many of the more capable software solutions available on the market are highly priced and unavailable for personal use by students. Such obstacles can divert students from learning about the principles those programs are meant to show. Given the rather highly demanding workload of the *Signal Processing and Communications* module such a distraction proves to be unfavorable.

In the past, at the TE group, an individual research assignment to create a demonstration tool specifically tailored for the SPC module was completed [2]. As a result of that assignment, *NI LabVIEW* software and *NI Universal Software Radio Peripheral (NI USRP)* hardware were used to create a LabVIEW Virtual Instrument, that allowed a great deal of interactivity and consequently provided more insight into the subject matter. Yet, the scope of said demonstration tool was limited to FM modulation. In addition, it was programmed in procedural rather that object-oriented manner leading to limited reusability and cumbersome expandability.

1.2 Assignment Description

To expand the set of function offered by such demonstration tools an additional individual design assignment was proposed. In the course of this assignment, the existing demo was used as a starting point to construct a new tool using the same hardware and software, while adhering to the key principle of optimizing the demo for educational use by the students involved in the SPC module. Furthermore, to enable expansion of the demo to cover more topics in the future, its code should be written with reusability in mind.

Given the ubiquitous presence of digital media nowadays, it was chosen that the scope of the new demo will be primarily focused on various digital modulation techniques as well as their performance when exposed to noisy communication channels. Using the demo, students would have to be able to experiment with the following concepts:

- How are modulated waveforms generated for different modulation techniques?
- What do modulated waveforms look like for a different modulation techniques?
- What are the spectral characteristics of the modulated waveform, for a different modulation techniques and a given set of input parameters?
- What are the effects of additive channel noise on the spectral contents and overall performance for different modulation techniques?

1.3 Contents of the Final Product

On the grounds that the demonstration tool was built for educational purposes, the set of modulation techniques simulated was chosen from among those that are taught in the course of the Communications part of the SPC module. The final product demonstrates the woking principles of the following modulation techniques:

1. Binary amplitude shift keying (BASK)
2. Binary phase shift keying (BPSK)
3. Quadriphase shift keying (QPSK)
4. Offset quadriphase shift keying (OQPSK)
5. Sunde's binary frequency shift keying (SBFSK)
6. Minimum shift keying (type I and type II) [3] (MSK).

While the necessity to implement the behavior of various modulation schemes utilizing the USRP hardware is important, it is recognized that the high purchasing cost of the hardware will lead to limited availability of the USRP units. Should the implementation of the demo be primarily based on having the USRP as a key part, the effectiveness of the final product would be hindered by the organizational and logistical aspects, exemplified by, for instance, the fact that the students would only have access to the USRP for a limited time.

Also, in the process of designing the final product, various experiments lead to an understanding that employing the USRP does not add much of an educational value to the demonstration tool. Because of that the final product was design in a way that makes the use of the USRP is completely optional.

To provide the students with the necessary information to use the demonstration, a user manual will be delivered together with the demonstration tool.

1.4 LabVIEW Programming Environment

LabVIEW is a programming environment developed by the National Instruments Inc. Programming in LabVIEW is done in G, a graphical programming language. Programs written in G language, are canonically called Virtual Instruments (VIs). The interface of each VI consists of two parts:

1. Front Panel - Front panel is how the user interacts with the VI. Front panel controls are to hold values used by the VI and indicators to hold the values returned by the VI. This is where the main GUI of the developed demonstration tool shall be placed.
2. Block Diagram - Various functions that need to be implemented by the VI are coded by means of placing blocks on the Block Diagram and connecting the block by the means of wires. In addition to a multitude of a functional blocks the block diagram supports graphical representation of usual programming paradigms one would expect from a text based language, such as 'if-else' statements, 'for'- and 'while'-loops, etc.

LabVIEW distinguishes between different types information by having numerous data types such as string, boolean, integer, double and so on. Furthermore, LabVIEW supports structural data type, such as array, waveform and cluster. The color of the wire on the block diagram indicates the data type carried by that wire.

LabVIEW and the USRP having been manufactured by the same enterprise are highly optimized to work with one another. Such synergy, absent for other hardware/software configurations justifies the usage of the USRP together with LabVIEW.

1.5 Outline of the Report

The rest of this report is organized as follows. In Chapter 2 the theory pertaining to concepts of digital modulation as well as other theory necessary to justify some of the design decisions made will be overviewed. In Chapter 3 the design procedure and methods used to construct the demonstration tool will be explored. Following that, Chapter 4 will be dedicated to the presentation of results as well as some of the limitation of the final demo. Chapter 5 will conclude the report with discussion and exploration of possible future work concerning this demo.

Chapter 2. Theory

In this chapter overview of the theory necessary to implement the demo will be presented.

2.1 Overview of Modulation Principles

2.1.1 Generation of Baseband Digital Information Signals

In a digital communication system, the process of transmitting a digital message signal begins with a binary data stream denoted as $\{b_k\}$ consisting of logical zeroes and ones emitted by the source of binary information.

The binary data stream $\{b_k\}$ is then passed through a level encoder, which assigns amplitudes to symbols in $\{b_k\}$ in a predetermined fashion, resulting in a set of amplitudes $\{a_k\}$.

These amplitudes enable electrical representation of the binary data stream $\left\{b_k\right\}$ as a sequence of pulses, formally defined as:

$$b\left(t\right) = \sum_{k=-\infty}^{\infty} a_k g\left(t - kT_b\right), \tag{T.1.1}$$

$$\text{where } g\left(t\right) = \text{rect}\left(\frac{t - \left(k + \frac{1}{2}\right)T_b}{T_b}\right) \tag{T.1.2}$$

where T_b is the bit duration used in the communication system and $g\left(t - kT_b\right)$ is the basic shape of a pulse shifted to time $t = kT_b$, which is the moment, when k-th bit starts being emitted from the source of binary information.

It should be noted that the basic pulse shape given by the particular formulation in (T.1.2) is not the only possible shape for the pulse. Different pulse shaping strategies can be used and and each have their own advantages, such as greater margin of error for sampling instances in the receiver and less inter-symbol interference. However, due to the fact that they were not implemented in the final demonstration tool, other pulse shapes are beyond the scope of this work.

If the system's design constraints allow it, the serial data stream $b(t)$ can already be used for transmissions across low-pass channels such as twisted wire pair or coaxial cable. However, to make transmission over band-pass channels, such as satellite or wireless, a possibility, the incoming binary stream must be modulated to a sinusoidal carrier of a higher frequency, hereby denoted by $c(t)$. The outcome of the modulation process is a high frequency band pass signal, that shall be referred to as $s(t)$.

2.1.2 Bandpass Assumption

Before continuing with the discussion, it is important to point out that it is assumed that the bandwidth of the modulating binary waveform $b(t)$, denoted as W, is much smaller than the carrier frequency f_c, or mathematically:

$$f_c \gg W \quad \textbf{(T.2)}$$

This condition, also called the *bandpass assumption*, guarantees that there will be no spectral overlap between negative and positive frequencies upon modulation and as a consequence it can be stated that regardless of the chosen modulation technique, the power spectrum of the modulated wave $s(t)$ will always be centered at the frequency f_c, although different modulation techniques will exhibit different spectral behaviors.

2.1.3 Modulation to RF Frequencies

Generic description of a carrier wave $c(t)$ is given by:

$$c(t) = A_c \cos\left(2\pi f_c t + \phi_c\right) \quad \textbf{(T.3)}$$

where A_c is the carrier amplitude, f_c is the carrier frequency and ϕ_c is the initial phase of the carrier wave at time $t = 0$.

It can be observed that there are three parameters that can be varied in accordance with the incoming binary message signal $b(t)$. Consequently, three distinct categories of digital modulation can be identified:

1. Amplitude shift keying: the frequency and the phase of the sinusoidal carrier wave are left unaltered, while the amplitude is varied, or keyed, between values corresponding to the incoming symbols, as a function of $b(t)$.
2. Phase shift keying: the amplitude and the instantaneous frequency of the carrier wave are kept constant while the phase of the carrier is modified in accordance to the source symbol.
3. Frequency shift keying: the amplitude and the phase of the carrier waveform are not directly influenced, while the instantaneous frequency indicates the transmitted symbol.

2.1.4 Carrier Amplitudes in Digital Modulation

While the expression for a carrier wave given in equation (T.3) holds in general, when considering digital communication systems, it is customary to assume that the carrier wave has unit energy, when measured over a bit duration. This translates to assigning a fixed value to for a desired bit duration.

Recalling, that for a sinusoid $c(t)$ to have a unit energy over a bit duration T_b, its RMS amplitude must be $A_{c,RMS}^2 = \frac{1}{T_b}$ and that a sinusoid $c(t)$ with a certain amplitude A_c has an RMS value defined as:

$$A_{c,RMS} = \sqrt{\frac{1}{T_b}\int_0^{T_b} \left|c(t)\right|^2 dt} = \sqrt{\frac{1}{T_b}\int_0^{T_b} A_c^2 \cos^2(2\pi f_c t) dt} = \sqrt{\frac{A_C^2}{2T_b}\int_0^{T_b} 1 + \cos\left(4\pi f_c t\right) dt} = \sqrt{\frac{A_C^2}{2}}$$

we can remove $A_{c,\,RMS}$ between the different expressions and obtain the following result, where

$$A_c = \sqrt{\frac{2}{T_b}} \qquad \textbf{(T.4.1)}$$

Accordingly, rewriting the expression for $c(t)$:

$$c(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_c t + \phi_c) \qquad \textbf{(T.4.2)}$$

for the remainder of this report A_c will always be assumed to be equal to $\sqrt{2/T_b}$.

In the meantime, allow for some modulation technique, according to which the band-pass signal is generated by multiplying the modulation signal $b(t)$ by the carrier wave $c(t)$ as it was the equation (T.3).

$$s\,(t) = b\,(t)\,c(t) = b\,(t)\,A_c \cos\left(2\pi f_c t\right) \qquad \textbf{(T.5)}$$

here for sake of simplicity and without loss of generality, phase ϕ_c has been set to zero. For the remainder of this report, it will remain assumed that the initial phase of the carrier $c(t)$ is zero.

Calculating the energy of the resulting modulated waveform $s(t)$ measured over one bit duration it is found that:

$$E_b = \int_0^{T_b} \left|s\,(t)\right|^2 dt = \int_0^{T_b} \left|b\,(t)\right|^2 \left(\frac{2}{T_b}\right) \cos^2\left(2\pi f_c t\right) dt \qquad \textbf{(T.6.1)}$$

$$E_b = \frac{1}{T_b}\int_0^{T_b} \left|b\,(t)\right|^2 \left[1 + \cos\left(4\pi f_c t\right)\right] dt \qquad \textbf{(T.6.2)}$$

$$E_b = \frac{1}{T_b}\int_0^{T_b} \left|b\,(t)\right|^2 dt + \int_0^{T_b} \left|b\,(t)\right|^2 \cos\left(4\pi f_c t\right) dt \qquad \textbf{(T.6.3)}$$

$$E_b = \frac{1}{T_b}\int_0^{T_b} \left|b\,(t)\right|^2 dt \qquad \textbf{(T.6.4)}$$

where the last line is permissible only if the band-pass assumption holds.

Thus, using an appropriate value for the amplitude of the carrier wave leads to a situation, where the energy spent for transmission of a single bit equals the scaled version of the energy of the incoming binary message signal, and the scaling factor is exactly equal to the bit duration. This in turn has a consequence of justifying the following corollary:

"whatever factor is multiplied by the amplitude of the carrier wave $\sqrt{2/T_b}$ in the formulation of $s(t)$is in fact what would be the square root of resulting energy spent in one bit duration".

2.1.5 Fundamentals of Quadrature Modulation

The method of digital modulation presented by equation (T.5), is mathematically identical to the DSB-SC modulation technique, which is a member of the *amplitude modulation* family used in analog communication systems. With DSB-SC, since the band-pass signal cannot be demodulated using envelope detection, demodulation was carried out using the *coherent detection* method. Given that mathematically the present problem is identical to what happens in DSB-SC, it is warranted to continue the development in parallel to it. A schematic representation of a coherent detector is depicted in Figure 1 [1].

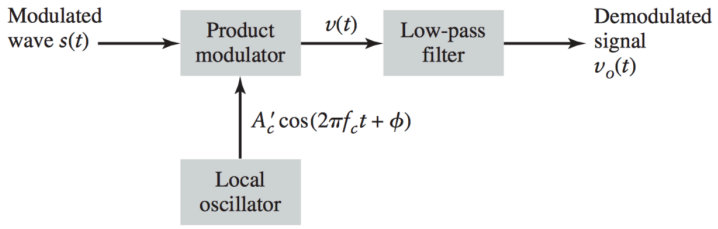


Figure 1: Coherent detector with phase error

Coherent demodulation starts with multiplication of the received signal by a locally generated carrier that should ideally be synchronized with the transmitter's carrier wave in both frequency and phase. Such a signal can be written as:

$$c'(t) = A_c' \cos\left(2\pi f_c' t + \phi_e\right) \qquad \textbf{(T.7)}$$

where ϕ_e represent the difference in phase between the transmitter and the receiver. For the ease of presentation it shall be assumed that there is no mismatch in frequency. Carrying out the product operation, also called *mixing*:

$$v(t) = s(t) A_c' \cos\left(2\pi f_c' t + \phi_e\right) \qquad \textbf{(T.8.1)}$$

$$v(t) = b(t) A_c \cos\left(2\pi f_c t\right) A_c' \cos\left(2\pi f_c' t + \phi_e\right) \qquad \textbf{(T.8.2)}$$

$$v(t) = \frac{1}{2} A_c A_c' \cos\left(4\pi f_c t + \phi_e\right) b(t) + \\ + \frac{1}{2} A_c A_c' \cos\left(\phi_e\right) b(t) \qquad \textbf{(T.8.3)}$$

From equation (T.8.3) it can be observed that the output of the mixer is a superposition of two terms. The first term is the scaled version of the modulating binary waveform $b(t)$, while the second term represents $b(t)$ modulating a carrier wave of a of twice the frequency . This term is subsequently removed by a low-pass filter that follows the mixer, whose output is $v_0(t)$ in equation (T.8.4).

$$v_0(t) = \frac{1}{2} A_c A_c' \cos\left(\phi_e\right) b(t) \qquad \textbf{(T.8.4)}$$

Conversely, it is also true, that in case the binary message signal was originally modulating a sine carrier $\hat{c}(t) = A_c \sin\left(2\pi f_c t\right)$ instead of the cosine carrier $c(t)$ an identical result would be obtained, if the local carrier would be a sine with phase error of ϕ_e radians. The output of the low-pass filter would conform to equation (T.8.4).

Then, from the above derivations, it can be seen that given some non-zero phase error ϕ_e in the demodulator, $b(t)$ that was separated in the outcome is now attenuated by amount equal to $\cos(\phi_e)$. This implies that if the error would be $\pi/2$ radians, the desired output signal $b(t)$ would be completely suppressed. Using the trigonometric rule $\cos(x + \pi/2) = \sin(x)$:

$$v(t) = b(t) A_c \cos\left(2\pi f_c t\right) A_c' \cos\left(2\pi f_c' t + \pi/2\right) \qquad \textbf{(T.9.1)}$$

$$v(t) = b(t) A_c \cos\left(2\pi f_c t\right) A_c' \sin\left(2\pi f_c' t\right) \qquad \textbf{(T.9.2)}$$

$$v(t) = \frac{1}{2} A_c A_c' \sin\left(4\pi f_c t\right) b(t) + \frac{1}{2} A_c A_c' \sin(0)\, b(t) \qquad \textbf{(T.9.3)}$$

and with the subsequent removal of the first term by a low-pass filter

$$v_0(t) = \frac{1}{2} A_c A_c' \sin(0)\, b(t) = 0 \cdot b(t) = 0 \qquad \textbf{(T.9.4)}$$

This phenomenon is known as the *quadrature null effect*.

Although at the first glance, it might seem undesirable, quadrature null effect can be utilized to have two completely independent messages occupy the same portion of the spectrum, whereas they can still be separated at the output.

Suppose that the signal applied to the mixer is a superposition of two modulated signals. One signal is obtained by modulating a message signal to an arbitrary sinusoidal carrier wave $c(t)$. The other is a result of modulating another message signal to the carrier wave $\hat{c}(t)$ of the same frequency, but 90 degrees offset in phase. Such sinusoidal carrier waves $c(t)$ and $\hat{c}(t)$ are said to be in phase quadrature, and therefore referred to as quadrature carriers.

Since sine and cosine waveforms are in phase quadrature we are allowed to generate $s(t)$ as follows:

$$s(t) = b_1(t) c(t) - b_2(t) \hat{c}(t) \qquad \textbf{(T.10.1)}$$

$$s(t) = b_1(t) A_c \cos\left(2\pi f_c t\right) - b_2(t) A_c \sin\left(2\pi f_c t\right) \qquad \textbf{(T.10.2)}$$

where the minus sign is simply a matter of convention.

If the locally generated carrier wave $c'(t)$ exhibits 0 degrees phase error with $c(t)$, at the output of the low-pass filter the message signal that was modulating this carrier $b_1(t)$ is not affected except for being scaled by $\frac{1}{2}A_c'A_c$. More importantly, because of the quadrature null effect the other message signal $b_2(t)$ is completely eliminated, as its carrier wave $\hat{c}(t)$ is in phase quadrature with the locally generated carrier $c'(t)$ and leads to a zero scaling factor as in equation (T.9.4). The other message signal $b_2(t)$ can be then extracted using a second carrier wave $\hat{c}'(t)$ that exhibits 0 degree of phase error with $\hat{c}(t)$.

Assuming ideally synchronized demodulation on the receiver side, the two information signals would not interfere with each other and it would be possible to detect each of them in a manner as if that modulating signal was the only message signal.

This effect can be used to conserve bandwidth when system design requires some set bitrate, via dividing the $b(t)$ in a certain fashion between the two quadrature carriers into $b_1(t)$ and $b_2(t)$. Also, the quadrature null effect can be exploited to utilize some limited bandwidth to the fullest by allowing the system designer to double the rate at which the source emits binary information. These notions can be further generalized into complex signal representation, an approach to study communication systems in a unified and consistent way.

2.2 Complex Signal Representation Approach

Generalizing the equation (T.10.2), we recognize $b_1(t)$ and $b_2(t)$ constitute *in-phase* and *quadrature* components of the complex baseband representation of the modulated band-pass signal $s(t)$, or mathematically:

$$s(t) = s_I(t)\,c(t) - s_Q(t)\,\hat{c}(t) \qquad \textbf{(T.11)}$$

with,$s_I(t)$ and $s_Q(t)$ denoting the *in-phase* and *quadrature* components of the band-pass modulated signal $s(t)$, respectively.

A system that takes arbitrary signals $s_I(t)$ and $s_Q(t)$ as inputs and places the modulated wave $s(t)$ at its output is referred to as a *complex synthesizer.* Conversely, given a band-pass signal $s(t)$ the in-phase and quadrature components are derived by a *complex analyzer*. Schematic representations of a complex analyzer and a complex synthesizer are given in (a) and (b) of Figure 2, respectively [1].

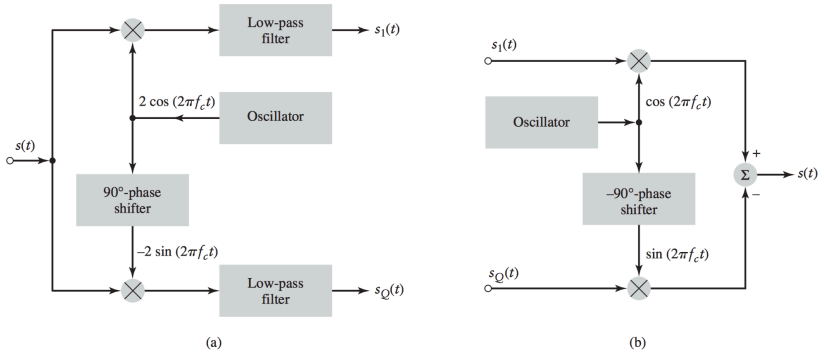


Figure 2: (a) complex analyzer; (b) complex synthesizer.

The in-phase and quadrature components carry the information we wish to transmit, and by themselves completely specify the band-pass signal $s(t)$, assuming that the carrier frequency f_c is already known. Generally, for equation (T.10) to hold these signals need not be binary in nature, the same way $b(t)$ is, meaning that they should not necessarily be generated, by pulse shaping a set of line encoded symbols, in accordance to equation (T.1). This means that it is possible to manipulate binary message signals $b_1(t)$ and $b_2(t)$ before multiplying them with the quadrature carriers.

Combined, $s_I(t)$ and $s_Q(t)$ are used to define $\tilde{s}(t)$, *complex envelope* of the modulated wave $s(t)$, and similarly quadrature carriers $c(t)$ and $\hat{c}(t)$ define $\tilde{c}(t)$ as follows:

$$\tilde{s}(t) = s_I(t) + js_Q(t) = a(t)\exp\left[j\theta(t)\right] \quad \textbf{(T.12.1)}$$

$$\tilde{c}(t) = c(t) + j\hat{c}(t) = A_c\exp\left(j2\pi f_c t\right) \quad \textbf{(T.12.2)}$$

where the complex signals have been described in both polar and Cartesian forms. Relations between $a(t)$, $\theta(t)$, $s_I(t)$ and $s_Q(t)$ follow from the general rules of how complex numbers are treated:

$$a(t) = \sqrt{s^2{}_I(t) + s^2{}_Q(t)} \quad \textbf{(T.13.1)}$$

$$\theta(t) = \tan^{-1}\left[-\frac{s_Q(t)}{s_I(t)}\right] \quad \textbf{(T.13.2)}$$

Using equations (T.11.1) and (T.11.2) $s(t)$ can be rewritten as:

$$s(t) = \Re\{\tilde{s}(t)\,\tilde{c}(t)\} = a(t)\,A_c\cos\left[2\pi f_c t + \theta(t)\right] \quad \textbf{(T.14)}$$

In equation (T.13) $a(t)$ is the envelope of the band-pass signal $s(t)$, and thus represents the amplitude modulated portion, i.e. $a(t)$ can be a function of the binary message of that signal $b(t)$, and $\theta(t)$ is its phase relative to the phase of the unmodulated carrier signal $c(t)$. This difference in phase is otherwise called the *phase evolution* of the signal $s(t)$ and it represents the angle modulated portion of the modulated wave, meaning that both phase shift keying and frequency shift keying techniques vary $\theta(t)$ as a function of the incoming binary wave $b(t)$. In the former case it is the phase evolution itself $\theta(t)$ that is a function of $b(t)$ while in the latter case it is its derivative.

Equation (T.14) can be reformulated into the same form as equations (T.11) using the trigonometric identity $\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$ as follows:

$$\begin{aligned} s(t) &= a(t)\,A_c\cos\left[2\pi f_c t + \theta(t)\right] \\ s(t) &= a(t)\cos\left(\theta(t)\right)A_c\cos\left(2\pi f_c t\right) \\ &\quad - a(t)\sin\left(\theta(t)\right)A_c\sin\left(2\pi f_c t\right) \end{aligned} \quad \textbf{(T.15)}$$

where it can be immediately be identified that:

$$s_I(t) = a(t)\cos\left(\theta(t)\right) \quad \textbf{(T.16.1)}$$

$$s_Q(t) = a(t)\sin\left(\theta(t)\right) \quad \textbf{(T.16.2)}$$

Equations (T.16.1) and (T.16.2) together with equation (T.11) provide all the instruction necessary for the complex synthesizer to generate modulated signals in accordance to an array of modulation techniques. Using this complex synthesizer/analyzer model, it is possible to choose the desired modulation technique by only specifying the corresponding in-phase and quadrature components.

2.3 Modulation Techniques

In the previous section, it was shown that modulated band-pass signals can be constructed from their respective complex envelopes by means of frequency up-translation. This was formulated as the complex synthesizer/analyzer model.

The advantage of such a model when it comes to the functional design of the demonstration tool is that there will not be a necessity to create new "infrastructure" for each and every modulation technique covered. In addition, the method used to generate modulated signal $s(t)$ in LabVIEW can naturally be used to feed data to the USRP.

In literature, the modulation techniques that were chosen to be included in this demo are often defined only in terms of modulated waves $s(t)$. Both polar and Cartesian representations of respective complex envelopes $\tilde{s}(t)$ are only briefly considered.

Because of that, in this section modulation techniques implemented in the final demo will be individually covered, and how their characterization translates to the complex analyzer/synthesizer model will be examined.

2.3.1 Binary Amplitude Shift Keying (BASK)

Prescribed by the BASK modulation scheme, the envelope $a(t)$ of the resulting band-pass signal $s(t)$ is keyed between $\sqrt{2E_b/T_b}$ and 0 for binary symbols 1 and 0, respectively. Formally, BASK modulated waveform is defined as follows:

$$s(t) = \begin{cases} \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t\right), & \text{for bit 1} \\ 0, & \text{for bit 0} \end{cases} \qquad \textbf{(T.17)}$$

Recognizing $\sqrt{2/T_b}$ as the amplitude of the unmodulated carrier wave $c(t)$, it follows that the band-pass signal is generated by a sole in-phase component:

$$s_I(t) = \begin{cases} \sqrt{E_b}, & \text{for bit 1} \\ 0, & \text{for bit 0} \end{cases} \qquad \textbf{(T.18)}$$

which corresponds to choosing the amplitude a_k in the line encoder being equal to $\sqrt{E_b}$ and 0, for b_k equal to 1 and 0, respectively. In other words, BASK is generated using a single in-phase component, which is equal to $b(t)$ generated using an 'on-off' line code:

$$s_I(t) = b\,(t) = \sum_{k=-\infty}^{\infty} a_k g\left(t - kT_b\right) \tag{T.19.1}$$

$$s_Q(t) = 0 \tag{T.19.2}$$

$$\text{where } g\,(t) = \text{rect}\left(\frac{t - \left(k + 1/2\right) T_b}{T_b}\right) \tag{T.20}$$

$$\text{and } a_k = \begin{cases} +\sqrt{E_b}, & \text{for} \quad b_k = 1 \\ 0, & \text{for} \quad b_k = 0 \end{cases} \tag{T.21}$$

2.3.2 Binary Phase Shift Keying (BPSK)

Being the simplest form of phase-shift keying, BPSK involves keying the phase evolution of the signal between 0 and π, for logical 1s and 0s, respectively:

$$s\,(t) = \begin{cases} \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t\right), \\ \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t + \pi\right) \end{cases} \tag{T.22}$$

Owing to the fact that increasing the phase evolution of the sinusoidal signal by half a cycle is equivalent to multiplying it by -1, it is possible to reformulate the definition of the modulated wave $s(t)$ as:

$$s\,(t) = \begin{cases} +\sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t\right), & \text{for} \quad b_k = 1 \\ -\sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t\right), & \text{for} \quad b_k = 0 \end{cases} \tag{T.23}$$

Then, as it was done in case of BASK, we first associate the factor $\sqrt{2/T_b}$ with the amplitude of the carrier wave, and it follows that the modulated wave $s(t)$ is created by a sole in-phase component, which in this case is equal to the binary message signal formed by employing the non-return-to-zero line code, where the value of a_k is equal to $-\sqrt{E_b}$ and $\sqrt{E_b}$ for b_k equal to 0 and 1, respectively. Therefore, for BPSK:

$$s_I(t) = b\,(t) = \sum_{k=-\infty}^{\infty} a_k g\left(t - kT_b\right) \tag{T.24.1}$$

$$s_Q(t) = 0 \tag{T.24.2}$$

$$\text{where } g\,(t) = \text{rect}\left(\frac{t - \left(k + 1/2\right) T_b}{T_b}\right) \tag{T.25}$$

$$\text{and } a_k = \begin{cases} +\sqrt{E_b}, & \text{for} \quad b_k = 1 \\ -\sqrt{E_b}, & \text{for} \quad b_k = 0 \end{cases} \tag{T.26}$$

2.3.3 Quadriphase Shift Keying (QPSK)

According to the QPSK modulation scheme the phase evolution $\theta(t)$ of the modulated wave $s(t)$ is keyed between one of the four equally spaced values $\pi/4$, $3\pi/4$, $5\pi/4$ or $7\pi/4$ to specify the pair of bits, otherwise called a *dibit*, emitted by the source of binary data. The keying action is performed every dibit duration $T_d = 2T_b$. Formal definition of a QPSK modulated signal is as follows:

$$s(t) = \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t + \theta(t)\right) \qquad \textbf{(T.27)}$$

$$\text{where } \theta(t) = \begin{cases} \frac{\pi}{4}, & \text{for dibit 11} \\ \frac{3\pi}{4}, & \text{for dibit 10} \\ \frac{5\pi}{4}, & \text{for dibit 00} \\ \frac{7\pi}{4}, & \text{for dibit 01} \end{cases} \qquad \textbf{(T.28)}$$

Expanding equation (T.27) using the 'cosine of a sum' trigonometric identity we obtain:

$$s(t) = \sqrt{\frac{2E_b}{T_b}} \cos\left(\theta(t)\right) \cos\left(2\pi f_c t\right) - \sqrt{\frac{2E_b}{T_b}} \sin\left(\theta(t)\right) \sin\left(2\pi f_c t\right) \qquad \textbf{(T.29)}$$

Then, upon separating $c(t) = \sqrt{2/T_b} \cos\left(2\pi f_c t\right)$ and $\hat{c}(t) = \sqrt{2/T_b} \sin\left(2\pi f_c t\right)$ as the yet unmodulated quadrature carriers, the following identifications can immediately be made:

$$s_I(t) = \sqrt{E_b} \cos\left(\theta(t)\right) \qquad \textbf{(T.30.1)}$$

$$s_Q(t) = \sqrt{E_b} \sin\left(\theta(t)\right) \qquad \textbf{(T.30.2)}$$

The manner in which the binary data stream coming from the source of data acts on the in-phase and quadrature components of the QPSK modulated wave can be more evident by incorporating the values, in which $\cos\left(\theta(t)\right)$ and $\sin\left(\theta(t)\right)$ result into the equations (T.30.1) and (T.30.2).

To begin with, for each of the values taken on by $\theta(t)$ equations (T.30.1) and (T.30.2) have been evaluated and the results have been placed in Table 1, where for convenience the resulting values of $\cos\left(\theta(t)\right)$ and $\sin\left(\theta(t)\right)$ have been placed as well.

Investigating the values in Table 1 the following facts can be noticed:

- $s_I(t)$ changes its value only as a function of the first bit (or odd bits in case more than one dibit is to be transmitted) with $\sqrt{E_b/2}$ representing a logical 1 and $-\sqrt{E_b/2}$ representing a logical 0;
- $s_Q(t)$ changes its value only as a function of the second bit (or even bits in case more than one dibit is to be transmitted) with $\sqrt{E_b/2}$ representing a logical 1 and $-\sqrt{E_b/2}$ representing a logical 0.

Table 1: In-phase and Quadrature Components in QPSK

	11	01	00	10
$s_I(t)$	$\sqrt{E_b/2}$	$-\sqrt{E_b/2}$	$-\sqrt{E_b/2}$	$\sqrt{E_b/2}$
$\cos\left(\theta\left(t\right)\right)$	$\sqrt{1/2}$	$-\sqrt{1/2}$	$-\sqrt{1/2}$	$\sqrt{1/2}$
$s_Q(t)$	$\sqrt{E_b/2}$	$\sqrt{E_b/2}$	$-\sqrt{E_b/2}$	$-\sqrt{E_b/2}$
$\sin\left(\theta\left(t\right)\right)$	$\sqrt{1/2}$	$\sqrt{1/2}$	$-\sqrt{1/2}$	$-\sqrt{1/2}$

In light of these observations, it follows that $s_I(t)$ and $s_Q(t)$ can be generated as follows. First we split the odd- and even-indexed bits of the original digital message into two different channels. Then, each channel is non-return-to-zero line encoded to produce the sequence of amplitudes $\{a_{k,odd}\}$ and $\{a_{k,even}\}$ that take on one of the values from the set $\left\{\sqrt{E_b/2},\ -\sqrt{E_b/2}\right\}$. Finally $\{a_{k,odd}\}$ and $\{a_{k,even}\}$ are pulse shaped with pulses of duration T_d, to generate two binary data signals $b_{odd}(t)$ and $b_{even}(t)$ for odd- and even-indexed bits, respectively. Mathematically,

$$s_I(t) = b_{odd}\left(t\right) = \sum_{k=-\infty}^{\infty} a_{k,odd} g\left(t - kT_d\right) \quad \textbf{(T.31.1)}$$

$$s_Q(t) = b_{even}\left(t\right) = \sum_{k=-\infty}^{\infty} a_{k,even} g\left(t - kT_d\right) \quad \textbf{(T.31.2)}$$

$$\text{where } g\left(t\right) = \text{rect}\left(\frac{t - \left(k + 1/2\right)T_d}{T_d}\right) \quad \textbf{(T.32)}$$

$$\text{and } a_{k,odd,even} = \begin{cases} +\sqrt{E_b/2}, & \text{for } \ b_{k,odd,even} = 0 \\ -\sqrt{E_b/2}, & \text{for } \ b_{k,odd,even} = 0 \end{cases} \quad \textbf{(T.33)}$$

2.3.4 Offset Quadriphase Shift Keying (OQPSK)

As it was previously shown, when using the QPSK modulation scheme, the in-phase and quadrature components — which are essentially binary signals originating from the odd- and even- indexed bits of the original digital message — of the modulated wave $s(t)$ assume updated values in accordance to the incoming dibits every T_d seconds. Since both of the components undergo an update simultaneously, this means that every T_d seconds the phase evolution $\theta(t)$ might experience:

- either 0° of discontinuity if the next dibit is identical;
- or ±90° of discontinuity if the next dibit is different in only a single bit;
- or ±180° of discontinuity if in the next dibit both bits are different.

The latter can be a disadvantage., because in that case, the linearity of the modulated wave $s(t)$ places more stringent requirements on performance of filters further in the communication path. One consequence of amplitude of the incoming modulated wave $s(t)$ crossing zero, is the fact that effects of non-linear components will become more substantial. For example, in order to avoid large amplitude distortions, it would be critical for filters acting on $s(t)$ to have linear phase characteristics.

One strategy to avoid such undesired amplitude fluctuations would be to offset one of the demultiplexed binary signals, for example $b_{even}(t)$ by T_b seconds with respect to the other, in this case $b_{odd}(t)$. Then, the phase evolution $\theta(t)$ is limited to 0° and ±90° with jumps occurring twice as often, every T_b seconds. Mathematically, with the choice to offset falling on $b_{even}(t)$ this can be written as follows:

$$b_{odd,offset}(t) = b_{odd}(t) \quad \textbf{(T.34.1)}$$

$$b_{even,offset}(t) = b_{even}\left(t - T_b\right) \quad \textbf{(T.34.2)}$$

where $b_{odd}(t)$ and $b_{even}(t)$ are defined by equation (T.31.1) and (T.31.2).

Consequently, the quadrature components in this modulation technique are determined as follows:

$$s_I(t) = b_{odd}(t) \quad \textbf{(T.35.1)}$$

$$s_Q(t) = b_{even}\left(t - T_b\right) \quad \textbf{(T.36.1)}$$

2.3.5 Sunde's Binary Frequency Shift Keying (SBFSK)

As the name suggests, BFSK is a modulation scheme, where transmission of binary symbols 0 and 1 is accomplished by means of alternating the instantaneous frequency of the modulated wave $s(t)$ between two values. A specific type of BFSK, known as Sunde's BFSK (SBFSK), dictates that the alternation should be done between two values that differ by an amount equal to the reciprocal of the bit duration, $1/T_b$, and average to what would be the frequency of the unmodulated carrier wave f_c. In other words, if we denote the instantaneous frequencies corresponding to logical 1's and 0's as f_1 and f_0, respectively then:

$$\frac{f_0 + f_1}{2} = f_c \qquad \text{and} \qquad f_1 - f_0 = \frac{1}{T_b}$$

It is a matter of convention that the higher frequency tone indicates the bit 1. With these relations the band-pass SBFSK modulated signal can be expressed as:

$$s(t) = \begin{cases} \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_1 t\right), & \text{for bit 1} \\ \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_0 t\right), & \text{for bit 0} \end{cases} \quad \textbf{(T.37)}$$

To determine $s_I(t)$ and $s_Q(t)$ that give rise to an $s(t)$ as defined in equation (T.37) we first consider the general description of a CPFSK modulated signal $s(t)$. CPFSK is a broader class of modulation techniques that encompasses both Sunde's BFSK and MSK, which will be dealt with in the next section [4].

$$s(t) = \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t + \theta(t)\right) \quad \textbf{(T.38)}$$

$$\text{where } \theta(t) = \theta(0) + 2\pi \frac{1}{2T_b} h \int_0^t b(\tau) d\tau \quad \textbf{(T.39)}$$

$$\text{where } h = (f_1 - f_0) T_b \quad \textbf{(T.40)}$$

$$\text{and } b(t) = \sum_{k=-\infty}^{\infty} a_k g\left(t - kT_b\right) \quad \textbf{(T.41)}$$

$$\text{where } g(t) = \text{rect}\left(\frac{t - \left(k + 1/2\right) T_b}{T_b}\right) \quad \textbf{(T.42)}$$

$$\text{and } a_k = \begin{cases} +1, & \text{for } \; b_k = 1 \\ -1, & \text{for } \; b_k = 0 \end{cases} \quad \textbf{(T.43)}$$

By definition, for Sunde's BFSK, $h = 1$ and we assume $\theta(0) = 0$. Then, expanding the expression for $s(t)$ using the "cosine of a sum" rule we obtain:

$$s(t) = \sqrt{E_b} \cos\left(2\pi \frac{1}{2T_b} \int_0^t b(\tau) d\tau)\right) \sqrt{\frac{2}{T_b}} \cos\left(2\pi f_c t\right) - \sqrt{E_b} \sin\left(2\pi \frac{1}{2T_b} \int_0^t b(\tau) d\tau\right) \sqrt{\frac{2}{T_b}} \sin\left(2\pi f_c t\right) \quad \textbf{(T.44)}$$

where we have isolated $\sqrt{2/T_b}$ factor as the amplitude of the quadrature carriers. Now, we are able to make the following identifications:

$$s_I(t) = \sqrt{E_b} \cos\left(2\pi \frac{1}{2T_b} h \int_0^t b(\tau) d\tau\right) \quad \textbf{(T.45.1)}$$

$$s_Q(t) = \sqrt{E_b} \cos\left(2\pi \frac{1}{2T_b} h \int_0^t b(\tau) d\tau\right) \quad \textbf{(T.45.2)}$$

In what follows exactly how $b(t)$ influences $s_I(t)$ and $s_Q(t)$ will be examined. Since $h = 1$ for Sunde's BFSK, we see that in any time interval $\left[(k-1)T_b, kT_b\right]$ the phase evolution $\theta(t)$ is being increased or decreased by an amount equal to π radian. More importantly, it should be noted that both an increase and a decrease of π radian lead to the same value of $\theta(t)$, because $\theta(t)$ uses modulo 2π algebra, where $-\pi = \pi$. Moreover, since the choice between whether in a given time interval $\theta(t)$ should increase or decrease is made at instance $t = kT_b$.

To see what the effects of these observations are on baseband components $s_I(t)$ and $s_Q(t)$, for an arbitrary bit sequence the phase evolution and the baseband components have been evaluated and plotted on Figure 3.

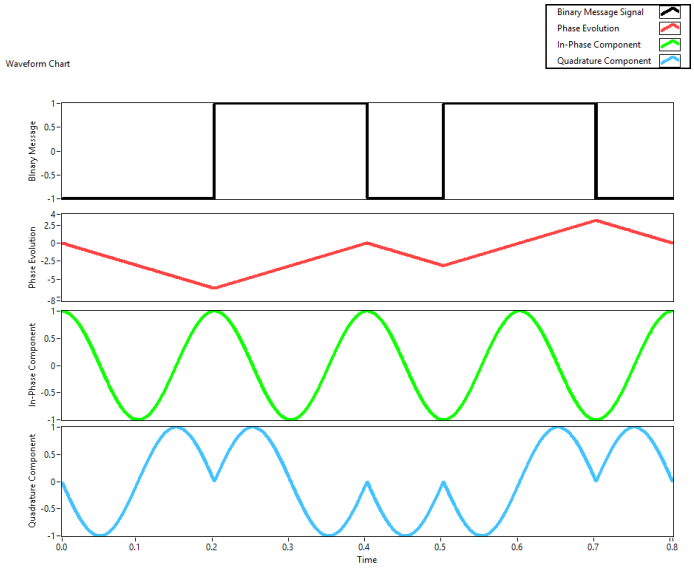


Figure 3: Generation of baseband components according to SBFSK modulation scheme.

Investigating this figure, we see that a reversal in the phase evolution corresponds to the sinusoid in the quadrature components going backwards. In fact, both components experience the "backwards" trend if there is a switch in the binary sequence, but because the shapes of the waveform for a cosine going from $-\pi$ to π and from π to $-\pi$ are identical this is not observable. The in-phase component, does not change its polarity regardless of the information signal.

Based on these arguments and seeing how the the binary waveform manifests itself in the in-phase and quadrature components in Figure 3 we reformulate equations (T.45.1) and (T.45.2):

$$s_I(t) = \sqrt{E_b}\,\left|b(t)\right|\,\cos\left(2\pi\frac{1}{2T_b}t\right) \qquad \textbf{(T.46.1)}$$

$$s_Q(t) = \sqrt{E_b}\;\; b(t)\;\sin\left(2\pi\frac{1}{2T_b}t\right) \qquad \textbf{(T.46.2)}$$

where $b(t)$ is defined in equation (T.41).

2.3.6 Minimum Shift Keying of Type I and Type II (MSK)

In MSK modulation scheme the difference in frequencies for waveforms representing binary 0's and 1's is defined to be equal to half the bit rate, $1/2T_b$. Such a choice is warranted by recognizing that for truncated portions of the modulated wave $s(t)$ signaling a binary 0 or a binary 1 to stay orthogonal to one another their instantaneous frequencies must differ by at least half a bit rate. The baseband components $s_I(t)$ and $s_Q(t)$ in case of MSK are defined in the same way as they were for Sunde's BFSK in equations (T.45.1) and (T.45.2). The phase evolution $\theta(t)$ in case of MSK is as was formulated in equation (T.39), albeit with $h = 1/2$.

$$s_I(t) = \sqrt{E_b}\cos\left(2\pi\frac{1}{4T_b}\int_0^t b(\tau)d\tau\right) \quad \textbf{(T.47.1)}$$

$$s_Q(t) = \sqrt{E_b}\cos\left(2\pi\frac{1}{4T_b}\int_0^t b(\tau)d\tau\right) \quad \textbf{(T.47.2)}$$

$$\theta(t) = \theta(0) + 2\pi\frac{1}{4T_b}\int_0^t b(\tau)d\tau \quad \textbf{(T.48)}$$

Now, consider the quadrature component $s_Q(t)$ and suppose that there two binary 4-bit sequences denoted as $\{b_{k,1}\} = \{1,1,1,1\}$ and $\{b_{k,2}\} = \{1,0,0,0\}$ that are identical in only one bit that are to be transmitted. In addition, we shall associate $\{b_{k,1}\}$ and $\{b_{k,2}\}$ with $\theta_1(t)$, $s_{Q,1}(t)$ and $\theta_2(t)$, $s_{Q,2}(t)$, respectively.

Starting with the first bit, $s_Q(t)$ for both sequences will result in a sine function increasing from 0 at time $t = 0$ to $\sqrt{E_b}$ at time $t = T_b$, since both $\theta_1(t)$ and $\theta_2(t)$ have increased from 0 to $\pi/2$. Then, based on the remaining bits in both sequences, $\theta_1(t)$ in the time interval $[T_b, 4T_b]$ would increase from $\pi/2$ to 2π in increments of $\pi/2$, while $\theta_2(t)$ will decrease from $\pi/2$ to $-\pi$ in increments of $-\pi/2$. If we then evaluate the quadrature component $s_Q(t)$, the resulting waveforms are identical, regardless of the fact that the bit streams that gave rise to them were completely different.

This ambiguity occurs because in time interval $[T_b, 2T_b]$ in this particular case and in any even-indexed interval $[(2k-1)T_b, 2kT_b]$ in general, the waveform for $s_Q(t)$ would be a sine function headed to zero regardless of the information bit. By having an agreement, that the binary symbols occurring in these time intervals are not random, but are the same as in the previous bit interval, the uncertainty of which binary sequence is responsible for producing a given $s_Q(t)$ can be avoided. A similar phenomenon would be observed for the in-phase component $s_I(t)$, which too can be avoided in a similar fashion.

Duplication of bits for instances, when the cosine in $s_I(t)$ and sine $s_Q(t)$ are headed to zero regardless of the message bits, is achieved, by replacing $b(t)$ in formulas for $s_I(t)$ and sine $s_Q(t)$ by $b_{odd}(t+T_b)$ and $b_{even}(t)$, respectively. An offset of the odd-indexed binary message signal for $s_I(t)$ is necessary to account for the fact that the cosine waveform is headed to zero from the outset. Consequently, based on these arguments we mathematically define the baseband components for MSK modulation scheme:

$$s_I(t) = \sqrt{E_b}\cos\left(2\pi\frac{1}{4T_b}\int_{-T_b}^{t} b_{odd}(\tau+T_b)d\tau\right) \qquad \textbf{(T.49.1)}$$

$$s_Q(t) = \sqrt{E_b}\cos\left(2\pi\frac{1}{4T_b}\int_{0}^{t} b_{even}(t)d\tau\right) \qquad \textbf{(T.49.2)}$$

where definitions of $b_{odd}(t+T_b)$ and $b_{even}(t)$ follow from equations (T.31) and (T.34), with line encoder values a_k limited to -1 and 1, for bits 0 and 1, respectively.

These equation can be reformulated yet again, because $b_{odd}(t+T_b)$ and $b_{even}(t)$, change their values every T_d seconds, and these instances coincide with zero crossings of sine and cosine:

$$s_I(t) = \sqrt{E_b}\;\; b_{odd}(t+T_b)\;\cos\left(2\pi\frac{1}{4T_b}t\right) \qquad \textbf{(T.50.1)}$$

$$s_Q(t) = \sqrt{E_b}\;\; b_{even}(t)\;\sin\left(2\pi\frac{1}{4T_b}t\right) \qquad \textbf{(T.50.2)}$$

MSK type II modulated signals build upon the last definition of MSK type I signals, by replacing $\cos[2\pi(1/4T_b)t]$ and $\sin[2\pi(1/4T_b)t]$ with $\left|\cos[2\pi(1/4T_b)t]\right|$ and $\left|\sin[2\pi(1/4T_b)t]\right|$.

2.4 Binary Symbol Detection

A complex analyzer takes the band-pass signal $s(t)$ as an input and extracts its baseband components. After this, it is still required to analyze the resulting waveforms to determine the binary sequences contained in them. In this section the theory necessary for detection of binary symbols will be discussed. It is assumed that $s_I(t)$ and $s_Q(t)$ have been detected by an ideally synchronized coherent detector.

We begin by investigating the expressions for baseband components for each of the modulation techniques. For convenience of the reader, these expressions have been reproduced, and can be examined in Table 2.

For all the modulation schemes, the expressions can be divided into two parts. One part is $\{a_k\}$, the the line encoded representation of the binary sequence $\{b_k\}$. This is what the receiver side has to determine in order to reproduce the digital message. The other part is a scheme-specific function. Since it is assumed that both parties are aware of the modulation scheme used, this part is known at the receiver. Thus, between any two instances, when $s_I(t)$ or $s_Q(t)$ change their values, the only unknown parameter is a_k.

In any time interval a_k is valid, say from $(k-1)T$ to kT, the baseband components of the modulated wave can analyzed using the concepts pertinent to a one-dimensional Euclidian signal-space [3]. Note, that the duration of this time interval is modulation scheme-specific. Then, a_k plays the role of the coefficient, and the term with which it is multiplied, plays the role of a basis function. Mathematically,

$$s_{I,Q}(t) = a_k \cdot p(t) \qquad \text{for} \qquad (k-1)T < t < kT \qquad \textbf{(T.51)}$$

where we define $p(t)$ abstractly denotes the function with which a_k is multiplied.

Table 2: In-phase and Quadrature Components in Implemented Modulation Schemes

	$s_I(t)$	$s_Q(t)$
BASK	$\sum_{k=-\infty}^{\infty} a_k \text{rect}\left(\frac{t-(k+1/2)\,T_b}{T_b}\right)$	0
BPSK	$\sum_{k=-\infty}^{\infty} a_k \text{rect}\left(\frac{t-(k+1/2)\,T_b}{T_b}\right)$	0
QPSK	$\sum_{k=-\infty}^{\infty} a_{k,odd}\ \text{rect}\left(\frac{t-(k+1/2)\,T_d}{T_d}\right)$	$\sum_{k=-\infty}^{\infty} a_{k,even}\ \text{rect}\left(\frac{t-(k+1/2)\,T_d}{T_d}\right)$
OQPSK	$\sum_{k=-\infty}^{\infty} a_{k,odd}\ \text{rect}\left(\frac{t-kT_d}{T_d}\right)$	$\sum_{k=-\infty}^{\infty} a_{k,even}\ \text{rect}\left(\frac{t-(k+1/2)\,T_d}{T_d}\right)$
SBFSK	$\sqrt{E_b}\sum_{k=-\infty}^{\infty} \text{rect}\left(\frac{t-(k+1/2)\,T_b}{T_b}\right)\cos\left(2\pi\frac{1}{2T_b}t\right)$	$\sqrt{E_b}\sum_{k=-\infty}^{\infty} a_k\text{rect}\left(\frac{t-(k+1/2)\,T_b}{T_b}\right)\sin\left(2\pi\frac{1}{2T_b}t\right)$
MSK I	$\sqrt{E_b}\sum_{k=-\infty}^{\infty} a_{k,odd}\ \text{rect}\left(\frac{t-kT_d}{T_d}\right)\cos\left(2\pi\frac{1}{4T_b}t\right)$	$\sqrt{E_b}\sum_{k=-\infty}^{\infty} a_{k,odd}\ \text{rect}\left(\frac{t-(k+1/2)\,T_d}{T_d}\right)\sin\left(2\pi\frac{1}{4T_b}t\right)$
MSK II	$\sqrt{E_b}\sum_{k=-\infty}^{\infty} a_{k,odd}\ \text{rect}\left(\frac{t-kT_d}{T_d}\right)\left\lvert\cos\left(2\pi\frac{1}{4T_b}t\right)\right\rvert$	$\sqrt{E_b}\sum_{k=-\infty}^{\infty} a_{k,odd}\ \text{rect}\left(\frac{t-(k+1/2)\,T_d}{T_d}\right)\left\lvert\sin\left(2\pi\frac{1}{4T_b}t\right)\right\rvert$

Given a one-dimensional signal-space we can determine the value of the coefficient a_k through a measure of orthogonality between the analyzed waveform and the corresponding basis function [3]. This can be done by means of a cross-correlation operation between the received baseband component and $p(t)$.

$$a_k = \frac{1}{P}\int_{(k-1)T}^{kT} s_{I,Q}(t)p(t)dt \quad \textbf{(T.52)}$$

$$\text{where } P = \int_{(k-1)T}^{kT} p^2(t)dt \quad \textbf{(T.53)}$$

here, P is the energy of the basis function when measure over one interval.

A device that performs the bit detection using as described by equation (T.52) is commonly referred to as the correlator receiver [4]. In general, a time offset τ should be considered in equation (T.53), where $p(t)$ should be replaced by $p(t-\tau)$, but since the timing issues were not encountered in designing of the demonstration tool, any effects stemming from timing errors are beyond the scope of this work.

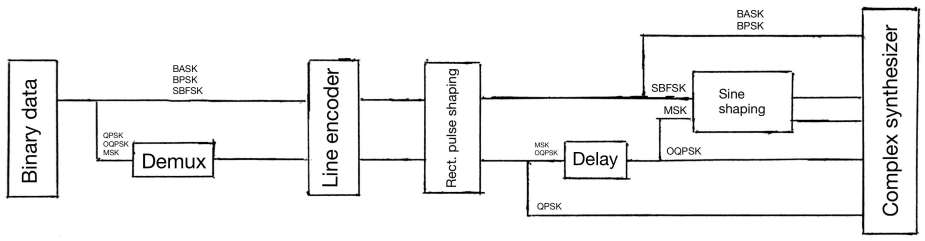
Closer examination of Table 2 and equation (T.52) and (T.53) reveals that for modulation techniques, where the basis functions are exclusively rectangle functions with no other multiplicands, the operation performed by the correlator-receiver reduces to a much simpler scaled integrate-and-dump function. Scaling factor $1/P$ in that case is exactly equal to $1/T$, the reciprocal of the integration interval. Such reduction can be explained if it is recognized that for rectangle functions the integrand in equation (T.53) is simply unity.

Then the binary symbol is decided upon, after the calculated a_k is processed by a decision making device, where it is compared to some a priori known threshold. In all the modulation techniques included with the final demo, except BASK, this threshold is 0. In BASK, it is half the value of a_k for bit 1. In other words, half of the amplitude that the line encoder assigns to bit 1.

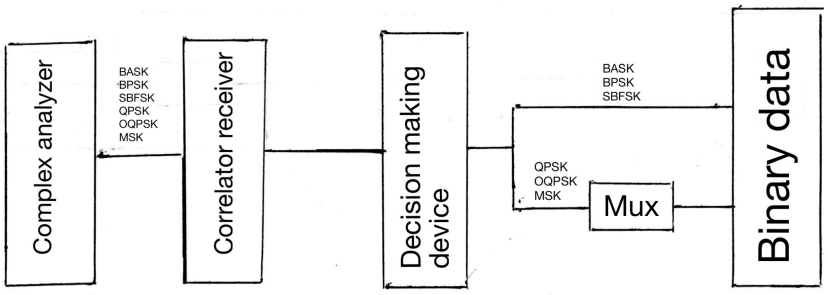
If the bandpass assumption holds, then the cross-correlation operation can be used not only to determine a_k, but also as a low-pass filter. The advantage of this is that we can use the correlator receiver to remove the high-frequency terms coming from the mixing operation in the coherent detector. This way, there is no need for another low-pass filter.

Chapter 3. Functional Design

The discussion in this section will serve as a basis for final implementation of the demonstration tool and present an overview of what kind of systems have to be realized using the LabVIEW programming environment and its tools, to apply the theoretical approach developed in the previous chapter.



(a)



(b)

Figure 4: Function block diagram of a band-pass communication system

To begin the design process of the demonstration, first a high-level block diagram of all the functions necessary to replicate a typical digital communication system in software was created and can be examined in Figure 4 (a) for the transmitter side and Figure 4 (b) for the receiver side.

This diagram serves as a summary for the theory presented in the previous chapter. Each path in this diagram shows the necessary actions to convert the binary sequence $\{b_k\}$ into appropriate in-phase and quadrature baseband components. Labels on the paths indicate, which modulation techniques take that path. Branching of a path means that the modulation techniques that previously had to undergo the same operations will have to be treated differently from there on.

First of all, any digital communication starts with the source of binary data emitting an array of binary symbols. This is the same for all modulation techniques involved.

First branching happens because of the fact that in M-ary modulation techniques such as QPSK and OQPSK, and the more involved MSK, the binary data stream $\{b_k\}$ has to be demultiplexed into odd- and even-indexed channels $\{b_{k,odd}\}$ and $\{b_{k,even}\}$. Furthermore, pulse shapes for each of these channels have twice the duration of what would be required from the binary modulation schemes.

Following this, the binary symbols should be line encoded and pulse shaped using one of the available line encoding schemes. It is important to recognize, that different modulation techniques require the binary symbols of the digital message to be line encoded in different ways. For example, while BPSK and BFSK require their binary signal $b(t)$ to be 'non-return-to-zero' line encoded, BASK demands an 'on-off' line encoded binary signal.

An additional function that has to be realized for OQPSK and MSK is the offset for one of the demultiplexed binary signals. In the diagram this is reflected as branching of the lower path. QPSK goes directly to the complex synthesizer.

After having been time delayed, paths for OQPSK split, because while OQPSK requires no further manipulations and goes into the complex synthesizer, whereas MSK has to be sine shaped.

In the meantime, in the top path another branching takes place. This is to indicate that much like MSK, for Sunde's BFSK the binary signal has to undergo additional sine shaping. BASK and BPSK are to go directly to the complex analyzer.

After all the necessary actions to convert the binary sequence $\{b_k\}$ into baseband components $s_I(t)$ and $s_Q(t)$ have been completed, they are multiplied with the quadrature carriers by the complex synthesizer, which subsequently generates the $s(t)$ by means of a subtraction.

On the receiver side the sequence of events begins with a complex analyzer, that takes the waveform $s(t)$ as an input and provides its baseband components $s_I(t)$ and $s_Q(t)$.

Throughout the construction of this demonstration tool it was assumed that both parties involved in signal transmission are aware of what specific modulation scheme is used. This means that the waveform shapes that correspond to binary symbols are known at the receiver end. With that in mind, given $s_I(t)$ and $s_Q(t)$ the receiver is able to detect the particular bit sequences encoded into these waveforms using the correlator-receiver and a relevant basis function $p(t)$ as it was described in the Section 2.4.

The output of the correlator is compared to a modulation scheme specific threshold and a decision between whether a binary symbol 0 or 1 was transmitted is made.

Following this, if the modulation scheme was such that it required demultiplexing of the binary stream prior to line encoding, in this case QPSK, OQPSK or MSK, the original binary sequence can be restored by passing the outputs of the decision making devices though a multiplexer.

Chapter 4. Implementation

The choice was made for students to have all the interaction with the final product through a limited set of top-level VIs. The students not concerned with the inner operation of the demo would not be required to know about how the VI works. However, because the demonstration tool is meant for educational purposes, the code governing its operation was written in a way that a student, or someone who is new to LabVIEW could understand it with little as little effort as possible.

The demonstration tool was organized in an object-oriented manner. A top-level VI was created to carry the GUI of the demo as well as serve as the communication hub between other VIs. To aid to the top-level VI, additional VIs were created to implement various theoretical concepts following the design approach presented in the previous chapter.

Attention was paid in order to ensure reusability of these auxiliary VIs if it is decided to extend the demo at some point in the future. Some of the auxiliary parts of the demo were coded in a way that would guarantee their satisfactory standalone operation.

4.1 Block Diagram of the Top-Level VI

The block digram of the top level VI was exported as an image file and can be viewed in Figure 5.

First, the entire code governing the operation of the VI is placed in a while-loop. This is for the top level VI to run continuously. It is also possible to launch the VI in the single execution mode if so is desired, by pressing the RUN ONCE button present of the front panel. In this mode the VI will stop the execution upon completion of the first loop. This proves to be advantageous if there is a need to run the VI using very high sampling frequencies or to simulate the transmission of very long digital messages.

The block digram of the demo is divided into frames. In LabVIEW such a structure guarantees sequential execution of frames left to right. This is necessary in order to ensure the correct order of execution of various operations. For example, this is necessary to prevent having the spectrum of the simulated received signal being shown, before the bits of the digital message are received and detected.

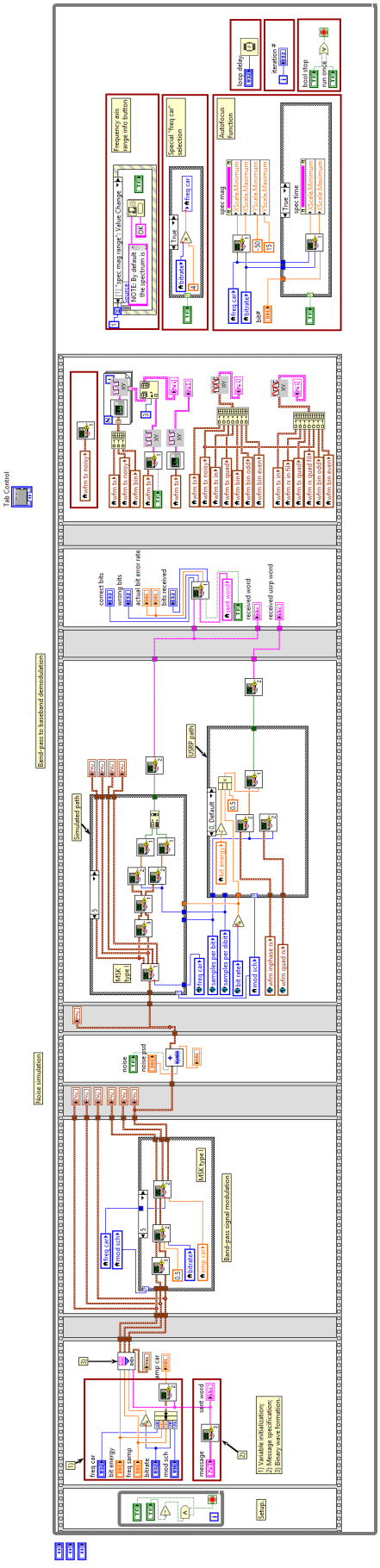


Figure 5: Block diagram of the top-level VI

4.1.1 Frame 1. Setup

We begin the discussion with the leftmost pane. This pane is used to allow an “engine lock” behavior for the top-level VI. Should the user not flip the START toggle on the front panel, the execution of the top-level VI will not continue past this frame. Moreover, a second boolean control present in the while-loop included in this frame allows the user to pause the execution of the demonstration tool without stopping the execution of the virtual instrument. This is useful in cases when one wants to investigate an element on the front panel or place a probe somewhere on the block diagram without the problem that the values are soon going to be updated in the next loop of the VI.

4.1.2 Frame 2. Message and Variable Specification. Binary Wave Generation

The main operation of the top-level VI begins in the second frame. Here in the bottom box the ‘message’ cluster can be seen. This cluster consists of three variables:

- User input message,
- A boolean variable determining whether the user input string is used, or a random one is generated instead,
- An integer that specifies what the length of the randomly generated string should be.

Generation of the random string is done by [randomWord.vi]. The block digram of this subVI can be examined in Figure 6, where its operation is explained by the means of comments.

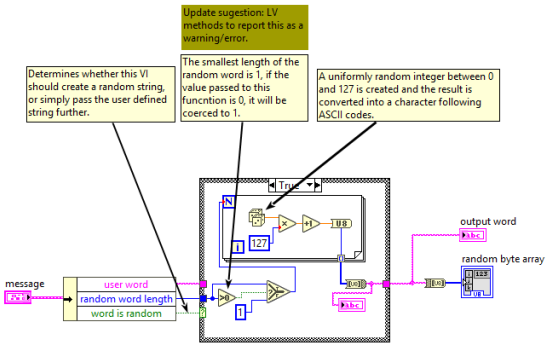


Figure 6: Block diagram of [randomWord.vi]

At the left side of the top box in Figure 5 one can see the 5 variables specified by the user on the front panel:

- Carrier frequency
- Modulated energy per bit
- Sampling frequency
- Bitrate
- Modulation scheme

These variables are combined into a cluster and sent to the [globalInitialization.vi] along with the 'message' cluster created by [randomWord.vi]. [globalInitialization.vi] takes these variables and does two functions. Firstly, it calculates new values that will be used later, such as the carrier amplitude according to the convention that was discussed in Section 2.1.4. Secondly, it passes all its inputs and the values it has calculated into the global variable [globalDigitalModulation.vi].

Global variables are a LabVIEW paradigm that allow different VIs to share values regardless of whether they are in the same hierarchical tree or not. The main reason to use them in the demo is to improve presentation on the block diagram. Calculations are done by placing blocks and connecting them with wires, which leads to unnecessary clutter. By having the values be easily accessible when needed eliminates the requirement to have them calculated locally every time they are needed.

The most important function carried out in this frame is done inside the [stringInBinaryComponentsOut.vi] subVI; its block digram can be examined in Figure 7. Here:

- The user-specified or randomly generated string is converted into ASCII values;
- Each ASCII value is converted into binary representation;
- Bits of the ASCII values are demultiplexed;
- Both demultiplexed and original binary streams are line encoded and subsequently pulse shaped by [lineEncoder_Mk2.vi], a second-level SubVI.
- The sampling information cluster is generated.

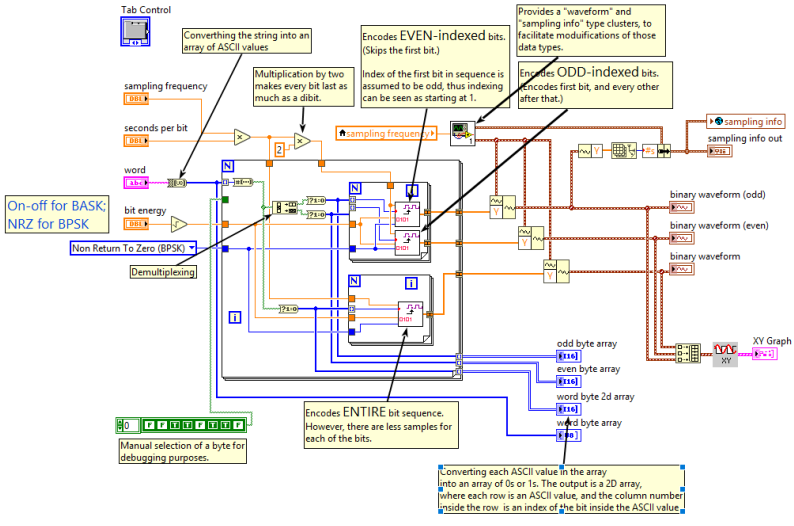


Figure 7: Block diagram of [stringInBinaryComponentsOut.vi]

In LabVIEW to be able to use various function generators a special kind of data cluster, called “sampling info” is required. Hence, the ‘sampling info’ cluster generated in [stringInBinaryComponentsOut.vi] subVI is central to ensure correct operation of the demonstration tool. This cluster contains two elements:

- Sampling frequency - Sampling frequency indicates what is the is the sampling interval dt used in waveform data types. [stringInBinaryComponentsOut.vi] uses sampling frequency specified by the user to define this element. In order to perform any mathematical operations on two waveform data type signals they need to have equal sampling frequencies.
- Number of samples - This element indicates the number of samples contained in the waveform data type signal.

Waveform data type (WDT) signals are used in LabVIEW as a way to represent time-domain signals. WDT is a collection of three separate LabVIEW entities:

- dt - This value of type double indicates the time between two samples in the Y-array. It is equal to the reciprocal of the sampling frequency specified by the user.
- Y-array - This is an array that represents the sampled version of an analog signal.
- t0 - This is a variable of LabVIEW data type called “timestamp”. This is the time of occurrence of the first sample in the Y-array.

4.1.3 Frame 3. Baseband and Bandpass Signal Generation

In this frame procedures necessary to embed the binary waveform into the baseband components of the band-pass signal take place. This frame contains a case structure, which is the LabVIEW equivalent of if-else statements found in the text based programming languages. This case structure has one case per modulation schemes used. Below, operations done in case of each modulation schemes are discussed.

Case 1. BASK.

In [stringInBinaryComponentsOut.vi] the binary sequences were line encoded only using the ‘non-return-to-zero’ line code. Such a decision was justified by the fact that of all the modulation techniques only BASK requires a different line code. ‘On-off’ keying used by the BASK can be obtained from the ‘non-return-to-zero’ line code by first offsetting the NRZ line encoded binary signal by an amount that represents a logical one and dividing the offset waveform by two. This is sufficient to obtain a binary waveform that conforms to what was discussed in Section 2.3.1.

Case 2. BPSK

BPSK requires no further manipulation of the NRZ line encoded binary signal. The output of the [stringInBinaryComponentsOut.vi] subVI can be directly used as the input to the complex synthesizer.

Case 3. QPSK

Because [stringInBinaryComponentsOut.vi] not only pulse shapes the full original binary sequence, but also its odd- and even-indexed bits, the outputs resulting from [stringInBinaryComponentsOut.vi] can be already be used to feed the complex analyzer, with no additional actions necessary.

Case 4. OQPSK

On the grounds of the fact that the only difference in baseband components of QPSK when compared to OQPSK is the time offset necessary for one of the channels, the only extra functionality required in order to proceed was the time offset. This was implemented by the [bitStreamOffset.vi] subVI. The block diagram of this subVI can be found in Figure 8, where its exact operation is explained in the comments.

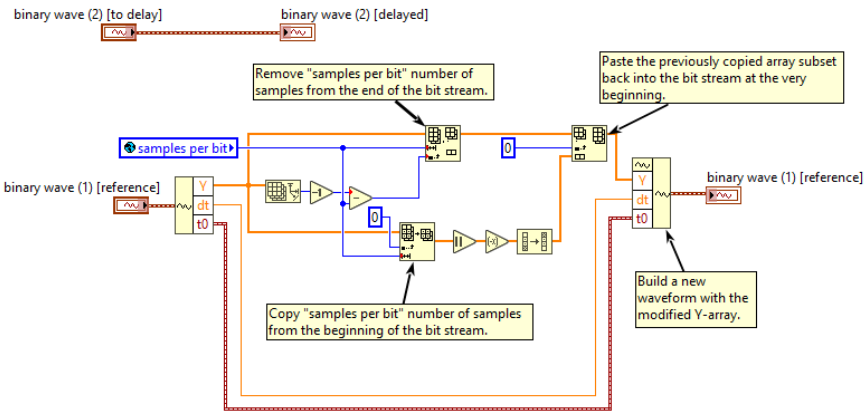


Figure 8: Block diagram of [bitStreamOffset.vi]

Case 5. Sunde's BFSK

Generation of Sunde's BFSK was carried out according the equations (T.45.1) and (T.45.2), rather than (T.46.1) and (T.46.2). For this purpose a subVI named [sundeModulator.vi] was created.

In order to perform the integration operation the built-in [integral x(t).vi] block was used, and the default option for the integration method — the Simpson's rule — was not changed, as it performance proved to be sufficiently adequate.

The full binary waveform generated by [stringInBinaryComponentsOut.vi] was normalized such that its amplitudes is confined to the values of +1 or -1 to ensure proper operation. Because the information about the modulated bit energy was contained in the amplitude of the binary waveform, the outputs of the cosine and sine operators had to be multiplied with the initial amplitude of the binary waveform. This ensures that the energy per bit in the resulting modulated waveform is at the right value.

Case 6. MSK (type I)

Baseband components for MSK type I were generated using the two odd- and even indexed binary waveforms outputted by [stringInBinaryComponentsOut.vi].

First, as prescribed by formulation in equations (T.49.1) and (T.49.2), one of the channels was offset from the other by one bit duration using the [bitStreamOffset.vi] subVI, previously mentioned in Case 4.

[sundeModulator.vi] was used with the same parameters for the integration method and the same adjustments were made to the binary waveforms involved to ensure that the energy per bit in the resulting modulated waveform is at the right value.

Case 7. MSK (type II)

Using the odd- and even indexed binary outputted by [stringInBinaryComponentsOut.vi], in-phase and quadrature components of the modulated wave for MSK type II were obtained according to the formulation presented in Section 2.3.6.

Because of the particular method of operation of [bitStreamOffset.vi] multiplication of the even- and odd-indexed binary waveforms with the sine and cosine half-cycles needed to precede the offsetting operation rather than succeed it. The multiplication action was realized in [mskShaping.vi].

The offsetting operation was carried out using [bitStreamOffset.vi] yet again.

The Complex Synthesizer

For all modulation techniques cases are terminated at the [complexSynthesizer.vi]. This VI performs two monumental tasks:

- It modulates the two quadrature carrier with the incoming in-phase and quadrature components. This is done simply by multiplication in agreement with equation (T.11).
- It sends the waveforms of the baseband components to a global variable [globalUSRP.vi], which is used to enable the transmission over the loopback cable using the USRP.

4.1.4 Frame 4. Noise Simulation.

In this frame a Gaussian distributed white noise is added to the modulated wave generated in the previous frame. The noise variance is defined by the user on the front panel.

4.1.5 Frame 5. Band-pass to Baseband Demodulation, Bit Detection

Frame 5 is the simulated representation of a receiver. Due to the fact that different modulation schemes dictate different detection strategies, a case structure is used to enable scheme-specific demodulation.

The Complex Analyzer

All cases in the case structure begin with a complex analyzer implemented in the subVI called [complexAnalyzer.vi]. Design of this subVI only implements ideally coherent demodulation.

The operation of the complex analyzer is governed by the quadrature null effect. Received modulated waveform is multiplied by two locally generated quadrature carriers. High frequency components created as a result of such mixing in the receiver are to be removed. It Section 2.4, it was already mentioned that this can be achieved in two ways: by using a dedicated low-pass filter, or leaving the high-frequency components to be removed by a correlator receiver, later in the path, when the bit detection is carried out.

In the final demo both approaches have been realized. Such redundancy was necessary since results of both approaches proved to be useful in in implementations of other portions of the demo. Because of that, [complexAnalyzer.vi] has four outputs: two for the mixed, but unfiltered signals, and two for the mixed and filtered.

First we will consider the design consideration pertaining to a dedicated lowpass filter. It the block diagram of [complexAnalyzer.vi] twin filters can be noticed in the rightmost box. This filter was implemented using the built-in LabVIEW Filter Express VI. The filter was configured to be a finite impulse response filter with the maximum possible 511 taps for best possible performance, and a cut-off frequency equal to the carrier frequency, as it was defined by the user. The choice for the carrier frequency is justified because after the mixer, the higher frequency components are shifted to twice the carrier frequency. As a result, the spectrum between W, which denotes the bandwidth of the digital message, and $2f_c$ is empty.

Filter Express VI executes the filtering process by means of discrete convolution, which has an undesirable side effect of introducing 255 leading zeroes to the beginning of the waveform and accompanying removal of 255 samples from the end of the waveform. Supposedly, this is to have the same lengths for input and output waveforms.

To counter this a special set of support VI named [zeroPadder.vi] and [zeroRemover.vi] have been created. [zeroPadder.vi] introduces 511 trailing zeroes to the waveform before it enters the filter. When filtering action happens, the waveform is offset by 255 samples to the right, but since the waveform was padded with zeroes no information is lost. Subsequently, to ensure that length of the waveform is restored to what it was before the [zeroPadder.vi], [zeroRemover.vi] takes action by removing 256 leading and 255 trailing samples.

Scheme-specific Demodulation Strategies

On the grounds of the fact that after the complex analyzer the system behavior in this frame is scheme-specific, the discussion is going to continue in the manner similar to Frame 3. Each of the cases in the case structure will be individually touched upon.

Case 1. BASK

For BASK modulation scheme the mixed but unfiltered in-phase component of the modulated wave is directly passed to the [integrateAndDump.vi]. This subVI was designed to implement the theory that was discussed in Section 2.4.

Integration is carried out every bit duration and the resulting values, in the form of an array, are passed to the output terminals for subsequent comparison to a threshold.

Comparison is accomplished by employing the [comparator.vi] subVI, where each element in the output array produced by [integrateAndDump.vi] is compared to a threshold. Threshold for BASK was determined to be equal to line encoded amplitude for bit 1 times bit duration divided by two. This value conforms to what was said about the BASK threshold in Section 2.4. [comparator.vi] outputs the bits as an array of boolean data type.

Case 2. BPSK

To extract the binary information, processing actions executed for BPSK scheme are the same as for BASK, with the only difference being the usage of a different value as the input to [comparator.vi]. For BPSK — and as it was mentioned in Section 2.4 — for all the other implemented schemes, except BASK, the threshold is equal to 0.

Case 3. QPSK

Based on the fact that one way to interpret QPSK is to view it as two parallel BPSK signals, there is significant similarity in methods of detection as well.

First, mixed and yet unfiltered signals from [complexAnalyzer.vi] are applied to two parallel [integrateAndDump.vi] subVIs. Integration intervals are adjusted to be equal to two bit durations, in order to correctly reflect the duration of the basis function.

Finally, two boolean data type arrays, corresponding to the odd- and even- indexed bits of the original digital message are interleaved.

Case 4. OQPSK

First, the unfiltered signals obtained from the [complexAnalyzer.vi] subVI are processed in order to remove the the offset that was introduced in frame 3 by [bitStreamOffset.vi]. This is accomplished by the supporting VI called [bitStreamOffsetRemover.vi]. At the output of this VI the signals are identical to what would have been the mixed unfiltered signals obtained from [complexAnalyzer.vi] should the modulation scheme have been QPSK instead of OQPSK.

Because of this equivalence, further operations done in this frame are the same as in the case of QPSK. The signals are passed through an [integrateAndDump.vi], whose outputs are then compared to a threshold in [comparator.Vi]. Two resulting boolean arrays are subsequently interleaved.

Case 5. Sunde's BFSK

In Sunde's BFSK, the entire bit sequence can be extracted solely from the quadrature component, the role of the in-phase components is only to maintain the envelope of the modulated wave constant. For this reason, only the mixed quadrature signal is considered. Detection of binary symbols is accomplished as follows.

First the unfiltered quadrature component of the modulated signal is applied to the [mskShaping.vi] subVI. Here, the signal is multiplied with the sine function whose half cycle equals the bit duration, in order to produce the integrand of equation (T.52).

Next, the product created in [mskShaping.vi] is fed to [integrateAndDump.vi], whose output is then passed to [comparator.vi] to produce the binary sequence.

Case 6. MSK (type I)

First, in a manner similar to OQPSK, the offset added on the transmitter side is removed by [bitOffsetRemover.vi]. Following this, both in-phase and quadrature channels are multiplied with a sine waveform, whose period is equal to four bit durations. This is done as it was in case of the Sunde's BFSK using the [mskShaping.vi] subVI to produce the integrand of equation (T.52). Next, [integrateAndDump.vi] performs the coefficient detection in accordance to (T.52). Decisions on whether bit 0 or a 1 was transmitted are made by twin [comparator.vi] subVIs, and the resulting arrays are interleaved.

Case 7. MSK (type II)

For this scheme, detection method is almost identical to what is mentioned above for MSK type I; the only difference being the fact that in [mskShaping.vi] an absolute value operator is used to rectify the sine waveform.

Finalizing the Detection

Continuing, bit sequences obtained at the end of each case are sent as inputs to [binaryArrayToString.vi], where they are converted back into string format to be displayed to the user as a proof of the fact that the transmission has ended. In case the noise functionality was enabled, the user can then investigate how the message has been affected, as the modulated wave became subject to channel noise.

Looking at the block diagram of frame 5, one can notice a second case structure labelled "USRP path". This case structure processes the in-phase and quadrature components that were received using the USRP. Since USRP only outputs the baseband components of the received band-pass signal, there is no need for a complex analyzer. To detect the bits sent over the loopback cable, all cases in this case-structure are duplicates of the previously described methods used for bit detection from the simulated modulated waveform. This is allowed since the correlator-receiver has the dual functionality of being both a filter and a line code detection device as was mentioned in Section 2.4.

4.1.6 Frame 6. Determining the Bit Error Rate

Frame 6 contains a single [bitErrorRate.vi] subVI. Its function is to keep track of how many bits were received erroneously. This is accomplished by giving this VI access to two message strings, that are the transmitted one and the one received at the end of frame 5.

Inside the [bitErrorRate.vi] subVI both message strings are converted back into binary sequences, which are subsequently compared to one another, on a bit by bit basis.

4.1.7 Frame 7. Graphing

Code for graphing of select waveforms is placed in this frame. Such concentration of all the graphing functionality of the demo into a single frame greatly reduces clutter and helps the VI stay more organized.

A few graphing subVIs can be seen in this frame. Of these [magnitudeSpectrumAverage.vi] and [displayComponentsUnitCircle_Mk2.vi] are worth mentioning.

The former provides a more informative frequency spectrum of the modulated signal, if transmission of random words (as opposed to user defined words) was enabled. By averaging the magnitudes over multiple random transmissions, displayed frequency spectrum is closer to theoretical smooth curve, often shown in literature in comparison to the some simulated curve.

The latter plots the baseband components of the modulated band-pass signal coming out of frame 4 on a unit circle. The front panel of this subVI opens automatically when the top-level VI is opened.

4.1.8 Outside the Sequence Structure

Outside the flat sequence structure one can see a few groupings of LabVIEW code. This code governs the execution of some of the implemented front panel functionality, as well as other functionality necessary to ensure proper execution of the demonstration tool.

GUI functionality implemented to increase user-friendliness

4.2 Incorporating the USRP

The LabVIEW environment in the host computer is used to generate the in-phase and quadrature components of the signal that is to be transmitted. This means that the digitally modulated signal can be delivered to the USRP in one of the two ways, which warrant different implementation approaches:

1. The band-pass modulated signal for a given digital modulation scheme can be generated in the LabVIEW software environment and be assigned to either the in-phase or quadrature channel of the USRP, while leaving the other channel vacant. The modulation in LabVIEW would be done with a carrier of an intermediate frequency. In this approach, the USRP would play the role of a frequency up-converter.
2. The band-pass signal is not generated in the LabVIEW environment, and instead the baseband components expected by the USRP are formed and then sent for transmission to the USRP. In this case, the signal sent over the loopback cable would be generated by frequency translation of the complex baseband representation of the modulated signal to the carrier frequency.

Both options have their advantages and disadvantages. In the first case, it would be advantageous to have the entire modulated waveform go through the loopback cable, because the effects of the channel on the modulated waveform would be clearly evident upon reception. The major setback in using this approach, however, is the fact that this would require having to accommodate sampling rates of at least 185 kS/s in software. For lower end machines this would result in sub-par performance of the demonstration tool exemplified by slow response times in interactions with the GUI, freezing of the mouse cursor, and could even lead to irresponsiveness and subsequent crashing of LabVIEW.

If the second approach is chosen, only the lower frequency baseband components of the modulated wave would be accessible at the output of the loopback cable. The obvious advantage in choosing this option is the possibility to choose lower values for sampling frequencies, however the modulated band-pass signal is then left unseen.

Since the final purpose of this demo is to be used by students who do not necessarily have access to more powerful computers, the second approach was found to be more appropriate.

4.2.1 USRP Communication Path

Communication over the USRP was realized to be independent of the simulated communication. The USPR is incorporated into the demo in a parallel rather than series manner.

Separate VIs control the transmitter and receiver sides, when sending data over the USRP. Transmission is managed by [usrpSend.vi] and reception by [usrpReceive.vi]. Both of these VIs are configured using the global variable [usrpGlobal.vi]. Configuration values in [usrpGlobal.vi] include the carrier frequency, the sampling frequency, the transmission and reception gains and USRP's IP address and antenna names.

In the demonstration tool, data is communicated to the USRP in the following manner. In [digitalModulation.vi] VI, simulated transmission path for all the modulation schemes includes the [complexSynthesizer.vi] block. Here, the in-phase and quadrature baseband components of the modulated wave are written to the [globalUSRP.vi] global variable. In turn, [usrpSend.vi] reads the values from this global VI and writes the data to the USRP after necessary configuration routines have been completed. This happens continuously and the data is written as long as the [usrpSend.vi] is running.

Similarly, data received from the USRP is accessed by reading the values that [usrpReceive.vi] has written to [globalUSRP.vi]

Having [globalUSRP.vi] as a figurative middleman, allows the entire USRP communication path to operate independently. Thus, should the main VI be completely replaced, no modification will be required to the subVIs pertaining to the USRP.

4.2.2 USRP Special Considerations

Using the USRP gives rise to two problems that are not encountered in the transmission which is simulated in software. First, the USRP receives a specified number of samples and presents these as an array of complex double values. The issues is that unless appropriate measures were taken, there is no indication of where the beginning of the waveform is. In other words, given a reception of any number number of samples, there no way of knowing which of the samples represent the first bit. Second, the USRP is not capable of coherent detection, and thus there is mutual leakage between the in-phase and quadrature channels. In what follows, the methods used to resolve these problems are going to be shown.

To mark the beginning of the waveform, before it is written to the USRP, the array of samples is modified. A single sample is added with an amplitude twice as much as that of any other sample. On the receiver side, built-in LabVIEW function block called "Array Max & Min" can be used to locate the marked sample. Subsequent samples are then rearranged in order to conform to the original sequence that was written to the USRP and the marked sample is removed.

The second issue manifests itself as follows. Consider the in-phase channel of a complex analyzer. With correct operation, i.e phase error $\phi_e = 0$, the output of the low-pass filter, here denoted as $v_0(t)$ should be:

$$v_0(t) = s_I(t)\cos(\phi_e) + s_Q(t)\sin(\phi_e) \quad \textbf{(I.1)}$$

$$v_0(t) = s_I(t)\cos(0) + s_Q(t)\sin(0) = s_I(t) \quad \textbf{(I.2)}$$

but with some non-zero phase error the correct value is attenuated by a factor equal to $\cos(\phi_e)$ and the quadrature component scaled by $\sin(\phi_e)$ is superimposed. This effect can be formulated mathematically as follows:

$$\tilde{s}'(t) = \tilde{s}(t)\exp(\phi_e), \quad \textbf{(I.3)}$$

where $\tilde{s}'(t)$ represents the erroneous complex envelope at the output of the USRP. It can clearly be seen that correction can be done by multiplying the erroneous complex envelope $\tilde{s}'(t)$ by $\exp(-\phi_e)$, however a method needs to be developed in order to determine ϕ_e.

The phase can be determined if it is a priori known that only the in-phase channel of the complex analyzer is non zero.

$$\tilde{s}(t) = s_I(t), \quad \textbf{(I.4)}$$

Then, with in-phase and quadrature components of $\tilde{s}'(t)$ denoted as $s_I'(t)$ and $s_Q'(t)$:

$$\tilde{s}'(t) = \tilde{s}(t)\exp(\phi_e) = s_I(t)\cos(\phi_e) + js_I(t)\sin(\phi_e) \quad \textbf{(I.5.1)}$$

$$s_I'(t) = s_I(t)\cos(\phi_e) \quad \textbf{(I.5.2)}$$

$$s_Q'(t) = s_I(t)\sin(\phi_e) \quad \textbf{(I.5.3)}$$

$$\frac{s_Q'(t)}{s_I'(t)} = \tan(\phi_e) \quad \textbf{(I.5.4)}$$

Subsequently, ϕ_e is defined as:

$$\phi_e = \tan^{-1}\left(\frac{s_Q'(t)}{s_I'(t)}\right) \quad \textbf{(I.6)}$$

To ensure correct operation of the USRP, this method is implemented as a handshake routine between the [usrpSend.vi] and [usrpReceive.vi]. The handshake routing takes place automatically and requires no user interaction. In what follows how the handshake is established will be described as it takes place step-by-step.

First, when [usrpSend.vi] is run, it checks a boolean variable 'bool ph error' in [globalUSRP.vi]. This boolean variable indicates whether the correction factor ϕ_e has been determined. If the boolean is true, the system is already synchronized and the data from [globalUSRP.vi] is sent after necessary manipulations such as inclusion of the marked sample to indicate the beginning of the waveform and amplitude normalization to avoid DSP overflows.

If the boolean is false, the handshake routine is initiated. A preformed waveform of 1000 samples is sent over the loopback cable by [usrpSend.vi]. No information samples are sent to the USRP as long as the handshake has not been finalized.

Next, to continue the handshake establishment, [usrpReceive.vi] should be launched and run. When usrpReceive.vi reads data from the loopback cable it check the same boolean variable 'bool ph error' in order to find out if the phase error has been determined.

Because of the fact that a constraint is placed on [usrpSend.vi] to not send any data samples if 'bool ph error' is false, a true value signals that the incoming samples are data samples, while a false value for 'bool ph error' tells that the samples read from the USRP are meant to finalize the handshake.

If 'bool ph error' is read as false, this results in a function call made to a subVI called [usrpFindPhaseError.vi], which determines the phase error as was prescribed by equation (I.some) through (I.some). After ϕ_e is found, its value is written to [globalUSRP.vi] and 'bool ph error' is toggled to become true. The handshake routine is hereby completed.

After this, [usrpSend.vi] is permitted to start sending the data samples to the the USRP, because 'bool ph error' has changed its value. These samples are received by [usrpReceive.vi], corrected in accordance to equation (I.3), and written to [usrpGlobal.vi] to be read from the main [digitalModulation.vi].

Chapter 5. Results and Discussion

A screenshot of the front panel of the finalized demonstration tool has been taken and can be observed in Figure A.

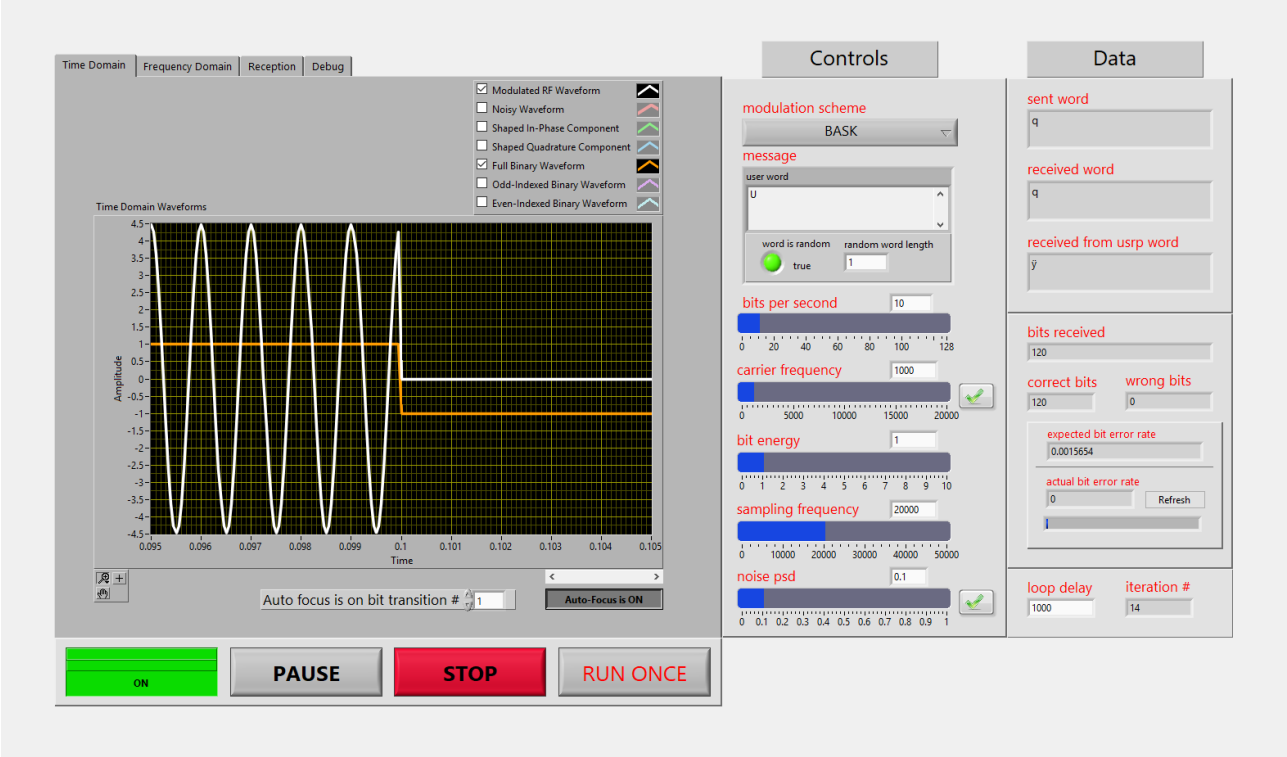


Figure 9: GUI of the demonstration tool

The user interface was organized into three main parts: graph tabs, controls and data.

Controls are the values that are used by the demonstration tool in order to make the implementation discussed in the previous chapter possible. Data are the values returned by the implemented system. Various signals generated in the demo are displayed as graphs.

5.1 Operation of the Demonstration Tool

Correctness of operation of the demo was judged by how accurate the graphs in both time and frequency domains are. In what follows, performance of the demo using these criteria will be discussed.

5.1.2 Time Domain View

In this section, the performance of the demo with regards to correctly displaying the time domain representations of signals in various places of the communication system will be assessed. Due to the fact that all of the realized modulation techniques were digital in nature, special attention was paid to whether or not the time domain graphs of the modulated waves were in agreement with theory at the bit switching instances.

The time waveform view is most useful for the BASK modulation technique, where the shape of the modulated wave changes significantly from bit to bit. Looking at the waveform, it is obvious whether at any given time a logical zero or a logical one is being signaled, since for the former case, the envelope of the modulated wave is reduced to zero. A time domain graph displaying a BASK modulated band-pass signal can be observed in Figure 10.

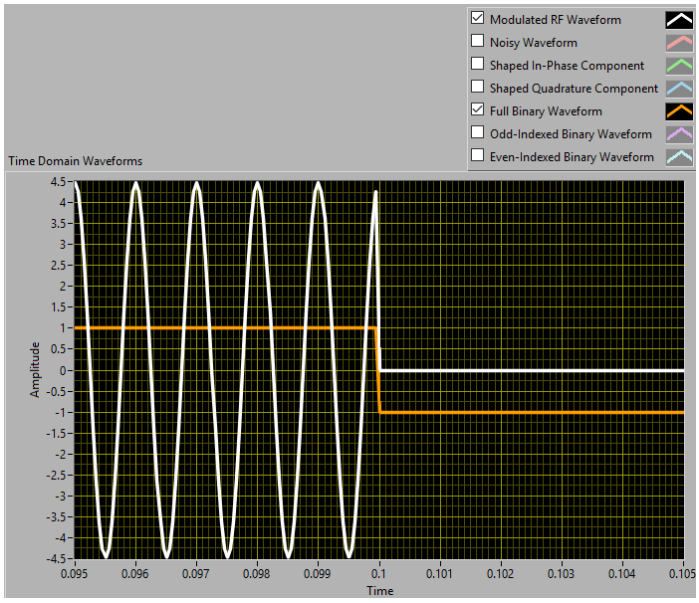


Figure 10: Time domain view of a BASK modulated signal

For other modulation techniques, the envelope of the band-pass signal is constant. Examining the time waveforms in their entirety does not provide any reasonable insight into how transmission of a certain binary symbol is reflected on the modulated signal. For this reason, dedicated controls were introduced on the front panel of the main VI, which focus the X-axis of the time waveform on a user-chosen bit transition. This allows the user to clearly observe that for example in case of BPSK, at the instance when a different bit is emitted from the source of binary data, the modulated wave undergoes a discontinuity of 180 degrees.

A screenshot of the time waveform graph showing the behavior of the simulated BPSK waveform when the binary symbol was switching from 1 to 0 is shown in Figure 11.

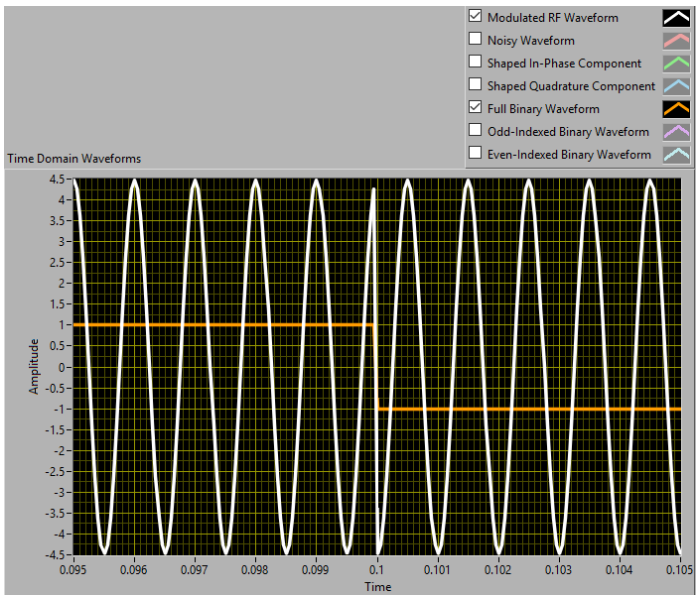


Figure 11: Time domain view of a BPSK modulated signal

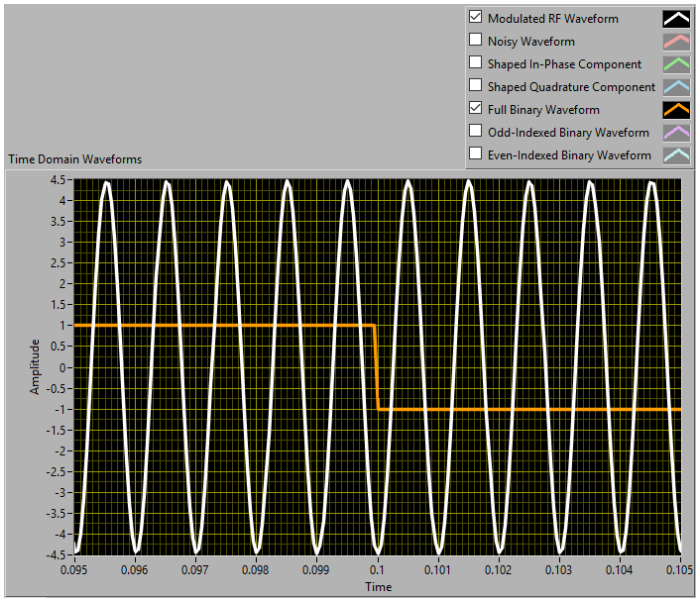


Figure 12: Time domain view of a Sunde's BFSK modulated signal

For frequency shift keying techniques like Sunde's BFSK and MSK, the band-pass signal should not experience any discontinuities. This is correctly shown in the demo and the behavior of the demo agrees with theory. Time waveforms for both SBFSK and MSK, exhibit no phase jumps when the bit changes. On Figure 12 it can be seen that for Sunde's BFSK, when the binary signal changes its value, there are no phase jumps in the modulated wave.

However, examining Figure 12 another issue can be identified. This has to do with the fact that both to the left and to the right of the switching moment, the modulated waveform seemingly looks the same. Since for the waveform on Figure 12, the carrier frequency is 10 kHz, and the bit rate is 10 bits per second, the difference in frequencies between transmitting a logical one and a logical zero is merely 10 Hz. It is clearly impossible for a user to distinguish between frequencies of 9990 Hz and 10010 Hz. For this reason, another special control was implemented, which changes the carrier frequency from whatever was chosen by the user to 40 Hz to make the waveforms corresponding to bits 0 and 1 more distinguishable. The control is realized in the form of a boolean button. In the GUI of the demo it can be seen as the green check-mark next to the carrier frequency slider.

In addition to the waveform for the modulated wave, GUI controls of the demo can be used to plot a variety of other relevant waveforms. For example, the in-phase and baseband components of the modulated wave can be chosen to be displayed alongside the RF waveform to better understand its formation. Using the plot visibility checkboxes that can be found on the top right-hand corner of the graph, the various waveforms can be made visible.

After examination of the time domain representations for all the implemented modulation schemes, the demo's performance was found to be accurate and satisfactory.

5.1.2 Frequency Domain View

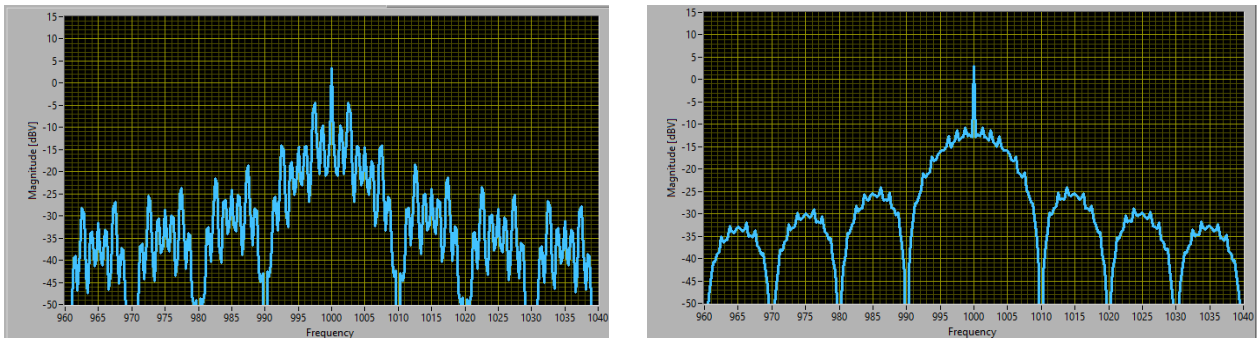
The graphs present in the frequency domain tab were assessed on how accurately they portray the frequency spectrum of the modulated wave. Spectra for other signals in the communication path are rarely studied in the course book. Because of that, it was decided to limit the drawn spectra to only include the modulated wave and the binary wave.

Given the fact that in the course book, for various modulation techniques, both the theoretical and the experimental spectrum curves have been depicted, it was decided to include a functionality to allow the the demo to imitate the theoretical curve. This was made possible via the realization that in case the spectrum is averaged over multiple random binary message transmissions, the experimental curve will begin to smooth out. In part (a) of Figure 13 spectrum obtained from transmission of single random three-character sequence is shown, while in part (b) the spectrum has been averaged over 300 transmissions. It can clearly be seen that the smoothed out curve is much better suited for educational purposes.

It was checked whether the bandwidth of the modulated signal is an agreement with the theory presented in the book. Furthermore, it was investigated if the overall shape of the spectrum around the carrier frequency is the same as theory predicts.

For example, it was confirmed that the spectrum of a BPSK modulated is signal is smooth around the carrier frequency and its bandwidth is indeed equal to $2/T_b$. For Sunde's BFSK it was confirmed that the spectrum of the modulated wave had two peaks centered at the carrier frequency with a spacing of $1/T_b$ Hz. Plots showing this can be found in Figure 14 for BPSK and Figure 15 for Sunde's BFSK.

For all the modulation techniques realized in the demo, no inaccuracies have been found in plots of the spectra.



(a) (b)

Figure 13: Spectrum of the BASK modulated signal for:
(a) transmission of a single random binary sequence;
(b) averaged over 300 transmissions of random binary sequences

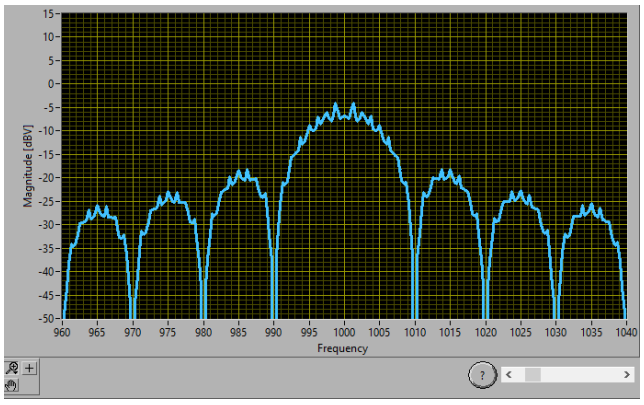


Figure 14: Averaged spectrum of BPSK modulated signal with bit duration of 0.1s.

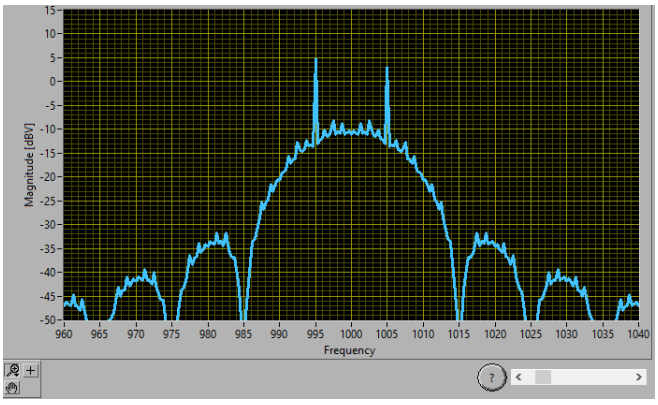


Figure 15: Averaged spectrum of SBFSK modulated signal with bit duration of 0.1s.

5.2 Noise Implementation

Added channel noise was implemented in the demo. It is specified by the user in terms of variance of the random variable that is superposed to each sample of the modulated signal. In Figure 16, the time waveform of a noisy QPSK modulated signal with noise variance equal to 4 can be observed. For the same signal, the spectrum has been plotted in Figure 17, where presence of noise can be observed by noticing a considerable noise floor.

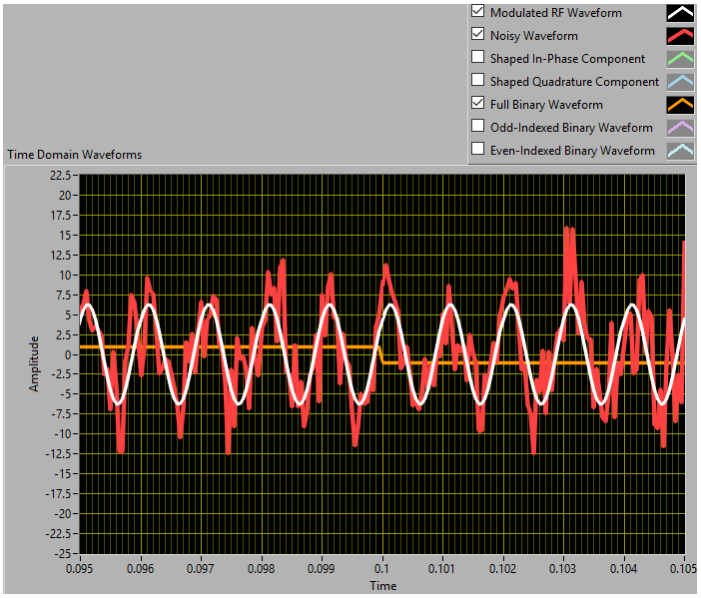


Figure 16: Time-domain view of a QPSK modulated signal with added Gaussian distributed noise with variance equal to 4.

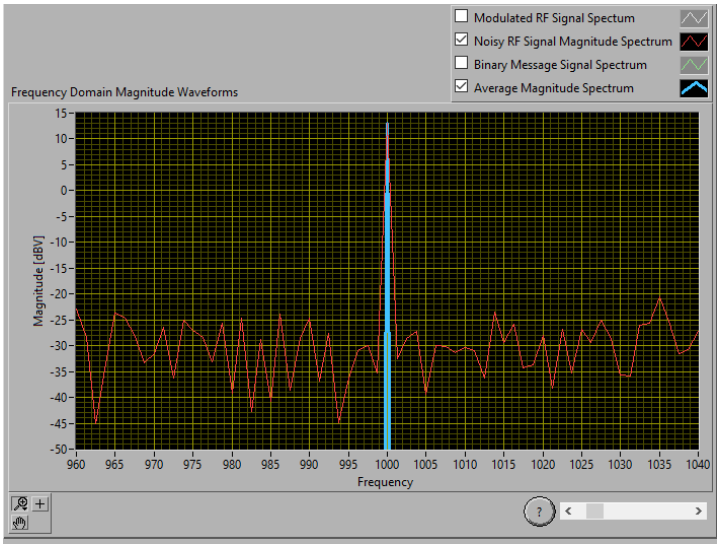


Figure 17: Frequency-domain view of a QPSK modulated signal with added Gaussian distributed noise with variance equal to 4.

Due to the time constraints, a full analysis of the effect of the channel noise was not carried out. A display was implemented that shows the expected error bit error rate given a certain value for the noise variance, according to the theory presented in the in Chapter 10 of the course book [1].

The main obstacle preventing the inclusion of the noise analysis into the scope of this work was stemming out from the discrete-time nature of the demo. Should the noise analysis be completed, then it would have had to be done using the noise theory pertinent to discrete-time noise.

5.3 Discussion about the GUI

In designing of the graphical user interface particular attention was paid in order to make it as intuitive as possible. As a result, all labels for controls and indicators on the front panel of the main VI were chosen to coincide with the terminology used in the course book. Controls were designed in the form of sliders to imply to the user, what the recommended minimum and maximum values for each control are.

Also, as time permitted, functionality of each VI was explained in properties of the VI. This would result in a context help window showing the description of a custom made VI. An example of this is shown on Figure 18.

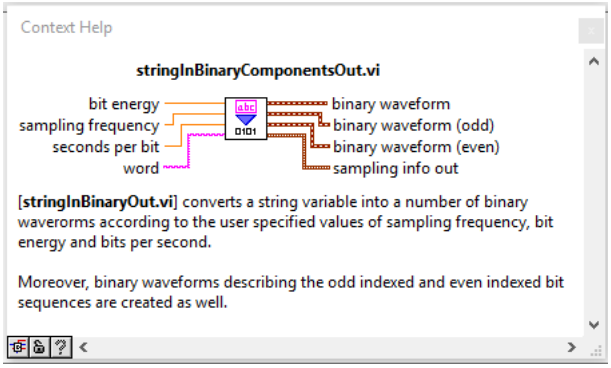


Figure 18: Context help window showing information about a custom-made VI.

In addition, properties of controls and indicators on the front panel, with functionality that is not straightforward or requires more explanation have been modified in order to display tip strips, when the mouse cursor is hovered on them. For example, the green check mark button on the front panel of [digitalModulation.vi] has been modified in this way and the result is shown in Figure 19.

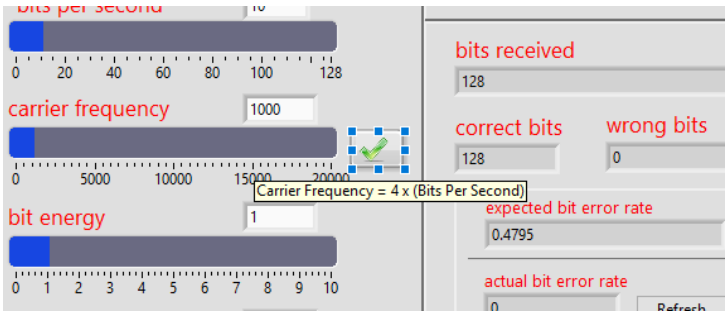


Figure 19: A tip strip displaying more information.

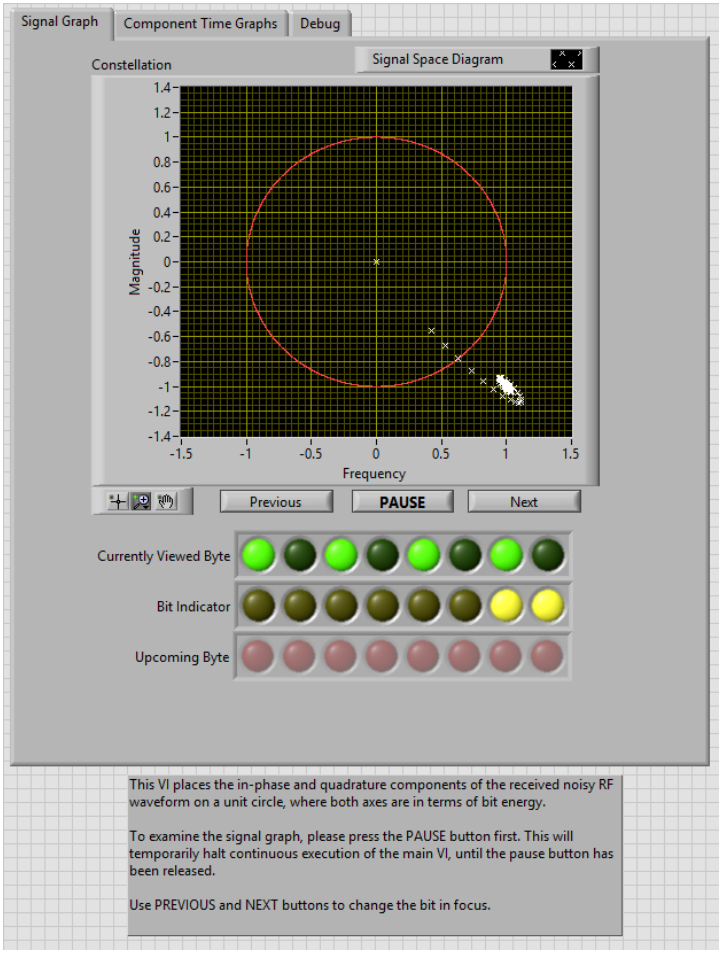


Figure 20: Front panel of [displayComponentsUnitCircle_Mk2.vi]

One of the limitations of the GUI, at the time of completion of this work, was the fact that it was not possible to have the front panel of [displayComponentsUnitCircle_Mk2.vi] be included on the front panel of [digitalModulation.vi]. The front panel of this VI, that places the baseband components of the modulated wave on a unit circle is shown in Figure 20. Possible methods to achieve that have been investigated. It was found that do accomplish that much more time would needed to be invested into studying LabVIEW environment and its more complicated paradigms.

An advantage of keeping the GUI simple is that the user manual created to help students accustom themselves to the demo can be kept short.

5.4 Discussion about the USRP

USRP was implemented to be a part of the demo via the auxiliary VIs. This implementation allows the USRP to be disconnected form the host computer at any moment and not cause any errors to the main VI. In addition, the chosen realization for the USRP allows the the students to fully utilize the demo outside of the laboratory environment.

The performance of phase error correction mechanism of the USRP was found to be highly satisfactory. Correction is done in a fraction of a second without paying close attention is almost impossible to notice.

However, it should be stated that the claim that using the USRP can help the students better understand the concepts given in the course book was found to be highly arguable. Its utilization does not provide any valuable information that the simulated communication path does not by itself.

Transmission of an MSK modulated signal through the loopback cable was tested and the resulting phase-corrected in-phase and quadrature components have been plotted in Figure 21.

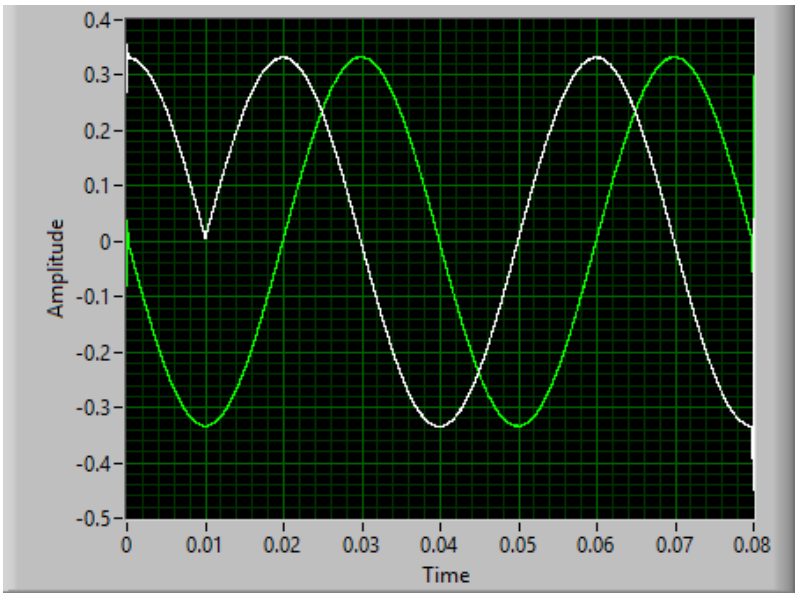


Figure 21: Baseband components of an MSK modulated signal received from the USRP

Also, the baseband components of the same, but entirely simulated MSK modulated wave were obtained. These are plotted in Figure 22.

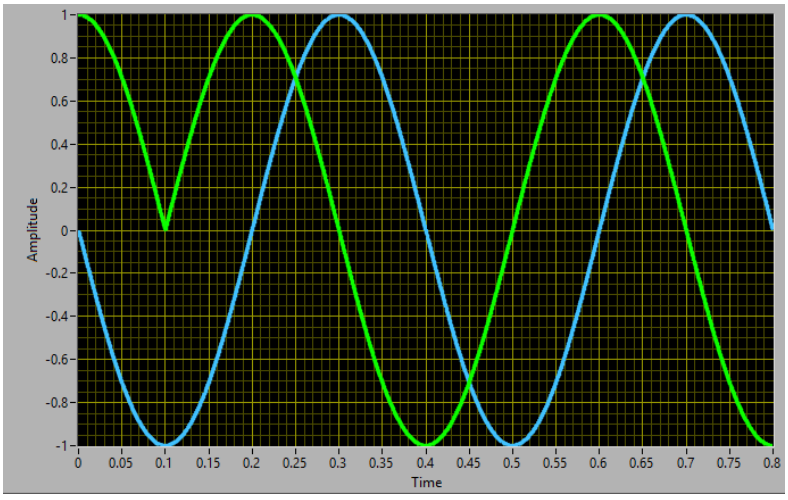


Figure 22: Baseband components of a simulated MSK modulated signal

As it can be seen, having the signal travel an actual communication medium did not produce any observable differences.

Chapter 6. Conclusions

6.1 Conclusions

In the course of this bachelor assignment a demo has been created to serve as a helping tool for students following the Communications course. For this reason, the demo is highly optimized to be used in unison with the course materials. The demo covers the fundamentals of digital modulation and provides experimental capabilities to compliment the course material.

The finalized demo is suitable for both classroom and personal use. This was made possible by parallel rather than serial integration of the USRP into the demo.

Implementation of the demo was carried out using the LabVIEW software and accompanying USRP hardware. Using the LabVIEW software the demo was realized into a single top-level VI and 35 subVIs performing various functions.

Noise was implemented in the demo in a limited manner, with its effects not being studied, due to discrete-time nature of the simulated noise, making it a rather involved subject, together with the time restrictions of a B.Sc. assignment.

Although at the outset of this assignment the emphasis was placed on the usage of the USRP, a multitude of experiments have shown limited benefits of its addition into the demo. Signals received from the USRP did not differ from those obtained entirely from simulation.

The infrastructure created to realize the USRP communication and the main VI with its respective subVIs have been designed to operate independently from one another with a LabVIEW global variable linking the two together. VIs created to implement the communication over the USRP can be used with any similar demonstration tool created in the future. To ensure compatibility of the newly created demos and the USRP infrastructure only a limited number of requirements have to be met.

6.2 Future Work

Given the educational purposes, in the future it is suggested to expand the demo by expanding the simulated path of communication rather than stronger USRP incorporation. With a strong framework of the present demo it is possible to use various function blocks that can operate in a standalone fashion. There is a high degree of reusability to enable expandability with relative ease. Possibilities for expansion of the demo include:

- In addition to the rectangular pulse shaping used in this demo, more pulse shaping techniques can be demonstrated.
- Furthermore, it is suggested to invest effort into proper simulation of noise to allow better understanding of the effects of the channel noise.

Finally, in the context of the Communications course, it is strongly suggested to completely eliminate the USRP from future demonstration tools, given its limited educational value, high operational costs and expensiveness to procure.

List of References

[1] Simon Haykin, Michael Moher, Introduction to Analog and Digital Communications, Second Edition, Wiley 2007

[2] Dativa Tizikara, An FM Radio Principles Demo Using a PC based software Radio Peripheral, 2014

[3] Rodger Zimmer, Willam Tranter, Principles of Communications: Systems, Modulation and Noise, 7th Edition, 2014

[4] Dayan Adionel Guimares, Contributions to the Understanding of the MSK Modulation, Revista Telecomunicacoes, vol. 11, no. 01, 2008