UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

GALILEO SIGNAL GENERATION
-SIMULATION ANALYSIS-

By Roger Canalda Pedrós
Student ID: 0833118

Supervisor: Dr. Máirtín O'Droma

Department of Computer and Electronic Engineering
University of Limerick
April 2009

GALILEO SIGNAL GENERATION

-SIMULATION ANALYSIS-

Abstract

This work presents the navigation signals and their allocation in the radio frequency band used in the new European Global Navigation Satellite System (GNSS): ***Galileo****. All signals are described mathematically and then simulated using* ***Matlab*** *language (in transmission). Results are shown thus proving the theory provided by the European Space Agency document: "Open Service Signal in Space Interface Control Document" (OS SIS ICD). Before introducing the navigation signals, a deep study and analysis on the modulations and multiplexing schemes (Binary Phase Shift Keying (BPSK), Binary Offset Carrier (BOC), Alternate Binary Offset Carrier (AltBOC), Carrier Adaptative Sub-Carrier Modulation (CASM) and Double Binary Offset Carrier (DBOC) is carried out, also giving detailed results and simulations in Matlab proving the theory again. Moreover, the acquirement of both the autocorrelation function (ACF) and the power spectral density (PSD) of all navigation signals (in Matlab) and all modulations (in theory) is emphasised in the study. Ranging codes are also studied in this work, however from a more practical than scientific point of view, showing how they have been implemented and what type of codes they are (memory codes, based register codes, gold codes). Finally, a paper is provided reviewing the innovative aspects of the navigation signals and why these have been selected for this new GNSS.*

Table of Contents

1. Document Overview

The present work is organised as follows:

- **Point 1** is this introduction which provides the scope of the document and an overview of the Galileo system.
- **Point 2** is an analysis of the different modulations that are used in Galileo: Binary Phase Shift Keying (BPSK), Binary Offset Carrier (BOC) and Coherent Adaptative Sub-carrier Modulation (CASM). The first two of them are studied and developed with their own autocorrelation functions (ACF) and power spectral density (PSD). In particular, it is studied the case of a BOCsin (BOC with a sine sub-carrier) and its parity depending on the parameter '*n*' : number of half-periods within a symbol (chip). Moreover, it is studied the shape of a BOC signal and the reasons for why the spectrum is like it is. It provides plots related with PSD and ACF wherein ideal behaviour can be appreciated. It also gives a detailed analysis about the theoretical Alternate BOC (AltBOC) modulation, introducing the modified AltBOC used in Galileo's E5 signal. Finally, CASM modulation is studied to see how the constant envelope issue is achieved in all three bands (E5, E6 and E1-L1). At the end of this section, Double BOC (DBOC) modulation is also presented with the aim of easing the nomenclature of BOC signals and unify them, also giving its reasons to be regarded as a possible modulation for Galileo.
- **Point 3** has to do with the ranging codes. It explains how the Code Division Multiple Access (CDMA) codification is done. This is one of the parts that more time has been invested in terms of Matlab. Codification having to do with signal E5 is the most difficult to achieve since it was via Linear Feedback Shift Register (LFSR) and the behaviour of this device had to be programmed. Regarding other signals, their codification has been assumed, in part, due to a lack of information in [1]. Afterwards, we can find the assignation for each signal and how the ranging codes have been programmed in appendix A.
- **Point 4** is the generation of the three navigation signals E5, E6 and E1-L1 with Matlab. For each signal, both the autocorrelation and the spectrum are represented. Moreover, it is explained what multiplexing techniques have been used, making special emphasis in signal E5, since it is the most complex to implement (and understand) due to its multiplexing technique: AltBOC. This signal is decomposed in its real and imaginary part and plotted to understand why the resulting spectrum has the shape it has. Regarding the other signals,

their implementation can be assumed from point 2, since the explanation of the spectrum of a BOC is shown in there. It also analyses the constant envelope behaviour of all of the signals by giving a plot of their constellations and figures like cumulative distribution function (CDF) and complementary cumulative distribution function (CCDF). Finally, it is added a plot of the Galileo band containing all signals studied in this document.

- **Point 5** (*written in a personal style*) focuses on the Matlab issues arisen while doing the simulations and providing all the plots in this project. It explains the procedures that have been followed to implement the navigation signals, ranging codes and other simulations needed for the understanding of the different parts. Add to this, it also reports the problems that have appeared during the programming and the solutions taken.
- **Point 6** tells about what points of this project could be investigated further to see if there is a better, new or different approach to do the same, meeting similar or better engineering requirements. It also provides not only different points of view of what could be studied regarding GNSS and compare it to GALIELO, but also possible extensions of the current work.

1.2 Introduction to Galileo Satellite System

Galileo is the European global navigation satellite system providing a highly accurate, guaranteed and global positioning service under civilian control. It is inter-operable with Global Positioning System (GPS) and GLONASS, the two other current global satellite navigation systems.

The fully deployed Galileo system consist of 30 satellites (27 operational and 3 spares), positioned in three circular Medium Earth Orbit planes at a nominal average orbit semi-major axis of 29601.297 Km, and at an inclination of the orbital planes of 56 degrees with reference to the equatorial plane [1].

The satellites are controlled and commanded from the Galileo Ground Control Segment (GCS) via its S-Band ground stations. The Galileo navigation service is achieved via transmission from each satellite of signals in L-Band comprising ranging codes and timing information, with the timing information included as part of a more general navigation message containing additional information relating to the satellite itself, the overall constellation and the integrity of the service. The Galileo Ground Mission Segment (GMS) determines the navigation, timing and integrity data part of the navigation messages and transmits it to the Satellite via its C-Band ground stations. The satellite constellation will be controlled from the GCS facilities installed in the GCC, and supported

by a worldwide network of Telemetry and Telecommand (TT&C) S-band stations [13].

The radio-frequency air interface between the space and user segments is composed of three independent CDMA signals, named E5, E6 and E1-L1, and they are permanently transmitted by all Galileo satellites. The E5 signal is further sub-divided into two signals denoted as E5a and E5b.

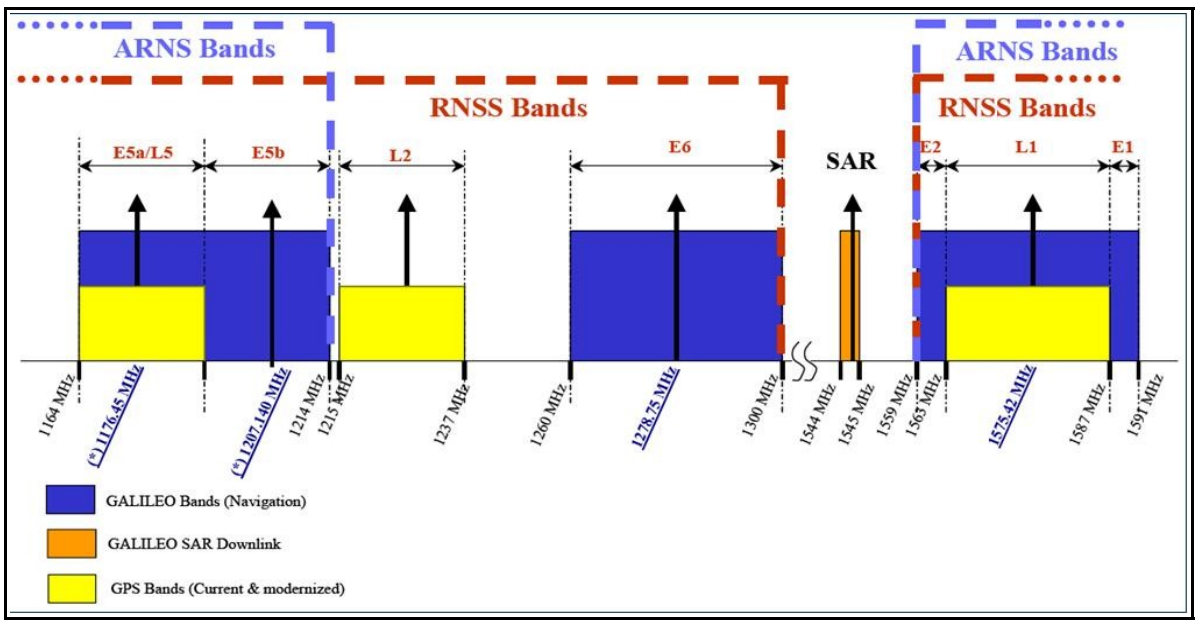


Figure 1: Galileo and GPS band

The Galileo system will provide a range of services to users. The following services will be provided worldwide and independently from other systems by Galileo:

- the **Open Service (OS)** combines the open signals, free-of-charge to users, to provide position and timing performance competitive with other satellite systems. A single frequency OS will be provided by each of the three signals: E1, E5a and E5b. Dual frequency OS will be provided by the following dual-frequency combinations: E1(B&C)-E5a and E1(B&C)-E5b [5];

- the **Safety-of-Life (SoL)** Service improves on the OS by providing timely warnings to the user when it fails to meet certain margins of accuracy or continuity (integrity). It is envisaged that a service guarantee will be provided. The mono-frequency SoL will be provided by each of the two signals E1(B&C) and E5b. The dual-frequency SoL will be provided by the following dual-frequency signal combination: E1(B&C)-E5b [5];

- the **Commercial Service (CS)** provides access to two additional signals for faster data throughput and increased accuracy. The signals are encrypted. It is envisaged that a service guarantee will be provided. This service will be provided by the E6(B&C) signals plus the OS signals (E1(B&C), E5a and E5b). The E6(B&C) signal contains the value-added data and is combined with OS signals for improved performance [5];

- the **Public Regulated Service (PRS)** provides position and timing to government-authorised users requiring a high continuity of service with controlled access. The PRS will be provided by E1-A and E6-A signals. These will use encrypted PRS ranging codes, navigation data messages and sub-carriers improving signal processing performances [5];

- the Galileo support to the **Search and Rescue (SAR)** service is Europe's contribution to the international COSPAS-SARSAT system. Galileo will play an important part in the MEO Search & Rescue system (MEOSAR). Galileo satellites will be able to pick up signals from emergency beacons carried by ships, planes and individuals, and ultimately relay them to national rescue centres. A rescue centre can thus know the precise location of an accident. At least one Galileo satellite will be in view of any point on Earth at all times, so real-time distress alerts will be possible. The SAR distress messages will be detected by the Galileo satellites in the 406-406.1 MHz and then broadcast to the dedicated receiving ground stations in the 1544-1545 MHz band, callel L6. The data will be embedded in the OS data of the signal transmitted in the E1 carrier frequency [5];

Each Galileo satellite will broadcast 10 different navigation signals, making it possible for Galileo to offer the open (OS), safety-of-life (SOL), commercial (CS) and public regulated (PRS) services. Galileo will have an integrity signal to ensure the quality of the signals received. Galileo signals will offer a guaranteed accuracy down to 1 metre, with value-added services achieving a real-time accuracy of 10 cm. The frequencies used by the satellites are within the 1.1-1.6 GHz band, a range of frequencies particularly well-suited to mobile navigation and communication services [12].

In *figure 2* we can see the different signals used in Galileo. They are composed by both in-phase and quadrature components. The first band we find is the E5, and it is subdivided into two subbands called E5a and E5b. These signals are modulated and multiplexed by AltBOC(15,10) technique.

The next band in question is E6, and the signals in that band are modulated by a BPSK(5) and a BOCcos(10,5). Finally, there is the L1 band, which can be subdivided as well in E2-L1-E1 bands. In here, the modulations are CBOC(6,1,1/11) and BOCcos(15,2.5).

All along this work, every signal will be studied and represented in Matlab as it should be seen in transmission from the satellite. Before, there will be a study about the modulations used in this satellite system, BPSK and BOC (with its variants), to understand why these modulations, and no others, have been chosen for such an important project.

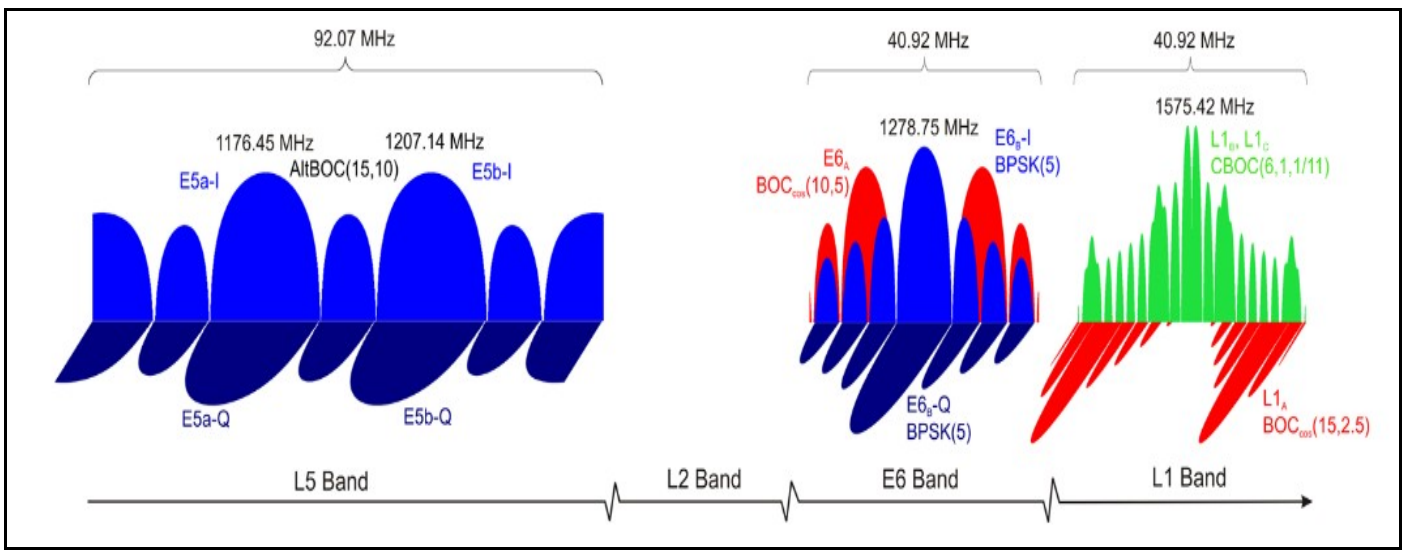


Figure 2: Galileo signals

2. Modulations in the Galileo Satellite System

2.1 BPSK modulation

The BPSK modulation (Binary Phase Shift Keying) is characterized for having only two levels (binary) of symbols {+1,-1} for bit 0 and bit 1 respectively. Its general expression in passband can be written like:

$$S(t)=Ac\cdot\sum_{k=-\infty}^{+\infty}\cos(2\pi f t+\phi_c+\theta_{i,k})\cdot p(t-kT)$$

where:

- Ac is the amplitude.
- Φ_c is the offset phase.
- $\theta_{i,k}$ is the binary symbol {± 1}.
- $p(t)$ is the shaping pulse of duration T.
- k index numerates symbols along time. It is taken from minus infinity to plus infinity as it is a theoretical form.
- i index numerates what symbol is taken from the binary alphabet {± 1} depending on i = 0,1.

If $S(t)$ is developed by "*cos(a + b) = cos(a) cos(b) – sin(a) sin(b)*" , it can be written as in-phase ($i_s(t)$) and quadrature ($q_s(t)$) components:

$$S(t)=Ac\cdot\underbrace{\sum_{k=-\infty}^{+\infty}\cos(\theta_k)\cdot p(t-kT)}_{i_s(t)}\cos(2\pi f t+\phi_c)-Ac\cdot\underbrace{\sum_{k=-\infty}^{+\infty}\sin(\theta_k)\cdot p(t-kT)}_{q_s(t)}\sin(2\pi f t+\phi_c)$$

So, once the I&Q components are given, it can be expressed by the baseband format by:

$$b_s(t)=i_s(t)+jq_s(t)$$

thus, it can be set like:

$$b_s(t) = Ac \cdot \sum_{k=-\infty}^{+\infty} e^{j\theta_k} p(t-kT) \quad ,$$

$$\text{since} \quad e^{j\theta_k} = \cos(\theta_k) + j \cdot \sin(\theta_k)$$

Hence, the autocorrelation function (ACF) of the baseband form can be obtained from this last expression more easily (*demonstrated at the end of this section*):

$$R_{b_s b_s}(t) = \frac{|Ac|^2}{T} \sum_{m=-\infty}^{+\infty} R_{\theta\theta}(m) R_p(t-mT)$$

where:

- $R_{\theta\theta}(m)$ is the ACF of the baseband modulated symbols, $e^{j\theta_k}$.
- $R_p(t)$ is the ACF of the shaping pulse $p(t)$.
- Index m numerates the symbols along time.

By applying the Fourier transform, the power spectral density (PSD) function is achieved:

$$F\{R_{b_s b_s}(t)\} = F\left\{\frac{|Ac|^2}{T} \sum_{m=-\infty}^{+\infty} R_{\theta\theta}(m) R_p(t-mT)\right\} = \frac{|Ac|^2}{T} \sum_{m=-\infty}^{+\infty} R_{\theta\theta}(m) \cdot F\{R_p(t-mT)\}$$

only the factor depending on time is transformed:

$$F\{R_p(t-mT)\} = F\{R_p(t) * \delta(t-mT)\} = F\{p(t) * \check{p}(-t) * \delta(t-mT)\} = P(f) \cdot P(-f) \cdot e^{-2\pi fmT}$$

Where $\check{p}(t)$ is the conjugate and $P(f)$ is the Fourier transform of $p(t)$ (which is a rectangular pulse of duration T):

$$F\{p(t)\} = F\{rect(\frac{t}{T})\} = T \cdot sinc(fT) = P(f)$$

Note that as $p(t)$ is a real rectangular pulse, the conjugate does not affect it. Moreover, as $P(f)$ is an even function ($sinc(x)$), $P(-f) = P(f)$. Hence, $P(f) \cdot P(f) = |P(f)|^2$, and this factor can be extracted from the summation because it does not depend on m.

$$S_{b_s}(f)=\frac{|Ac|^2}{T}|P(f)|^2\sum_{m=-\infty}^{+\infty}R_{\theta\theta}(m)\cdot e^{(-j2\pi fmT)}$$

Following this, the autocorrelation of the baseband modulated symbols can be seen to be the sum of the covariance and the square average of the symbols as follows:

$$R_{\theta\theta}(m)=C_{\theta\theta}(m)+|m_\theta|^2$$

To examine the impact the statistics have on the PSD, the equation can be reorganized like:

$$S_{b_s}(f)=\frac{|Ac|^2|P(f)|^2}{T}\cdot\sum_{m=-\infty}^{+\infty}C_{\theta\theta}(m)e^{-j2\pi fmT}\quad+\quad\frac{|Ac|^2\cdot|m_\theta|^2}{T^2}\cdot\sum_{m=-\infty}^{+\infty}\left|P(\frac{m}{T})\right|^2\delta(f-\frac{m}{T})$$

So, it is clear that the ACF and the PSD depend not only on the shaping pulse, *p(t)*, but also on the symbols' statistics. It can also be seen that there will be “carriers” within the spectrum, however these carriers do not provide any information, so there turns out to be inefficiency in the use of the power needed, in case the signal is transmitted and the mean is not zero.

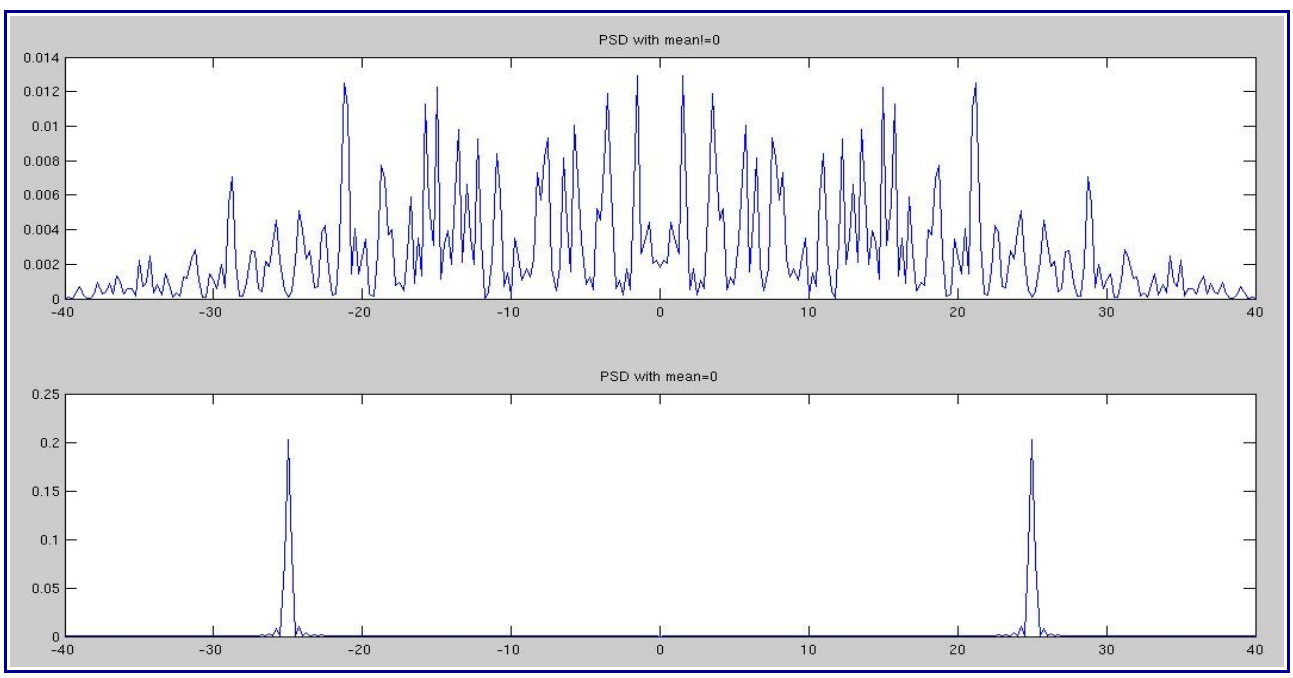


Figure 3: PSD with different mean values (0 and 1)

As shown in figure 3, we have the plot of a BPSK PSD with two different symbols. These symbols are square waves with bipolar amplitude {+1,-1} and with a duration of 40 ms. The first ones are with mean not equal to zero and the second ones with mean equal to zero. As it can be appreciated,

in the second plot, the two deltas at frequency of 25 hertzs are achieved sharply. Conversely, if the mean is not zero, there exist different sub-carriers that waste the power of the spectrum.

Nevertheless, if symbols are statistically independent and with mean value $m_\theta = 0$, and since:

$$C_{\theta\theta}(m) = \sigma_\theta^2 \cdot \delta[m]$$

then

$$S_{b_s}(f) = \frac{|Ac|^2 \, \sigma_\theta^2 \, |P(f)|^2}{T}$$

And once the PSD in baseband is obtained, it is easier to get the bandpass PSD by applying the following:

$$S_s^{PSK}(f) = \frac{1}{4}\left[S_{b_s}(f - fc) + S_{b_s}(f + fc)\right] = \frac{|Ac|^2\sigma^2}{4\mathrm{T}} \cdot \left[|P(f - fc)|^2 + |P(f + fc)|^2\right]$$

where:

$$\sigma^2 = E\left\{\left|e^{j\theta_k}\right|^2\right\} - E^2\left\{e^{j\theta_k}\right\} = 1 - 0$$

$$E\left\{e^{j\theta_k}\right\} = \int_{-\infty}^{\infty} pdf(\theta_k)\, e^{j\theta_k} d\theta_k = \int_0^{2\pi} \frac{1}{2\pi} \cdot e^{j\theta_k} d\theta_k = 0$$

$$E\left\{\left|e^{j\theta_k}\right|^2\right\} = \int_{-\infty}^{\infty} pdf(\theta_k)\left|e^{j\theta_k}\right|^2 d\theta_k = \int_0^{2\pi} \frac{1}{2\pi} \cdot 1 \, d\theta_k = 1$$

with symbols distributed uniformly and E is the expected value [8]:

The expected value of the random variable X is defined $E(X) = \int_{-\infty}^{+\infty} x \cdot f_x(x)\, dx$ and it is usually denoted by m_x. Where $f_x(x)$ is the density function of X.

2.1.2 Demonstrations

To demonstrate how the ACF of a BPSK is achieved, we start with the baseband expression:

$$b_s(t) = Ac \cdot \sum_{k=-\infty}^{+\infty} e^{j\theta_k} p(t - kT) \quad \rightarrow$$

$$R_{b_s b_s}(t+\tau, t) = E\left\{ b_s(t+\tau) \cdot \check{b}_s(t) \right\} \quad where \;\; \check{x} \;\; means\, conjugate \;\; \rightarrow$$

$$E\left\{ \sum_{k=-\infty}^{\infty} Ac \cdot e^{j\theta_k} \cdot p(t+\tau-kT) \cdot \sum_{l=-\infty}^{\infty} \check{A}c \cdot e^{-j\theta_l} \cdot \check{p}(t-lT) \right\} =$$

$$= |Ac|^2 \cdot \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} E\left\{ e^{j\theta_k} \cdot e^{-j\theta_l} \right\} \cdot p(t+\tau-kT) \cdot \check{p}(t-lT) =$$

$$= |Ac|^2 \cdot \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} R_{\theta\theta}(k-l) \cdot p(t+\tau-kT)\, \check{p}(t-lT)$$

$$letting \quad k-l = m, \;\; then$$

$$R_{b_s b_s}(t+\tau, t) = |Ac|^2 \cdot \sum_{m=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} R_{\theta\theta}(m) \cdot p(t+\tau-mT-lT) \cdot \check{p}(t-lT)$$

The definition of $k - l = m$ breaks the symbol cross-dependency.

At this point, it can be seen that the ACF is a cyclostationary process[1], since the signal depends periodically (and with the same period *T*) on time. Therefore, the Wiener-Kinchine theorem will be applied to obtain the time-average autocorrelation function so that the PSD can be calculated, which is the important aspect [8].

If *x(t)* is cyclostationary, from a corollary of the Wiener-Kinchine theorem, we can write the time-average ACF like:

$$\overline{R_{b_s b_s}(t+\tau, t)} = \lim_{k \to +\infty} \frac{k}{kT_0} \int_{\langle T_0 \rangle} R_{b_s b_s}(t+\tau, t)\, dt = \frac{1}{T_0} \int_{\langle T_0 \rangle} R_{b_s b_s}(t+\tau, t)\, dt$$

so, it leads us to:

1 A random process x(t) with mean $m_x(t)$ and autocorrelation function $R_x(t+\tau,t)$ is called cyclostationary if both the mean and the autocorrelation are periodic in t with some period T_0, i.e., if $m_x(t+kT_0)=m_x(t)$ and $R_x(t+\tau+kT_0, t+kT_0)=R_x(t+\tau,t)$ for all t, τ and k.

$$\overline{R_{b_s b_s}(t+\tau,t)}=\frac{1}{T}\int_{-T/2}^{T/2}|Ac|^2\cdot\sum_{m=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty}R_{\theta\theta}(m)\cdot p(t+\tau-(m+l)T)\cdot\check{p}(t-lT)dt=$$

$$letting\quad t-lT=\lambda\ ;\ dt=d\lambda,$$

$$=\frac{|Ac|^2}{T}\sum_{m=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty}R_{\theta\theta}(m)\int_{-T/2-lT}^{T/2-lT}p(\lambda+\tau-mT)\cdot\check{p}(\lambda)d\lambda=$$

$$=\frac{|Ac|^2}{T}\sum_{m=-\infty}^{+\infty}R_{\theta\theta}(m)\underbrace{\sum_{l=-\infty}^{+\infty}\int_{-T/2-lT}^{T/2-lT}}_{\int_{-\infty}^{\infty}}p(\lambda+\tau-mT)\cdot\check{p}(\lambda)d\lambda=$$

$$=\frac{|Ac|^2}{T}\sum_{m=-\infty}^{+\infty}R_{\theta\theta}(m)\int_{-\infty}^{\infty}p(\lambda+\tau-mT)\cdot\check{p}(\lambda)d\lambda=$$

$$letting\quad R_{xx}(\alpha)=\int_{-\infty}^{\infty}x(\lambda+\alpha)\check{x}(\lambda)d\lambda$$

$$=\frac{|Ac|^2}{T}\sum_{m=-\infty}^{+\infty}R_{\theta\theta}(m)R_p(\tau-mT)=R_{b_s b_s}(\tau)$$

2.2 BOC modulation

The Binary Offset Carrier (BOC) was designed to modernize the military Global Positioning System (GPS) service because it was needed to add another signal within the radio frequency bands that were already being used. At a first attempt, BOC modulations were developed to provide spectral isolation from heritage signals modulating the same carrier frequency, it was quickly determined that it also had lots of advantages, such as their constant-modulus envelope and binary phase, fact that makes them straightforward to implement.

Moreover, by moving signal power away from the band centre, they offer the potential for better code-tracking accuracy and multipath rejection. Since BOC modulations offer two independent design parameters, *fs* and *fc*, they provide freedom to concentrate signal power within specific parts of the allocated band to reduce interference with the reception of other signals. Furthermore, **the redundancy** in the upper and low sidebands of BOC modulations offers practical advantages in receiver processing for signal acquisition, code tracking, carrier tracking, and data demodulation.

Nonetheless, there are also some inconveniences. For instance, if the sub-carrier frequency is much greater than the spreading code rate, the correlation function has many closely spaced peaks that can introduce anomalous effects in code tracking. Also, wide spacing of sub-carriers can limit the spectral coherence of the received signal as a result of the dispersive effect of hardware imperfections and other channel conditions, such as ionospheric distortion [4].

2.2.1 Spectral Analysis of BOC

The following development has been initiated from [4].

The expression of a baseband offset carrier signal is given by:

$$s(t) = e^{-j\theta} \cdot \sum_{k=-\infty}^{+\infty} a_k \mu_{nTs}(t - knTs - t_0) c_{Ts}(t - t_0) \quad (1)$$

where:

• a_k is the data modulated spreading code. Unit magnitude with phase values chosen randomly from an alphabet.

• c_{Ts} is the sub-carrier. Periodic function with period $2Ts$.

• μ_{nTs} spreading symbol.

• n is the number of half periods of the sub-carrier during which the spreading code value remains the same.

• θ and t_0 reflect arbitrary offsets in phase and time.

In this representation it is proved that, when the spreading code is rectangular, this offset carrier signal is a conventional Phase Shift Keying-Rectangular (PSK-R) signal modulating a periodic signal.

Initial work on offset carrier modulations focused on a sinusoidal sub-carrier and a lowpass-filtered spreading symbol. The resulting complex envelope was not constant modulus. This class of offset carrier modulation has been termed *linear offset carrier*, since its modulus takes on a continuum of values. Further considerations lead to use square-waved sub-carrier C_{Ts} and a rectangular (unfiltered) spreading symbol μ_{nTs}, producing a constant modulus complex envelope. The resulting binary-valued modulation is called **binary offset carrier**.

A BOC modulation is denoted by *BOC(fs,fc)*, where *fs* is the sub-carrier frequency and *fc* is the code rate:

$$f_s = \frac{1}{2\,Ts} \qquad f_c = \frac{1}{nTs} = \frac{2}{n} f_s$$

Both *fs* and *fc* will be multiples of 1.023 MHz and Mcps respectively.

Expression (1) can be written in a different way depending on the parity of *n*. When *n* is even it is like this:

$$s_{BOC(f_s,f_c)}(t) = e^{-j\theta} \cdot \sum_{k=-\infty}^{+\infty} a_k q_{nTs}(t - knTs - t_0) \tag{2}$$

and when *n* is odd:

$$s_{BOC(f_s,f_c)}(t) = e^{-j\theta} \cdot \sum_{k=-\infty}^{+\infty} a_k (-1)^k q_{nTs}(t - knTs - t_0) \tag{3}$$

with spreading symbol:

$$q_{nTs}(t) = \sum_{m=0}^{n-1} (-1)^m \mu_{Ts}(t - mTs) \tag{4}$$

Where this last summation is the *n* half-cycles of a square wave: *n* alternating ± 1 values within a (chip) duration (splitting it).

For *n* even, q_{nTs} has an average value of zero (balanced symbol). When *n* = 1 and using a rectangular no-return to zero (NRZ) spreading symbol, yields a BOC(fc/2,fc) having the same second-order parameters (ACF and PSD) as a conventional PSK-R. However, the waveforms of a BOC (fc/2,fc) are not the same because of the -1 when *n* is odd.

Assuming that the binary values of the BOC spreading sequence are equally likely, independent,

and identically distributed, the normalised baseband PSD of a BOC modulation is:

$$S_{BOC_{(f_s,f_c)}}(f) = \frac{1}{nTs}\left(\frac{\sin(\pi fTs)\sin(\pi fnTs)}{\pi f\cos(\pi fTs)}\right)^2 \quad \text{for } n \text{ even} \qquad (5)$$

$$S_{BOC_{(f_s,f_c)}}(f) = \frac{1}{nTs}\left(\frac{\sin(\pi fTs)\cos(\pi fnTs)}{\pi f\cos(\pi fTs)}\right)^2 \quad \text{for } n \text{ odd} \qquad (6)$$

Note that when $n = 1$ (odd), the equation turns into:

$$S_{BOC_{(f_s,f_c)}}(f) = \frac{1}{nTs}\left(\frac{\sin(\pi fTs)\cos(\pi fTs)}{\pi f\cos(\pi fTs)}\right)^2 = \frac{1}{Ts}\left(\frac{\sin(\pi fTs)}{\pi f}\right)^2 = Ts\ sinc^2(fTs) = \frac{1}{f_c}sinc^2(fTs)$$

which is the conventional expression for the power spectral density of a PSK-R modulation, as seen above.

In the PSD, the sum of mainlobes and sidelobes between the mainlobes is equal to n, twice the ratio of the sub-carrier frequency to the code rate. As in conventional PSK, the zero crossing of each lobe are spaced by twice the code rate, while the zero crossings of each sidelobe are spaced at the code rate. Maxima of the mainlobes occur at frequencies somewhat below the sub-carrier frequency because of the coherent interactions between the upper and lower sidebands [4].

To obtain the PSD function, we will start with the PSD form of a PSK modulation with independent, identically distributed spreading code and spreading symbol $p(t)$ with spreading period T:

$$S(f) = \frac{|P(f)|^2}{T}$$

Note that it is the same expression from section *2.1*, except with $Ac = 1$ and variance equal to the unit. Therefore, the PSD depends on the shaping pulse used.

If we get the BOC spreading symbol, $q_{nTs}(t)=\sum_{m=0}^{n-1}(-1)^m \mu_{Ts}(t-mTs)$, and $T = nTs$, its Fourier transform will be:

$$Q_{nTs}(f)=\sum_{m=0}^{n-1}(-1)^m F\{\mu_{Ts}(t-mTs)\} = \sum_{m=0}^{n-1}(-1)^m F\{rect(\frac{t-mTs}{Ts})\} \quad =$$

$$= \sum_{m=0}^{n-1}(-1)^m e^{-j2\pi fmTs}\frac{\sin(\pi fTs)}{\pi f}e^{-j\pi fTs} = \frac{e^{-j\pi fTs}\sin(\pi fTs)}{\pi f}\sum_{m=0}^{n-1}(-1)^m e^{-j2\pi fmTs} \qquad \textbf{(7)}$$

The factor $e^{-j\pi fTs}$ comes from the integral limits of the Fourier transform, $m \cdot Ts$ and $(m+1) \cdot Ts$, because only one period is integrated.

When this point has been reached, there is a *two-pronged* way of developing the Fourier transform of the spreading symbol q_{nTs} *depending on the parity of n.*

<u>*On one hand*</u>, **if *n* is even**, the summation $\sum_{m=0}^{n-1}(-1)^m e^{-j2\pi fmTs}$ yields:

$$1-e^{-j2\pi fTs}+e^{-j2\pi f2Ts}-e^{-j2\pi f3Ts}+\ldots+e^{-j2\pi f(n-2)Ts}-e^{-j2\pi f(n-1)Ts}$$

To demonstrate what this series can turn into in simpler terms, $n = 6$ will be used as an example. Then, the summation will give:

$$1-e^{-j2\pi fTs}+e^{-j2\pi f2Ts}-e^{-j2\pi f3Ts}+e^{-j2\pi f4Ts}-e^{-j2\pi f5Ts}=\ldots$$

$$\ldots=(1-e^{-j2\pi fTs})\cdot\left(1+\frac{e^{-j2\pi f\frac{5}{2}Ts}}{e^{-j\pi fTs}}\cdot\underbrace{\frac{e^{j2\pi f\frac{1}{2}Ts}-e^{-j2\pi f\frac{1}{2}Ts}}{e^{j\pi fTs}-e^{-j\pi fTs}}}_{1}\right)\cdot\left(1+\frac{e^{-j2\pi f\frac{9}{2}Ts}}{e^{-j\pi fTs}}\cdot\underbrace{\frac{e^{j2\pi f\frac{1}{2}Ts}-e^{-j2\pi f\frac{1}{2}Ts}}{e^{j\pi fTs}-e^{-j\pi fTs}}}_{1}\right)=\ldots$$

$$\ldots=(1-e^{-j2\pi fTs})\cdot(1-e^{-j2\pi f2Ts})\cdot(1-e^{-j2\pi f4Ts})$$

Therefore, n even will give in a general case:

$$\sum_{m=0}^{n-1}(-1)^m e^{-j2\pi\, fmTs} = (1-e^{-j2\pi\, fTs})\cdot\sum_{m=0}^{\frac{n}{2}-1} e^{-j2\pi\, f2mTs} \tag{8}$$

and if the subsequent geometric series is developed:

$$\sum_{m=0}^{\frac{n}{2}-1} e^{-j2\pi\, f2mTs} = \frac{1-e^{-j2\pi\, fnTs}}{1-e^{-j4\pi\, fTs}}$$

it is finally proved that:

$$\sum_{m=0}^{n-1}(-1)^m e^{-j2\pi\, fmTs} = (1-e^{-j2\pi\, fTs})\cdot\frac{1-e^{-j2\pi\, fnTs}}{1-e^{-j4\pi\, fTs}} \tag{9}$$

If in this expression common factors are extracted, it can be put in a quotient with sine and cosine functions which take part in the PSD general formula:

$$(1-e^{-j2\pi\, fTs})\cdot\frac{1-e^{-j2\pi\, fnTs}}{1-e^{-j4\pi\, fTs}} = e^{-j\pi\, fTs}\cdot(e^{j\pi\, fTs}-e^{-j\pi\, fTs})\cdot\frac{e^{-j\pi\, fnTs}}{e^{-2j\pi\, fTs}}\cdot\frac{e^{j\pi\, fnTs}-e^{-j\pi\, fnTs}}{e^{j2\pi\, fTs}-e^{-j2\pi\, fTs}} =$$

$$= 2j\cdot\sin(\pi\, fTs)\cdot\frac{\sin(\pi\, fnTs)}{\sin(2\pi\, fTs)}\cdot e^{-j\pi\, fTs}e^{-j\pi\, f(n-2)Ts} = \ldots$$

and using the identity "*sin(2x) = 2 sin(x) cos(x)*" in the denominator:

$$\ldots = j\cdot e^{-j\pi\, f(n-1)Ts}\cdot\frac{\sin(\pi\, fnTs)}{\cos(\pi\, fTs)}$$

Finally, remembering that the aim of this demonstration is the Fourier transform of the spreading symbol (4) **for *n* even**, it is obtained substituting this last simplification in (7):

$$Q_{nTs}(f) = e^{-j\pi\, fTs}\cdot\frac{\sin(\pi\, fTs)}{\pi\, f}\, j\cdot e^{-j\pi\, f(n-1)Ts}\cdot\frac{\sin(\pi\, fnTs)}{\cos(\pi\, fTs)} = j\cdot e^{-j\pi\, fnTs}\cdot\frac{\sin(\pi\, fTs)\cdot\sin(\pi\, fnTs)}{\pi\, f\cdot\cos(\pi\, fTs)} \tag{10}$$

Thus, knowing that $S(f) = \frac{|P(f)|^2}{T} = \frac{|P(f)|^2}{nTs}$ and in this case *P(f)* is $Q_{nTs}(f)$, if *P(f)* is substituted in this last expression for (10):

$$S_{BOC(f_s,\,f_c)}(f) = \frac{1}{nTs}\cdot\left(\frac{\sin(\pi\, fTs)\cdot\sin(n\pi\, fTs)}{\pi f\cdot\cos(\pi\, fTs)}\right)^2 \quad \text{with } f_c=(n\cdot Ts)^{-1} \tag{11}$$

<u>*On the other hand*</u>, **if *n* is odd**, the summation $\sum_{m=0}^{n-1}(-1)^m e^{-j2\pi\, fmTs}$ will give an even number of terms and it will be the same as the case above but with one more addend at the end. So, adapting equation (9) simplified, the summation will give:

$$\sum_{m=0}^{n-1}(-1)^m e^{-j2\pi\, fmTs} = j\cdot e^{-j\pi f(n-2)Ts}\cdot\frac{\sin(\pi f(n-1)Ts)}{\cos(\pi\, fTs)} + e^{-j2\pi f(n-1)Ts} \tag{12}$$

Note that the indexes have been changed as required. The appearance of this last term makes it more difficult to achieve the PSD expression when *n* is odd.

We begin by extracting common factor in (12) and giving the expression its real and imaginary parts except from the common factor:

$$e^{-j2\pi f(n-1)Ts}\cdot\left(j\cdot e^{j\pi\, fnTs}\cdot\frac{\sin(\pi f(n-1)Ts)}{\cos(\pi\, fTs)}+1\right) = e^{-j2\pi f(n-1)Ts}\cdot\left(j\cdot\cos(n\pi\, fTs)\cdot\frac{\sin(\pi f(n-1)Ts)}{\cos(\pi\, fTs)} - \right.$$
$$\left. -\sin(n\pi\, fTs)\cdot\frac{\sin(\pi f(n-1)Ts)}{\cos(\pi\, fTs)}+1\right) \tag{13}$$

By applying the identity "*sin(a-b) = sin(a) cos(b) – sin(b) cos(a)*" and treating the second addend within the brackets, it is obtained that:

$$-\left(\frac{\sin(n\pi\, fTs)}{\cos(\pi\, fTs)}\cdot(\sin(n\pi\, fTs)\cos(\pi\, fTs)-\sin(\pi\, fTs)\cos(n\pi\, fTS))\right)$$

then, if this last expression is operated together with the third addend "1" from (13):

$$\left(\frac{-\sin^2(n\pi fTs)\cos(\pi fTs)+\sin(n\pi fTs)\sin(\pi fTs)\cos(n\pi fTs)+\cos(\pi fTs)}{\cos(\pi fTs)}\right)$$

and extracting the common factor twice *cos(π f Ts)* and using "*1-sin²(n π f Ts) = cos²(n π f Ts)*":

$$\left(\frac{\cos^2(n\pi fTs)\cos(\pi fTs)+\sin(n\pi fTs)\sin(\pi fTs)\cos(n\pi fTs)}{\cos(\pi fTs)}\right)$$

$$\frac{\cos(n\pi fTs)}{\cos(\pi fTs)}\cdot\left(\cos(\pi fTs)\cos(n\pi fTs)+\sin(n\pi fTs)\sin(\pi fTs)\right)$$

The last term can be simplified by applying the identity "*cos (a-b)*" the other way round:

$$\cos(\pi fTs)\cos(n\pi fTs)+\sin(n\pi fTs)\sin(\pi fTs)=\cos(\pi f(n-1)Ts)$$

Now, remodulating equation (13) with simplified version of the last expression, the summation yields:

$$\sum_{m=0}^{n-1}(-1)^m e^{-j2\pi fmTs}=e^{-j2\pi f(n-1)Ts}\cdot\frac{\cos(\pi fnTs)}{\cos(\pi fTs)}\cdot\left[\cos(\pi f(n-1)Ts)+j\cdot\sin(\pi f(n-1)fTs)\right]=$$
$$=e^{-j2\pi f(n-1)Ts}\cdot e^{j\pi f(n-1)Ts}\frac{\cos(\pi fnTs)}{\cos(\pi fTs)}=e^{-j\pi f(n-1)Ts}\cdot\frac{\cos(\pi fnTs)}{\cos(\pi fTs)} \quad (14)$$

Finally, remembering that the aim of this demonstration is the Fourier transform of the spreading symbol (4) **for *n* odd**, it is obtained substituting (14) in (7):

$$Q_{nTs}(f)=e^{-j\pi fTs}\cdot\frac{\sin(\pi fTs)}{\pi f}\cdot e^{-j\pi f(n-1)Ts}\cdot\frac{\cos(\pi fnTs)}{\cos(\pi fTs)}=e^{-j\pi fnTs}\cdot\frac{\sin(\pi fTs)\cos(n\pi fTs)}{\pi f\ \cos(\pi fTs)} \quad (15)$$

Thus, knowing that $S(f)=\frac{|P(f)|^2}{T}=\frac{|P(f)|^2}{nTs}$ and in this case *P(f)* is $Q_{nTs}(f)$, if *P(f)* is substituted in this last expression for (15):

$$S_{BOC(f_s,f_c)}(f) = \frac{1}{nTs}\cdot\left(\frac{\sin(\pi\, fTs)\cdot\cos(n\pi\, fTs)}{\pi\, f\cdot\cos(\pi\, fTs)}\right)^2 \quad \text{with } f_c = (n\cdot Ts)^{-1} \tag{16}$$

At the end of this demonstration, some plots regarding Galileo signals' PSD will be shown, it will be also possible to discern the differences between a sine or cosine square sub-carrier.

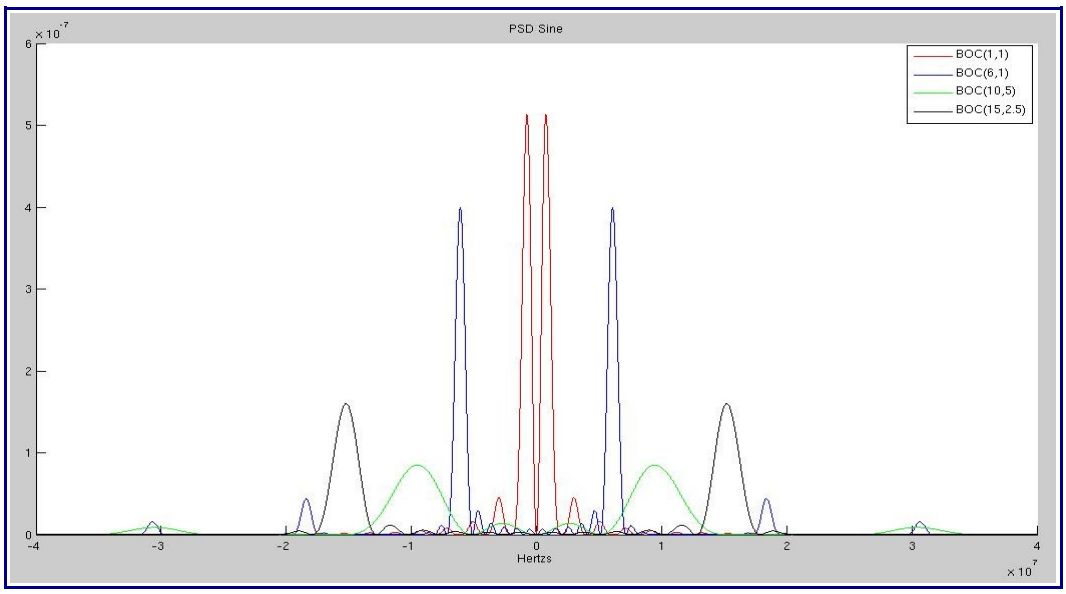


Figure 4: PSD with sine sub-carriers for all signals

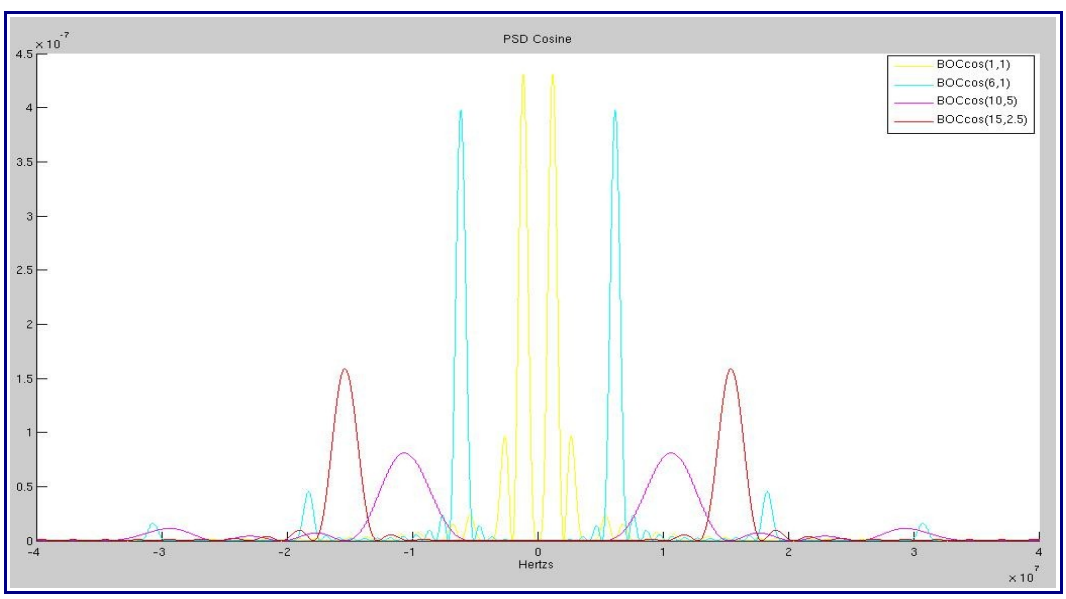


Figure 5: PSD with cosine sub-carriers for all signals

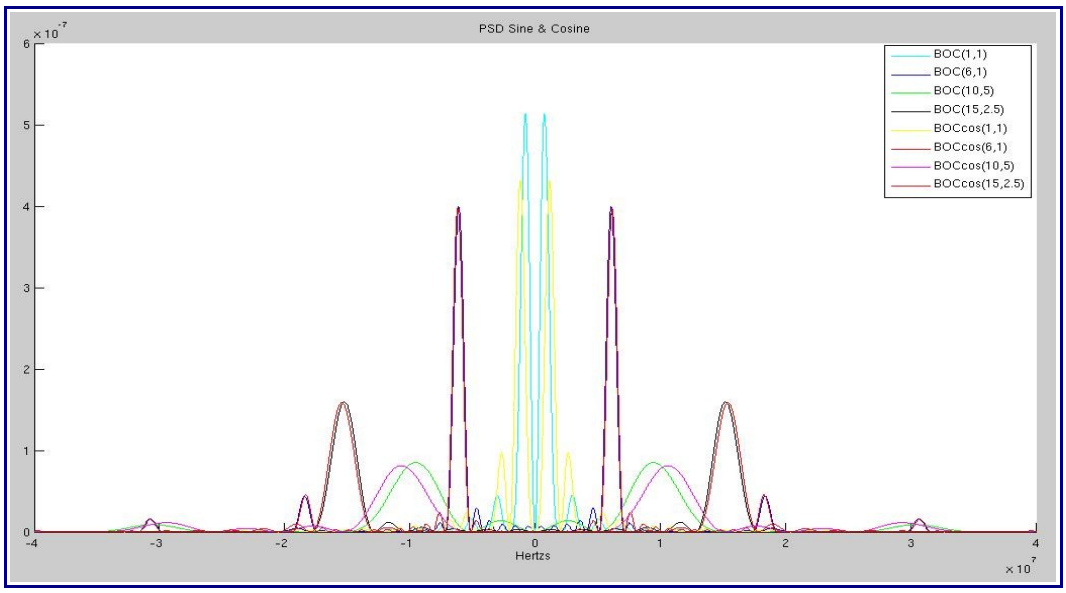


Figure 6: PSD with sine and cosine sub-carriers for all signals

All these plots have been obtained from the ideal formula for a BOCsin or BOCcos substituting the *fs* and *fc* values appropriately:

$$G_{BOCsin(fs,fc)}(f) = fc \cdot \left(\frac{\sin\left(\frac{\pi f}{2\text{fs}}\right)\sin\left(\frac{\pi f}{fc}\right)}{\pi f \cos\left(\frac{\pi f}{2\text{fs}}\right)} \right)^2$$

$$G_{BOCcos(fs,fc)}(f) = fc \cdot \left(\frac{2 \cdot \sin^2\left(\frac{\pi f}{4\text{fs}}\right)\sin\left(\frac{\pi f}{fc}\right)}{\pi f \cos\left(\frac{\pi f}{2\text{fs}}\right)} \right)^2$$

As it is shown in the plots, the PSD is split in two lobes, the frequency of the maximum PSD from a cosine sub-carrier is higher than that of the maximum PSD from a sine sub-carrier, the lobe attenuation of BOC(fs,fc) is slower than that of a BPSK-R(1), and the maximum PSD of a BOC signal is about 3 dB lower than that of a BPSK-R(1) [9]. Moreover, PSD with sine sub-carrier spread more power on mainlobes whereas PSD cosine sub-carriers spread more power in sidelobes.

Now, to get a better approach and to see from another point of view what a BOC modulation consists in, we will make the Fourier transform of a signal composed by data[2], CDMA encoding[3] and a square sub-carrier: $s(t)=d(t)\cdot c(t)\cdot sign(x(t))$, where *x(t)* indicates whether it is a sine or cosine sub-carrier.

Each of the three different components has its own purpose in this modulation. For instance, *d(t)* is the transmitted modulated data. The signal can also be dataless and become a pilot signal including only ranging code and sub-carrier. *c(t)* is the spreading code, which spreads the spectrum and gives to the modulation the characteristic features of a spread spectrum modulation. This factor is the one which influences the most when it comes to shape the spectrum. So, the higher the chip rate the wider the main lobe. Finally, we have the square sub-carrier. This way of modulating allows spectrum allocation in a specific part of a band. That is because the Fourier transform consists in deltas which are equally spaced *k/Ts* hertzs (for *k* = odd and *Ts* is the period of the sub-carrier).

In the next lines we will provide the mathematical explanation of the paragraph from above:

If we get a sine squared sub-carrier:

$$sc(t)=sign(\sin(2\pi f t))$$

with sub-carrier frequency *f*, it can be also expressed as a pair of summations:

$$sc(t)=\sum_{n=-\infty}^{+\infty} rect\left(\frac{t-nT-\frac{T}{4}}{T/2}\right)-\sum_{n=-\infty}^{+\infty} rect\left(\frac{t-nT-\frac{3T}{4}}{T/2}\right)=$$

$$=\left[rect\left(\frac{t}{T/2}\right)*\delta(t-T/4)*\sum_{n=-\infty}^{+\infty}\delta(t-nT)\right]-\left[rect\left(\frac{t}{T/2}\right)*\delta(t-3T/4)*\sum_{n=-\infty}^{+\infty}\delta(t-nT)\right]=$$

$$=\left[rect\left(\frac{t}{T/2}\right)*\delta(t-T/4)*\sum_{n=-\infty}^{+\infty}\delta(t-nT)\right]-\left[rect\left(\frac{t}{T/2}\right)*\delta(t-T/4)*\delta(t-T/2)\sum_{n=-\infty}^{+\infty}\delta(t-nT)\right]$$

2 It is formed by a random string of ±1 in this example. The data in the Galileo Satellite System depend on each signal and service provided.

3 These ranging codes are the ones used in signal E5 for the Galileo Satellite System, as it will be shown in section 4 (Signal Generation).

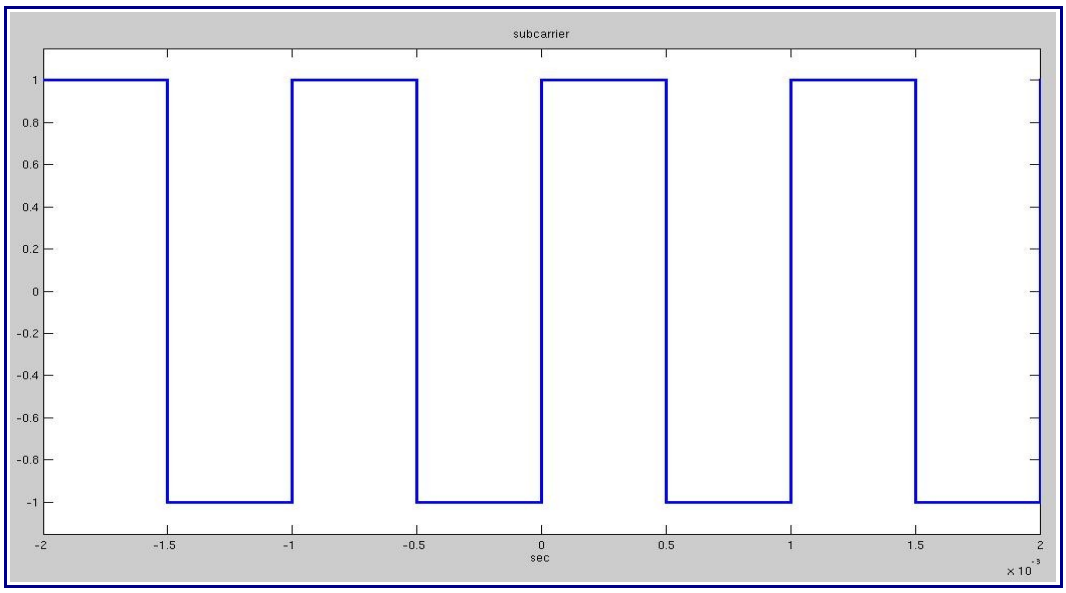


Figure 7: sc(t)=sign(sin(2π1000t)). We choose a period of 1 ms.

Now, if we apply the Fourier transform to this last expression:

$$SC(f)=\frac{T}{2}sinc(\frac{T}{2}f)e^{-j2\pi f\,T/4}\frac{1}{T}\sum_{m=-\infty}^{+\infty}\delta(f-\frac{m}{T})-\frac{T}{2}sinc(\frac{T}{2}f)e^{-j2\pi f\,T/4}e^{-j2\pi f\,T/2}\frac{1}{T}\sum_{m=-\infty}^{+\infty}\delta(f-\frac{m}{T})=$$

$$=\frac{1}{2}sinc(\frac{T}{2}f)e^{-j\pi fT/2}(1-e^{-j2\pi fT/2})\sum_{m=-\infty}^{+\infty}\delta(f-\frac{m}{T})=$$

$$=\frac{1}{2}\sum_{m=-\infty}^{+\infty}sinc(\frac{m}{T}\frac{T}{2})e^{-j\pi\frac{m}{T}\frac{T}{2}}(1-e^{-j2\pi\frac{m}{T}\frac{T}{2}})\delta(f-\frac{m}{T})=$$

$$=\frac{1}{2}\sum_{m=-\infty}^{+\infty}sinc(\frac{m}{2})e^{-j\frac{\pi}{2}m}(1-e^{-j\pi m})\delta(f-\frac{m}{T})=$$

$$=\frac{1}{2}\sum_{m=-\infty}^{+\infty}sinc(\frac{m}{2})e^{-j\frac{\pi}{2}m}(1-(-1)^m)\delta(f-\frac{m}{T})$$

In this point, if *m* takes even values, the transform is equal to zero, so there is no signal for even values. Therefore, there will be only spectrum for odd values: *m=2k+1*

$$SC(f)=\frac{1}{2}\sum_{k=-\infty}^{\infty} sinc\left(\frac{2k+1}{2}\right)e^{-j\frac{\pi}{2}(2k+1)}\,2\delta\left(f-\frac{2k+1}{T}\right);$$

$$since\quad e^{-j\frac{\pi}{2}(2k+1)}=e^{-j\frac{\pi}{2}2k}\,e^{-j\frac{\pi}{2}}=e^{-j\pi k}(-j)=(-1)^k(-j),$$

$$SC(f)=-j\sum_{k=-\infty}^{+\infty}(-1)^k\,sinc\left(\frac{2k+1}{2}\right)\delta\left(f-\frac{2k+1}{T}\right)$$

Where the Fourier transform of a train of deltas [14] (periodic signal) is:

$$\sum_{n=-\infty}^{+\infty}\delta(t-nT)\Leftrightarrow\frac{1}{T}\sum_{m=-\infty}^{+\infty}\delta(f-\frac{m}{T})$$

If now we apply the absolute value we obtain:

$$|SC(f)|=\left|\sum_{k=-\infty}^{+\infty}(-1)^k\,sinc\left(\frac{2k+1}{2}\right)\delta\left(f-\frac{2k+1}{T}\right)\right|$$

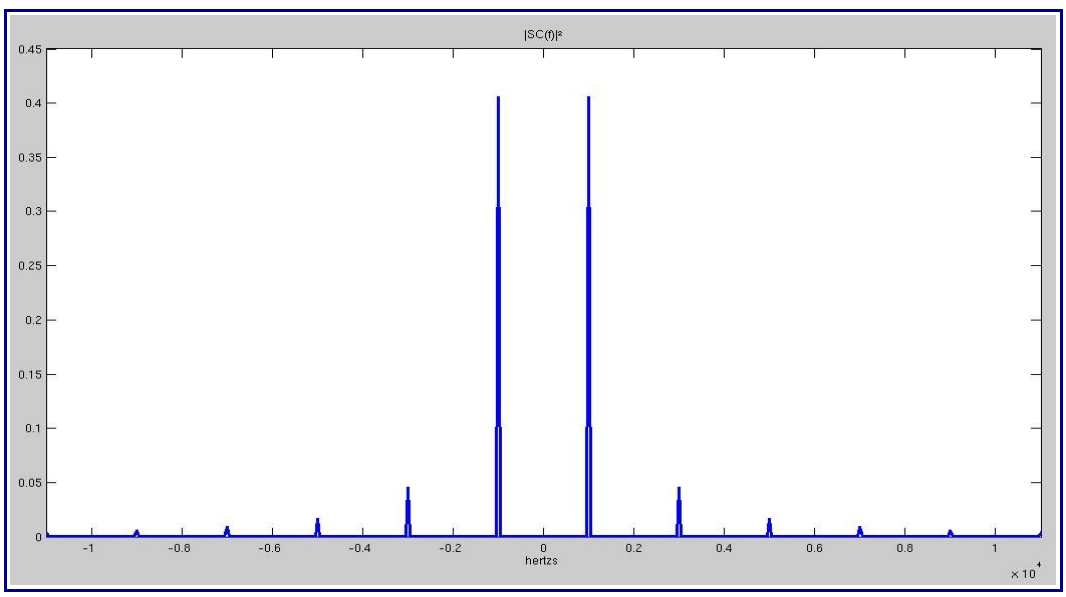


Figure 8: squared magnitude of FT{sign(sin(2π1000t))}

As it can be seen in the result, the Fourier transform of *sc(t)* will be a train of deltas weighted by the factors of a sinc function and only for odd values. That means that the deltas will be positioned every *k/T* hertzs as commented above, and the spectrum of the data plus CDMA encoding will fit in every delta of this sub-carrier, where the main ones are the first two in *±1/T*.

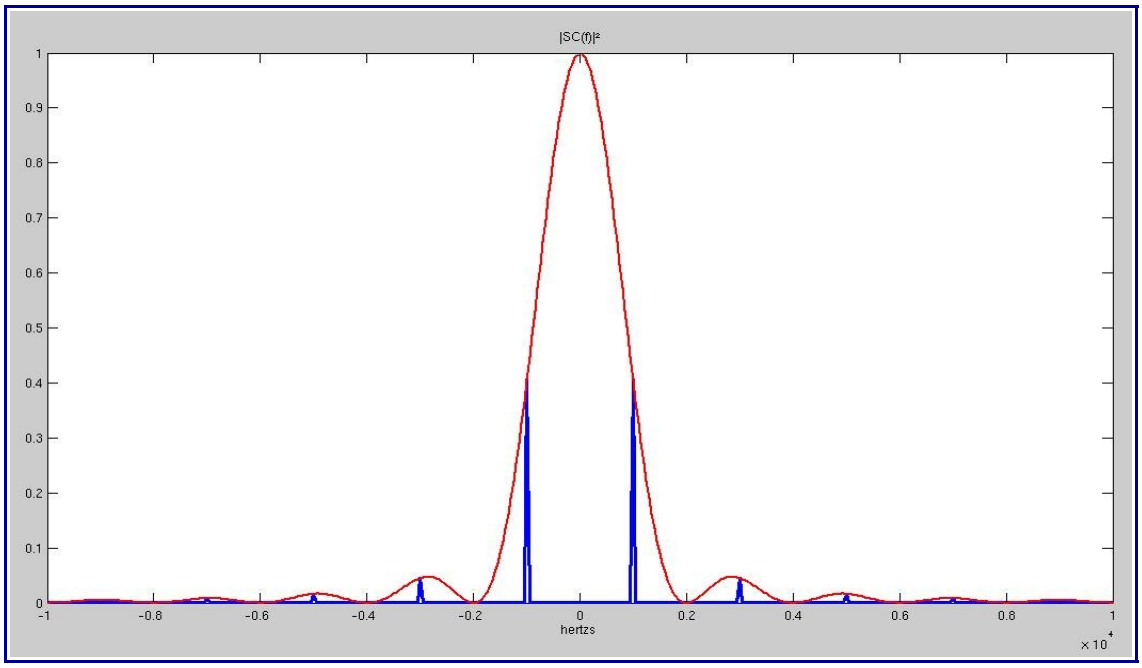


Figure 9: squared magnitude of FT{sc(t)} and sinc(0.0005 f)

If we opt for a cosine sub-carrier, the transformation is easier than that of a sine:

$$sc(t) = sign(\cos(2\pi f t))$$

with sub-carrier frequency *f*, it can be also expressed as a pair of summations:

$$sc(t) = \sum_{n=-\infty}^{+\infty} rect\left(\frac{t-nT}{T/2}\right) - \sum_{n=-\infty}^{+\infty} rect\left(\frac{t-nT-\frac{T}{2}}{T/2}\right) =$$

$$= rect\left(\frac{t}{T/2}\right) * \sum_{m=-\infty}^{+\infty} \delta(t-nT) - rect\left(\frac{t}{T/2}\right) * \delta(t-T/2) * \sum_{m=-\infty}^{+\infty} \delta(t-nT)$$

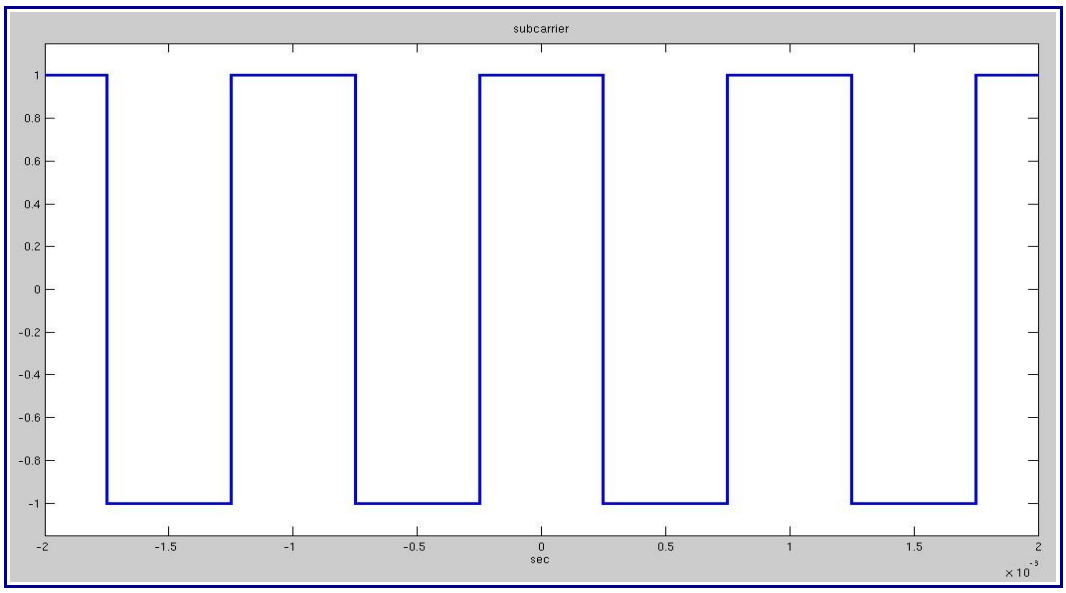


Figure 10: sc(t)=sign(cos(2π1000t))

Now, if we apply the Fourier transform to this last expression:

$$SC(f)=\frac{T}{2}sinc\left(\frac{T}{2}f\right)\frac{1}{T}\sum_{m=-\infty}^{+\infty}\delta(f-\frac{m}{T})-\frac{T}{2}sinc\left(\frac{T}{2}f\right)e^{-j2\pi f\frac{T}{2}}\frac{1}{T}\sum_{m=-\infty}^{+\infty}\delta(f-\frac{m}{T})=$$

$$=\frac{1}{2}sinc\left(\frac{T}{2}f\right)(1-e^{-j\pi fT})\sum_{m=-\infty}^{+\infty}\delta(f-\frac{m}{T})=$$

$$=\frac{1}{2}\sum_{m=-\infty}^{+\infty}sinc\left(\frac{T}{2}\frac{m}{T}\right)(1-e^{-j\pi\frac{m}{T}T})\,\delta(f-\frac{m}{T});$$

$$since\quad e^{-j\pi m}=(-1)^m,$$

$$SC(f)=\frac{1}{2}\sum_{m=-\infty}^{+\infty}sinc\left(\frac{m}{2}\right)(1-(-1^m))\,\delta(f-\frac{m}{T});$$

Again, if *m* takes even values, the Fourier transform is zero and we have no spectrum. Therefore, *m* will only take odd values, *m=2k+1*.

$$SC(f)=\frac{1}{2}\sum_{k=-\infty}^{+\infty} sinc\left(\frac{2k+1}{2}\right)2\delta\left(f-\frac{2k+1}{T}\right)=\sum_{k=-\infty}^{+\infty} sinc\left(\frac{2k+1}{2}\right)\delta\left(f-\frac{2k+1}{T}\right)$$

and applying the absolute value:

$$|SC(f)|=\left|\sum_{k=-\infty}^{+\infty} sinc\left(\frac{2k+1}{2}\right)\delta\left(f-\frac{2k+1}{T}\right)\right|$$

So, the result is the same than that of a sine sub-carrier, provided that the absolute value is used.

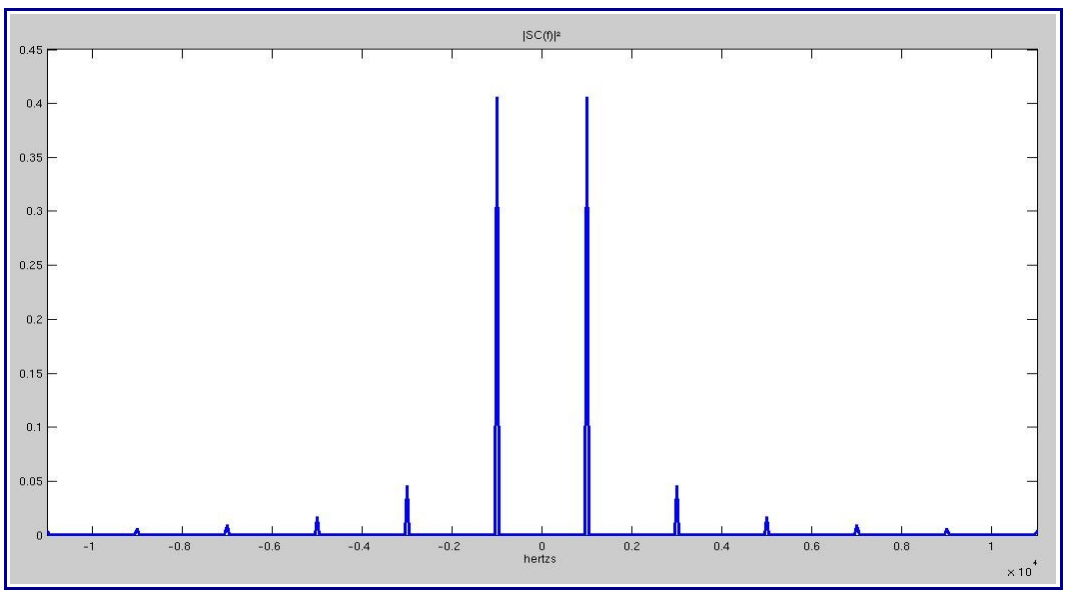


Figure 11: squared magnitude of FT{sign(cos(2π1000t))}

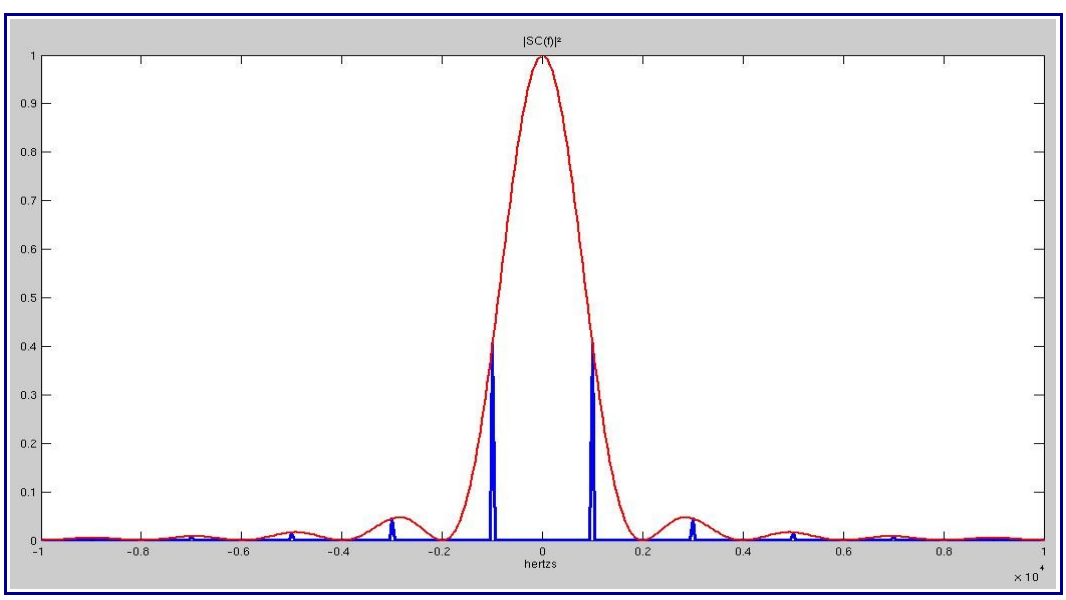


Figure 12: squared magnitude of FT{sign(cos(2π1000t))} and sinc(0.0005 f)

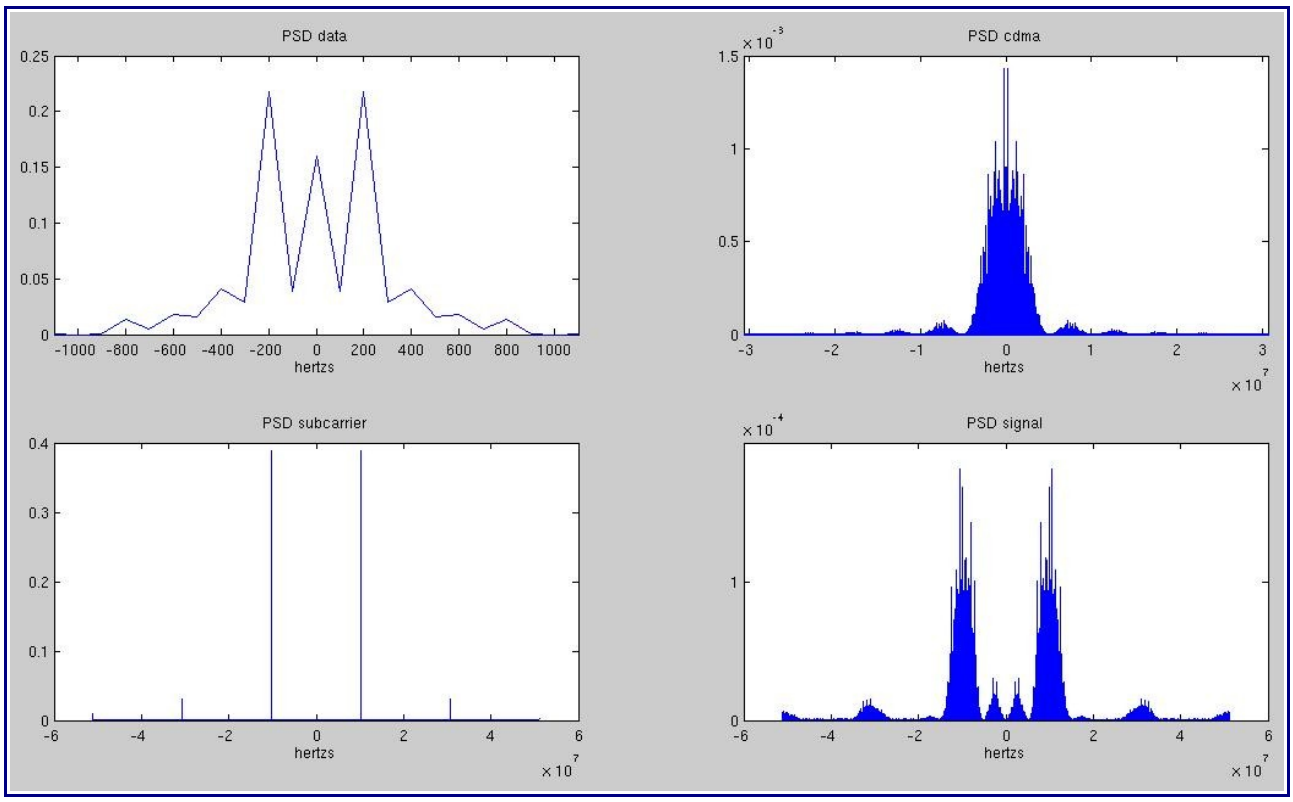


Figure 13: Data (with m_θ!=0), CDMA encoding, sub-carrier and signal s(t) PSD

After all, the spectrum will be represented as one lobe repeated in the different sub-carriers frequencies provided by the transform of the sign of the sub-carrier.

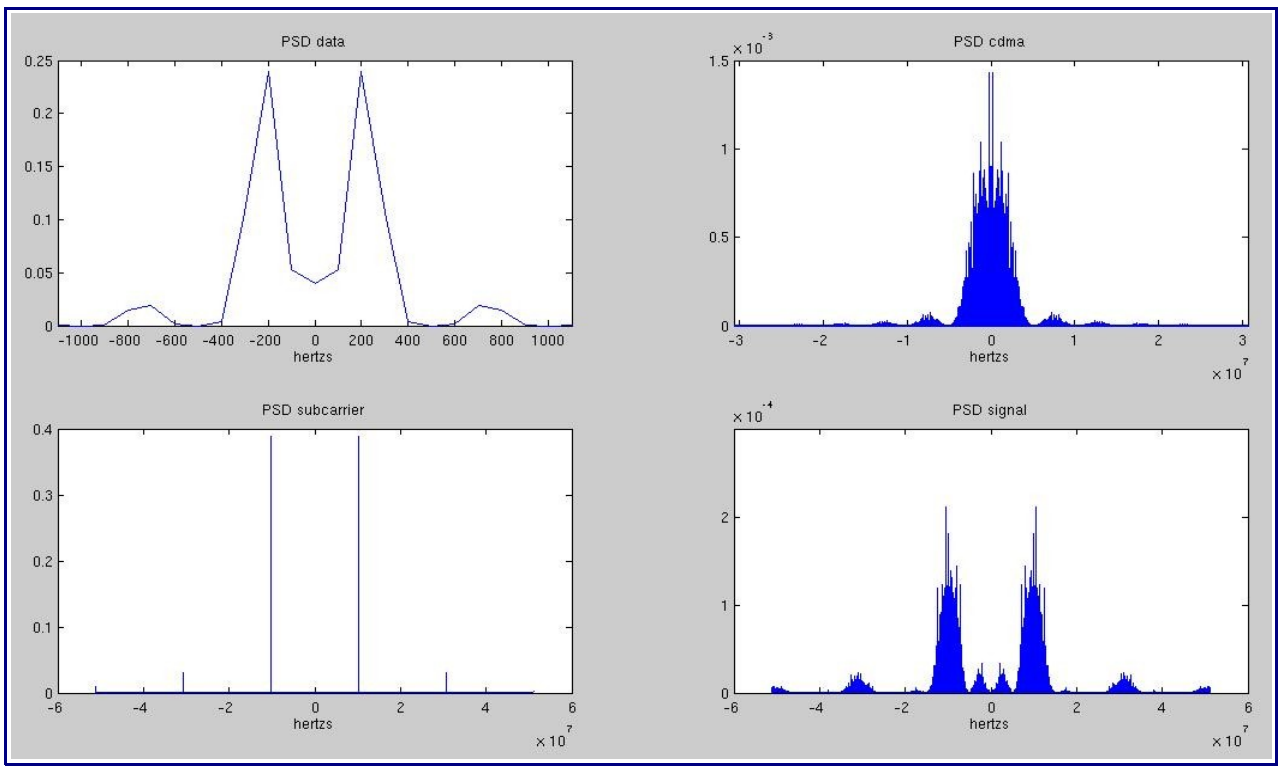


Figure 14: Data (m_θ=0), CDMA encoding, sub-carrier and signal s(t) PSD

2.2.2 Alternate BOC

The Alternate BOC (AltBOC) is a multiplexing and modulating technique, as it will be shown later in this document, that is used in Galileo E5 signal. The case of this modulation will be studied because it is of remarkable interest and because the Galileo E5 signal uses a slightly different and more complex version of this technique. Understanding this first model will ease the understanding of the real version for E5 navigation signal.

From the appendix of [11] we can say that the alternate BOC modulation scheme aims at generating a single sub-carrier signal adopting a source coding similarly to the one involved in the classical BOC. The process allows to keep the BOC implementation simplicity and a constant envelope while permitting to differentiate the lobe.

The standard BOC modulation, as it has shown above, is a square sub-carrier modulation and consists in multipliying a signal *s(t)* with a square sub-carrier of frequency *fs* which splits the spectrum of the signal into two parts, located at the left and right side of the carrier frequency.
The idea of alternate (or baseband) BOC modulation is to perform the same process but multiplying the base band signal by a 'complex' rectangular sub-carrier:

$$rcs(t) = sign(\cos(2\pi f_s t)) + j \cdot sign(\sin(2\pi f_s t)) = cr(t) + j \cdot sr(t) \tag{17}$$

In that way the signal spectrum is not split up, but only shifted to higher frequencies. A different signal *s(t),* containing a different ranging code and navigation data message, can be used for shifting to the lower and upper frequency range. **By this principle the two side lobes of a BOC signal can carry different information**.

The alternate BOC signal can be expressed as:

$$s(t) = C_a(t) \cdot rcs(t) + C_b(t) \cdot r\check{c}s(t) \tag{18}$$

Where:

- $C_a(t)$ is a pseudo random noise (PRN)[4] code for channel *a.*

4 Generated via Linear Feedback Shift Register (LFSR) in Matlab (c.f. Point 3 (Galileo Spreading Codes) and appendix A to check the Matlab script). PRN has to be regarded as a sequence of pulses that repeat themselves, but after a long time or a long sequence of pulses, although it seems to lack any definite pattern.

- $C_b(t)$ is a PRN code for channel *b*.
- *rcs* is the rectangular complex sub-carrier defined above.

As it can be appreciated in the next figure, the signal spectrum comprises a main line which is the same as the line for the ideal (sinusoidal) complex exponential with the same frequency *fs* and minor harmonics spaced every 4*fs*.

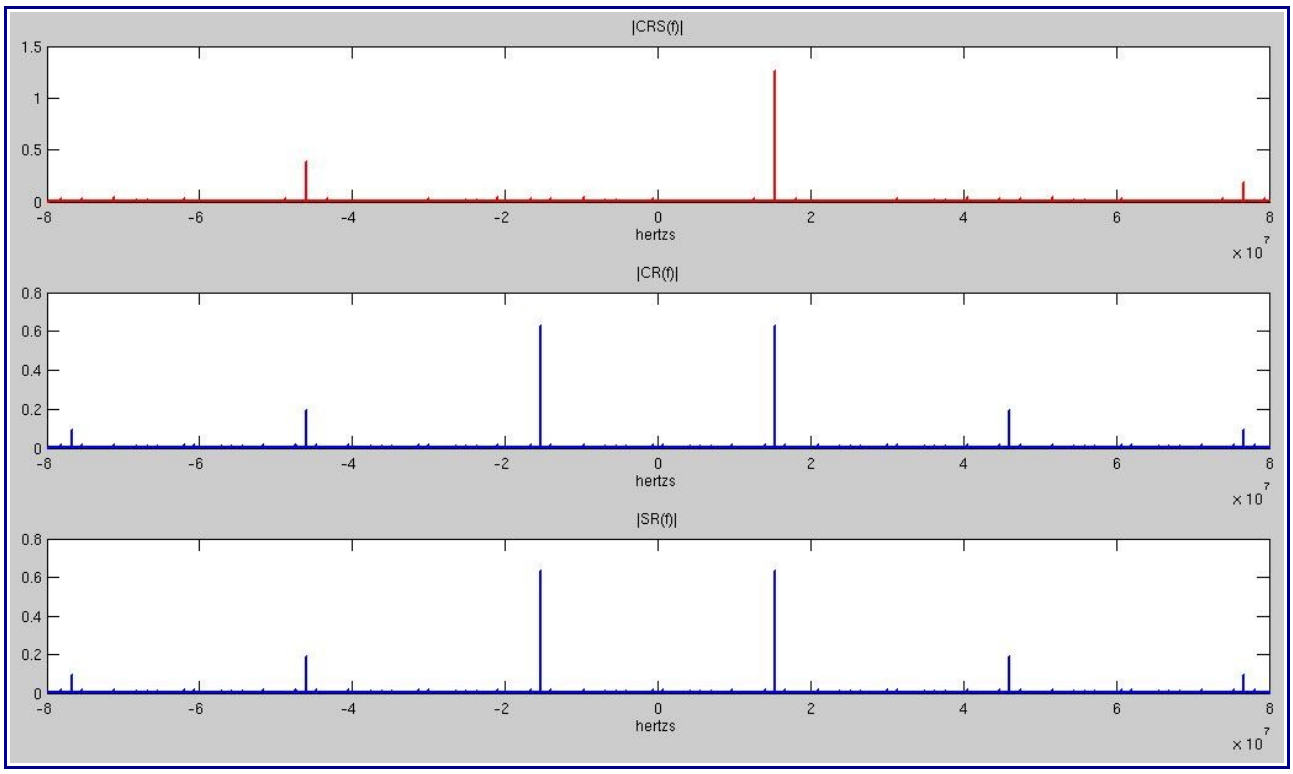


Figure 15: Magnitude of the Fourier transform of the rectangular complex sub-carriers

In this plot we can see how the sum of the different components of the complex exponential are added (it has been plotted without applying the square to appreciate with more detail the several deltas with low amplitude). As it has been shown in section 2.2.1, the Fourier transform of the sign of a cosine is a real function, while the Fourier transform of the sign of a sine is an imaginary function, therefore, when they are joined in a complex exponential, the imaginary part becomes real and they can be treated in the same axe. Thus, for *m* odd and positive, the deltas that will null each other will be the ones located when *m* = 3, 7, 11... For *m* odd and negative, the ones that will null each other will be the ones when *m* = -1, -5, -9... This behaviour can be checked out in the figure of above, where the multiples are located at *fs*, -3·*fs*, 5·*fs*,(every 4·*fs)*, where *fs* = 15·1.023MHz = 15.345 MHz.

It is equivalent to modulate the data flow *Ca(t)+Cb(t)* by the waveform *cr(t)* and to add in quadrature the data flow C*a(t)-Cb(t)* modulated by the waveform *sr(t)*, because the alternate signal expression can be arranged as:

$$\begin{aligned} s(t) &= C_a(t)[sign(\cos(2\pi f_s t)) + j \cdot sign(\sin(2\pi f_s t))] + C_b(t)[sign(\cos(2\pi f_s t)) - j \cdot sign(\sin(2\pi f_s t))] = \\ &= sign(\cos(2\pi f_s t)) \cdot [C_a(t) + C_b(t)] + j \cdot sign(\sin(2\pi f_s t)) \cdot [C_a(t) + C_b(t)] = \\ &= [C_a(t) + C_b(t)] \cdot sign(\cos(2\pi f_s t)) + j \cdot [C_a(t) - C_b(t)] \cdot sign(\sin(2\pi f_s t)) = \\ &= [C_a(t) + C_b(t)] cr(t) + j \cdot [C_a(t) - C_b(t)] sr(t) \end{aligned} \quad (19)$$

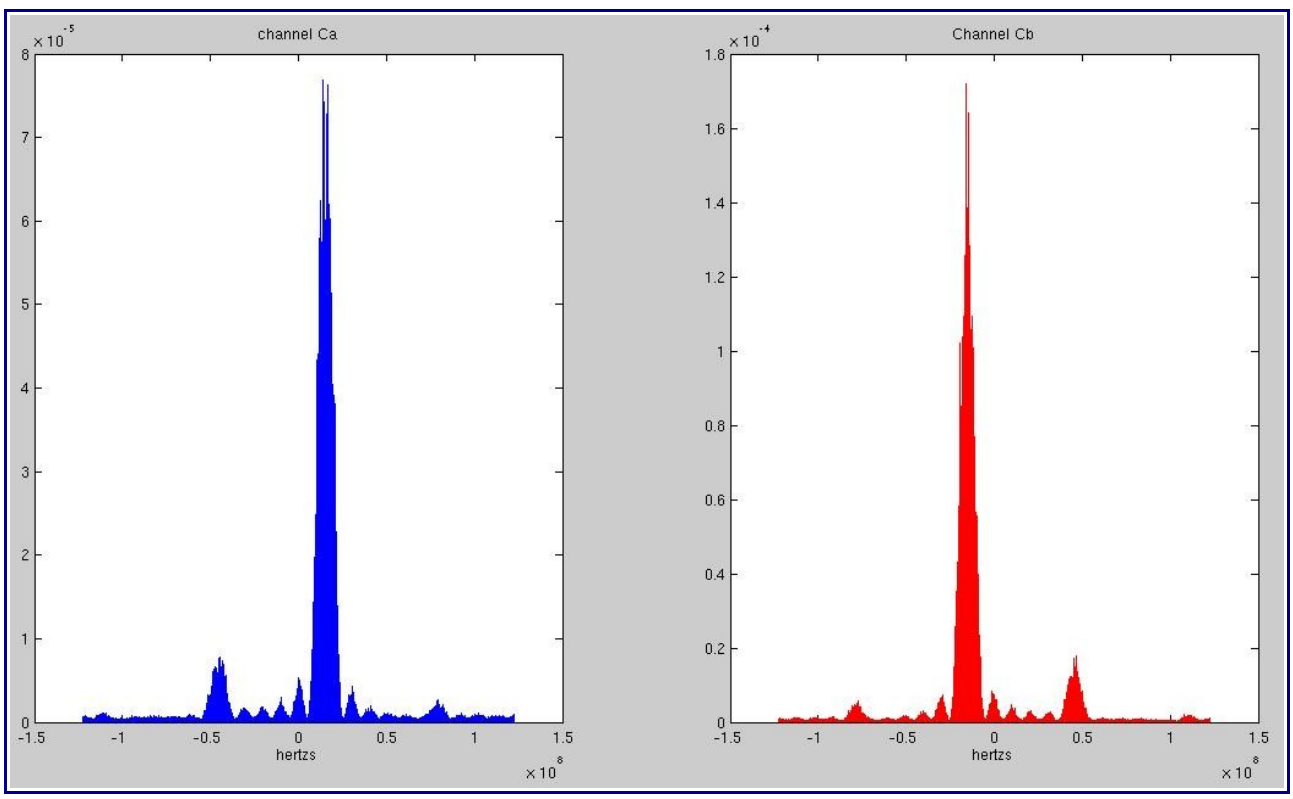


Figure 16: PSD of the formula (18) and (19) with separated channels Ca and Cb

This plot is easier to understand if equation (18) is followed, since it is clearly seen how every channel is multiplied by the rectangular complex sub-carrier (one of them conjugated) and the result is the baseband PRN spectrum convoluted by the different deltas provided by the Fourier transform of the complex exponential, as *figure 16* shows.

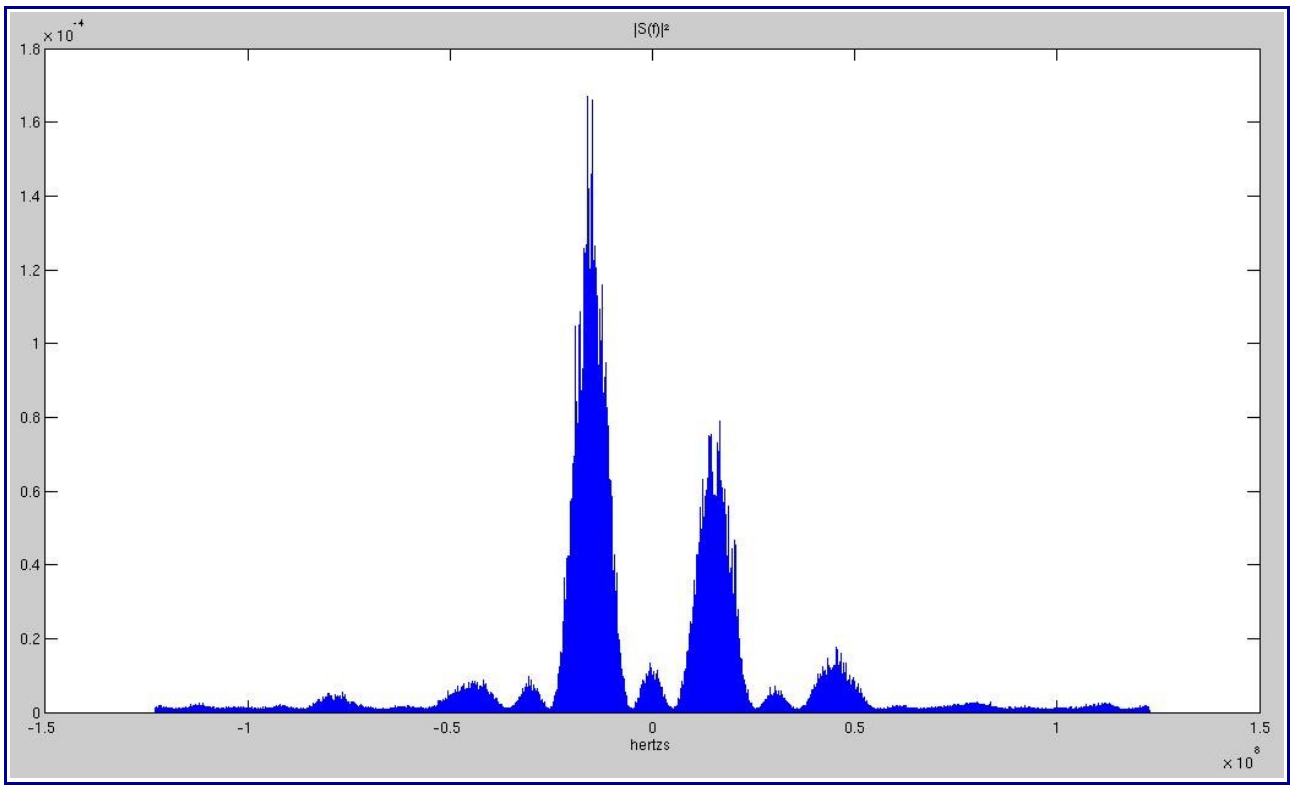


Figure 17: PSD of the formula (18) and (19) with both channels Ca and Cb

This figure shows the sum of both channels Ca(t) and Cb(t) in frequency, where it can be seen that this signal is no mirrored at all and that both lobes are different.

Since the flows C*a(t)* and C*b(t)* can assume only values of +1 and –1, the signal s*(t)* can be written as:

$$s(t) = 2 \cdot e^{jk\frac{\pi}{2}} \qquad k \in \{1,2,3,4\} \tag{20}$$

Thus, we verify that the amplitude of I and Q channel is constant.

The limitations of this basic concept lies in the fact that each signal in each sideband must be a BPSK and no a QPSK, to include pilot channels, are allowed if the good constant envelope characteristics are to be kept, because some portions of the alternate signal will be at null power.

So, if we wanted to add pilot channels, the expression of *s(t)* changes and becomes a signal with in-phase and quadrature components, where the data channels[5] will go in the in-phase component and

5 The data used for this example is at a rate of 50 sps and 250 sps for r_a and r_b respectively (c.f. Section 4 (Signal Generation) and appendix A). It consists in a random string of ±1.

the pilot channels will go in the quadrature component:

$$s(t)=Ca(t)\cdot rcs(t)+Cb(t)r\check{c}s(t)+j[Ca'(t)rcs(t)+Cb'r\check{c}s(t)] \quad (21)$$

Where:

- *Ca(t)* is the data multiplied by the PRN code for channel *a.*
- *Cb(t)* is the data multiplied by the PRN code for channel *b.*
- *rcs(t)* is the rectangular complex sub-carrier.
- *Ca'(t)* is the pilot channel (only PRN code) for channel *a.*
- *Cb'(t)* is the pilot channel (only PRN code) for channel *b.*

Again, if this expression of *s(t)* is developed following the same steps than before, it can be written like:

$$s(t)=\{(Ca(t)+Cb(t))\cdot Cr(t)-(Ca'(t)-Cb'(t))\cdot Sr(t)\}+j\cdot\{(Ca'(t)+Cb'(t))\cdot Cr(t)+(Ca(t)-Cb(t))\cdot Sr(t)\} \quad (22)$$

This signal can take 9 different values, which can be written by the following formula:

$$\begin{aligned} &s(t)=A_k\cdot e^{jk\frac{\pi}{4}} \qquad k\in\{0,1,2,3,4,5,6,7,9\} \\ &with: \\ &A_k=0 \qquad for\ \ k=0 \\ &A_k=2\sqrt{2} \qquad for\ \ k\ \ odd \\ &A_k=4 \qquad for\ \ k\ \ even \end{aligned} \quad (23)$$

It is clearly seen that the resulting modulation will not be a constant envelope modulation. For that reason Galileo AltBOC has been modified introducing two different sub-carriers and the products of the components in order to obtain a signal that lies on an 8-PSK constellation (constant envelope) which is very important for satellites high power amplifiers.

This last form of *s(t)* is very important because this is how E5 signal is composed. As we will see in section 4 (Navigation Signals), E5 signal will be shown as the sum of four terms, where each of the two pairs will be expressed as *s(t)* above.

It is easier to figure out how the shape of the spectrum will be like looking at (21), since each channel is multiplied by the complex sub-carrier and we can foresee where the spectrum will be located.

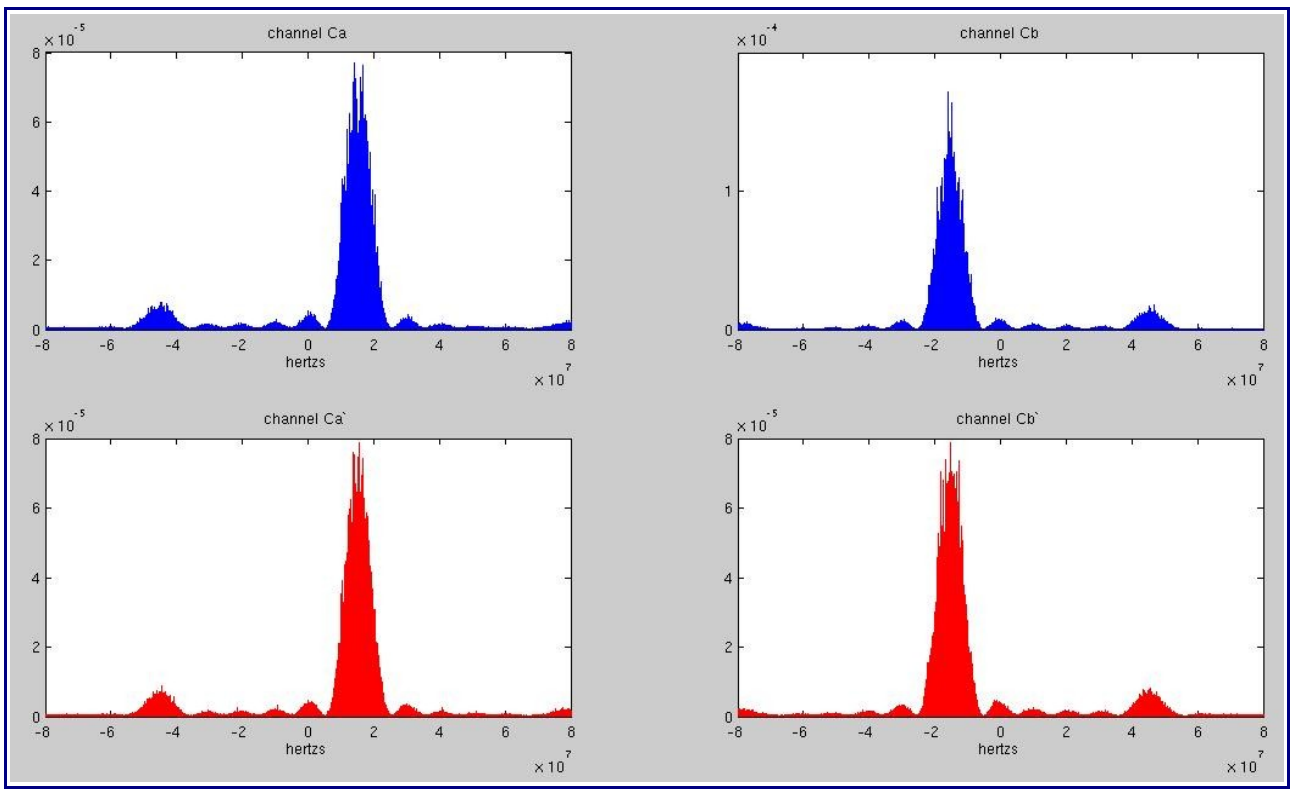


Figure 18: PSD of data (blue) and pilot (red) channels

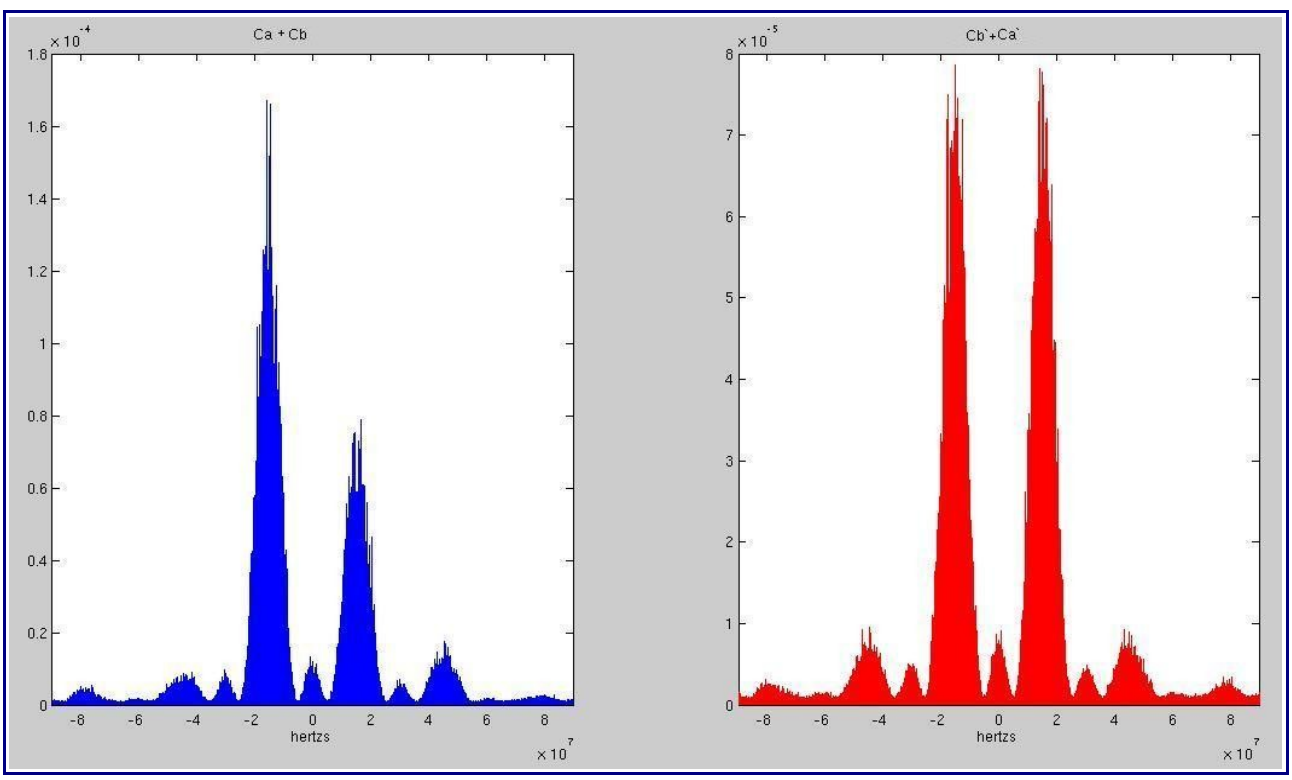


Figure 19: PSD of all channels together in in-phase (blue) and quadrature (red)

2.2.3 Autocorrelation function of BOC

Since the correlation function is the most important point for the acquisition stage of the receiver, it will be analysed how to obtain the ACF of a BOC signal, seeing that it is easier to do the inverse Fourier transform from the PSD form. As an example the BOCsin(1,1) ACF will be derived from its PSD. From this demonstration the importance of following the adequate steps is highlighted, despite the fact that these values ($fs = fc = 1$), lead to a PSK. After demonstrating this, there will be an example of a BOC(2x,x).

$$fs = \frac{1}{2 \cdot Ts} = 1 \quad fc = \frac{1}{n \cdot Ts} = 1 \;\; ; \;\; Ts = 1/2 \quad n = 2$$

$$S_{BOC(1,1)}(f) = \left(\frac{\sin(\frac{\pi f}{2}) \cdot \sin(\pi f)}{\pi f \cdot \cos(\frac{\pi f}{2})} \right)^2 = \left(\frac{\sin(\frac{\pi f}{2}) \cdot 2\sin(\frac{\pi f}{2}) \cdot \cos(\frac{\pi f}{2})}{\pi f \cdot \cos(\frac{\pi f}{2})} \right)^2 = \left(\frac{2 \cdot \sin^2(\frac{\pi f}{2})}{\pi f} \right)^2 =$$

$$= \frac{16}{4\pi^2 f^2} \sin^4(\frac{\pi f}{2}) = \frac{2}{4\pi^2 f^2} \cdot (\cos(2\pi f) - 4\cos(\pi f) + 3) = \frac{2}{(2\pi f)^2} \cdot (3 + \cos(2\pi f) - 4\cos(\frac{2\pi f}{2}))$$

$$where \quad \sin^4(x) = \frac{3}{8} + \frac{\cos(4x)}{8} - \frac{\cos(2x)}{2}$$

So, to transform the last expression inversely, let $\frac{\cos(2\pi fk)}{4\pi^2 f^2}$ be the reference function $G_k(f)$ and $\frac{1}{4k^2\pi^2 f^2}$ be the reference function $S_k(f)$.

Then, the ACF will be :

$$g_k(t) = F^{-1}\left\{ \frac{1}{4\pi^2 f^2} \right\} * F^{-1}\{\cos(2\pi fk)\} = \frac{-t \cdot sign(t)}{2} * \frac{\delta(t-k) + \delta(t+k)}{2} = \frac{-(t-k)sign(t-k) - (t+k)sign(t+k)}{4}$$

$$s_k(t) = F^{-1}\left\{ \frac{1}{4k^2\pi f^2} \right\} = \frac{-t \cdot sign(t)}{2k^2}$$

Once this is defined, we only need to make a linear combination of these two functions [20]:

$$ACF_{BOC_{(1,1)}}(t) = 6 \cdot s_1(t) + 2 \cdot g_1(t) - 8 \cdot g_{\frac{1}{2}}(t)$$

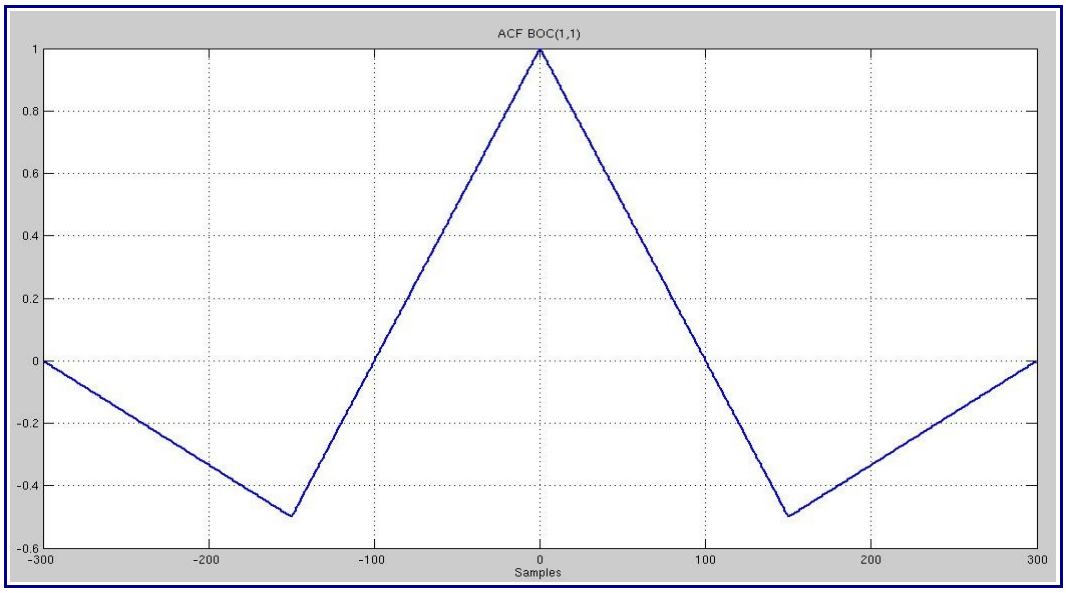


Figure 20: Normalised ACF BOC(1,1)

When the BOC signal is more complex, such as BOC(2x,x), the procedures to obtain the ACF analytically are similar to those of a BOC(1,1), however, the functions to transform inversely are more complicated. As an example, the procedure followed to obtain the ACF of a BOC(10,5) is shown below.

$$fs = \frac{1}{2 \cdot Ts} = 10 \quad fc = \frac{1}{n \cdot Ts} = 5 \;\; ; \;\; Ts = 1/20 \quad n = 4$$

$$S_{BOC(10,5)}(f) = fc \left(\frac{\sin(\pi f \, Ts) \cdot \sin(4 \pi f \, Ts)}{\pi f \cdot \cos(\pi f \cdot Ts)} \right)^2$$

The factor which complicates the inverse transform is the cos(x) in the quotient, since it is not a direct inverse transform. Therefore, trigonometric identities are applied in order to get rid of that cosine function. Whenever *fs = 2fc*, the *n* parameter will be 4, so then it will be possible to apply identity "*sin(2x) = 2 sin(x) cos(x)*" as many times as needed. In this case it is done like this:

$$\sin^2(4 \pi f \, Ts) = (2 \cdot \sin(2 \pi f \, Ts) \cos(2 \pi f \, Ts))^2 = (2 \cdot 2 \cdot \sin(\pi f \, Ts) \cos(\pi f \, Ts) \cos(2 \pi f \, Ts))^2 =$$

$$= 16 \cdot \sin^2(\pi f \, Ts) \cdot \cos^2(2 \pi f \, Ts) \cdot \cos^2(\pi f \, Ts)$$

Therefore, the PSD yields:

$$S_{BOC(10,5)}(f) = \frac{16\,fc}{\pi^2 f^2} \cdot \sin^4(\pi f Ts) \cdot \cos^2(2\pi f Ts)$$

One option after this, is to develop the sinusoid as before, or to try to put both trigonometric functions with the same arguments, since the cosine argument is twice the sinusoid one. If it is applied the identity "$cos^2(2x) = 1 - sin^2(2x)$", it can be rewritten like:

$$\cos^2(2x) = 1 - \sin^2(2x) = 1 - 4\sin^2(x)\cos^2(x)$$

hence,

$$S_{BOC(10,5)}(f) = \frac{16fc}{\pi^2 f^2} \cdot \left(\sin^4(\pi f Ts) - 4\sin^6(\pi f Ts)\cos^2(\pi f Ts)\right)$$

Once this point is reached, one way to continue developing this is to transform all trigonometric functions inversely, in order to have several delayed and amplitude-valued deltas to convolute them in time. Then, it has to be convoluted by the term at the beginning of the expression (whose inverse transform has already been developed) and finally obtain the ACF of BOC(10,5).

By tackling this problem from another point of view (less mathematical and more practical), what it simply happens is that the autocorrelation of a binary square sub-carrier takes place. Therefore, it is enough to simulate a sine and cosine waveform with BOC(fs,fc) characteristics by using appropriate parameters such as the number of samples (sampling frequency). The main important point in this simulation is the parameter *n*, the number of half-periods within a chip. As the sub-carrier is always the component with highest frequency, it is what finally splits the signals the most (even the chips). In the Matlab scripts (*c.f. Appendix A*) the procedure followed to obtain the ACF functions can be seen.

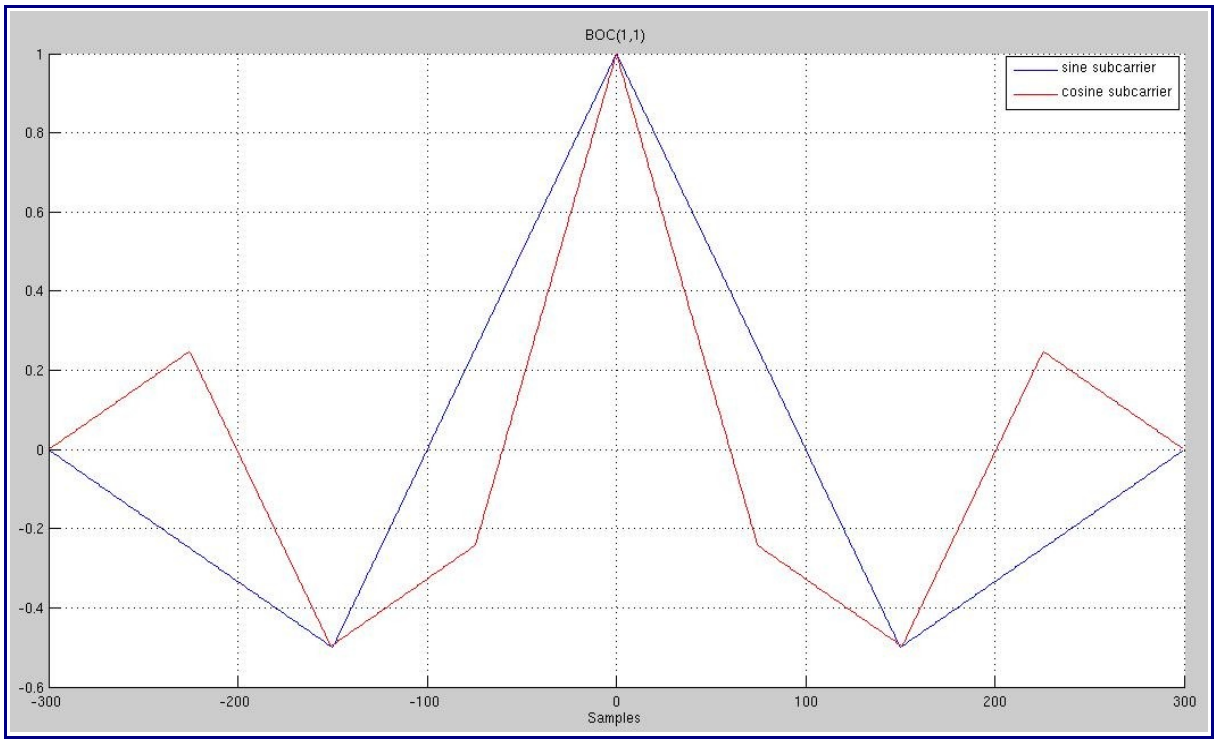


Figure 21: ACF BOC(1,1)

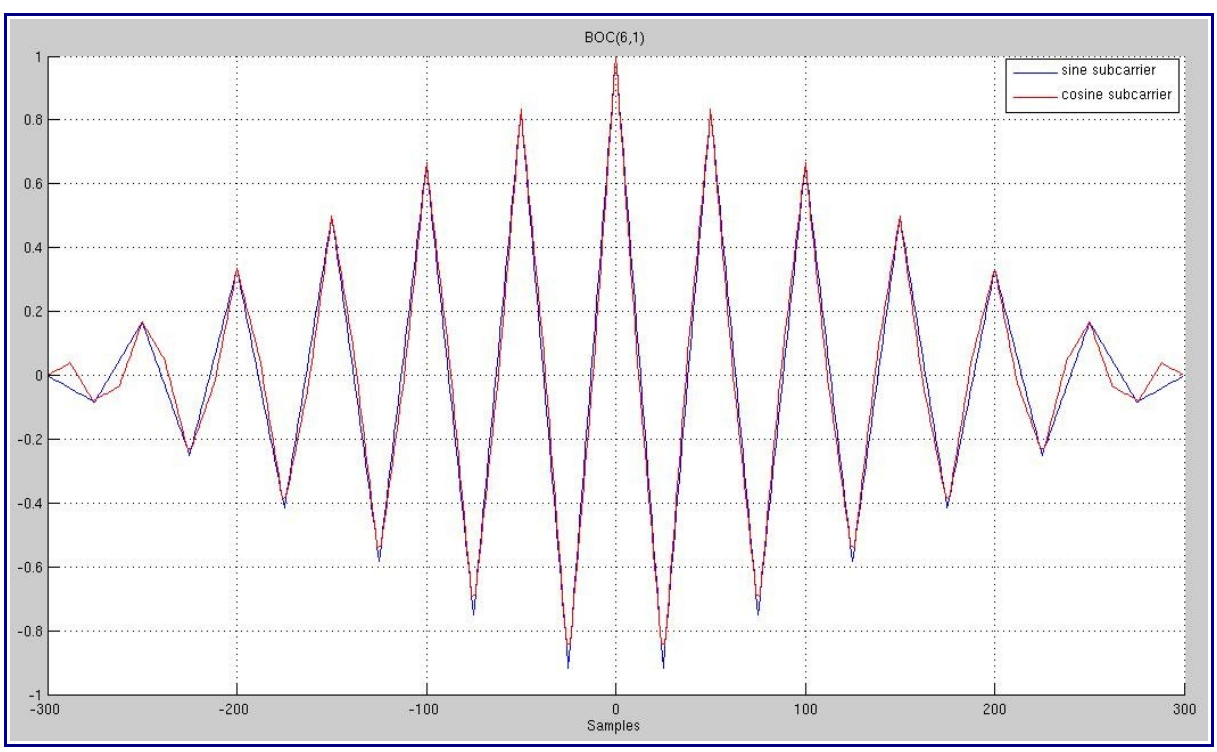


Figure 22: ACF BOC(6,1)

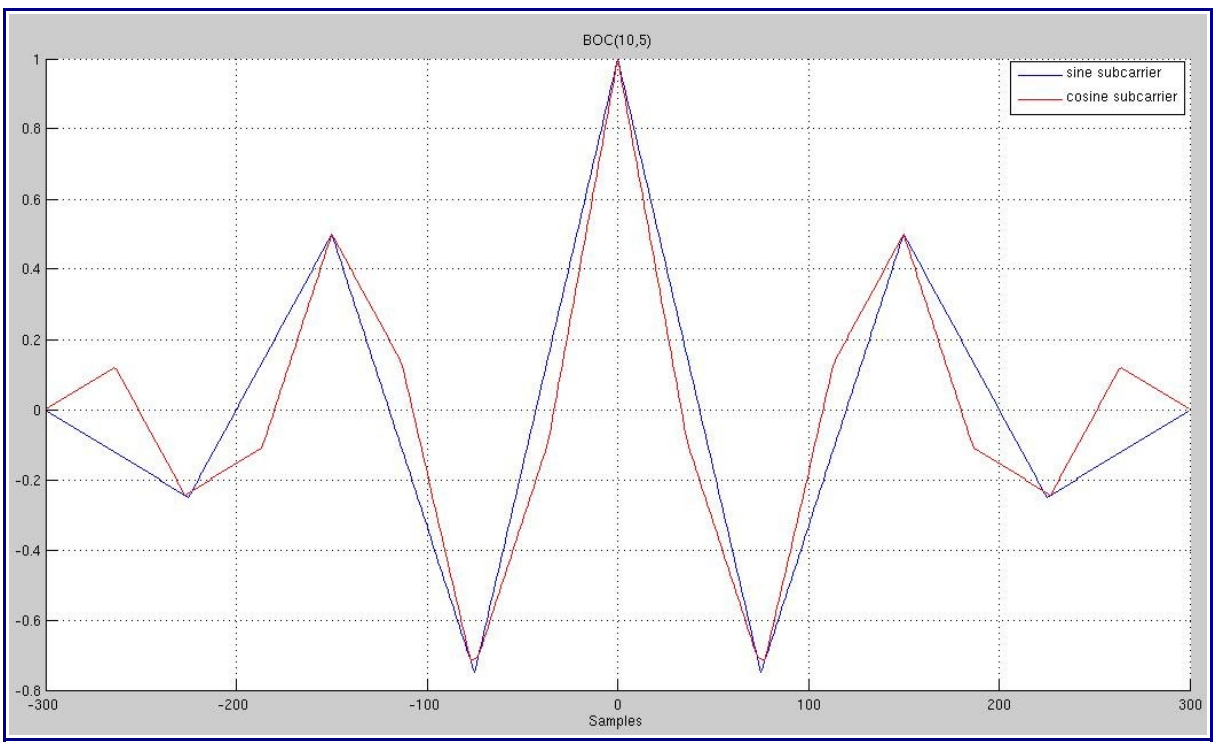


Figure 23: ACF BOC(10,5)

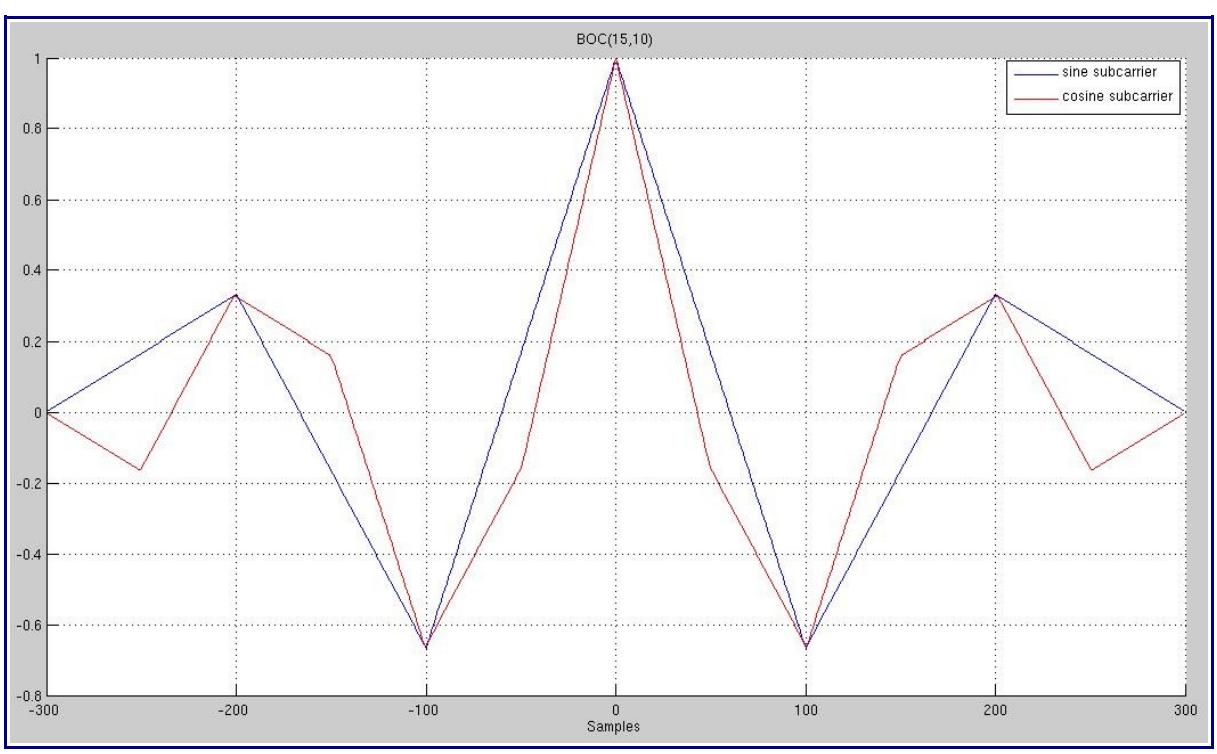


Figure 24: ACF BOC(15,10)

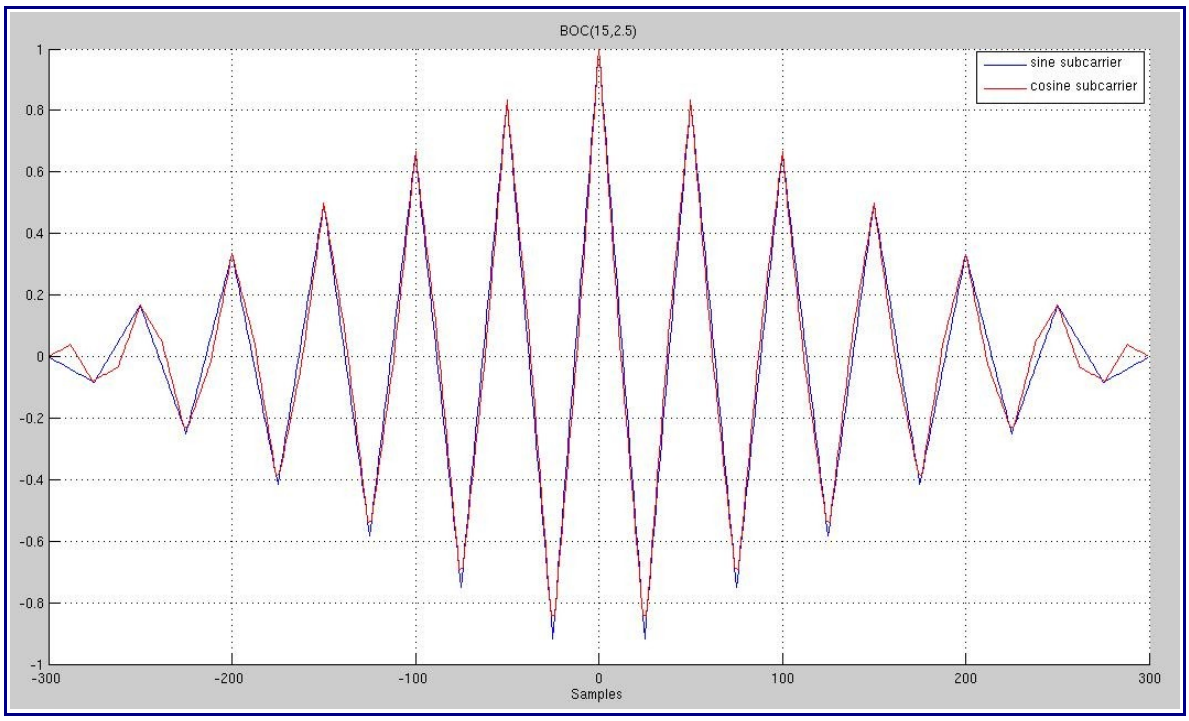


Figure 25: ACF BOC(15,2.5)

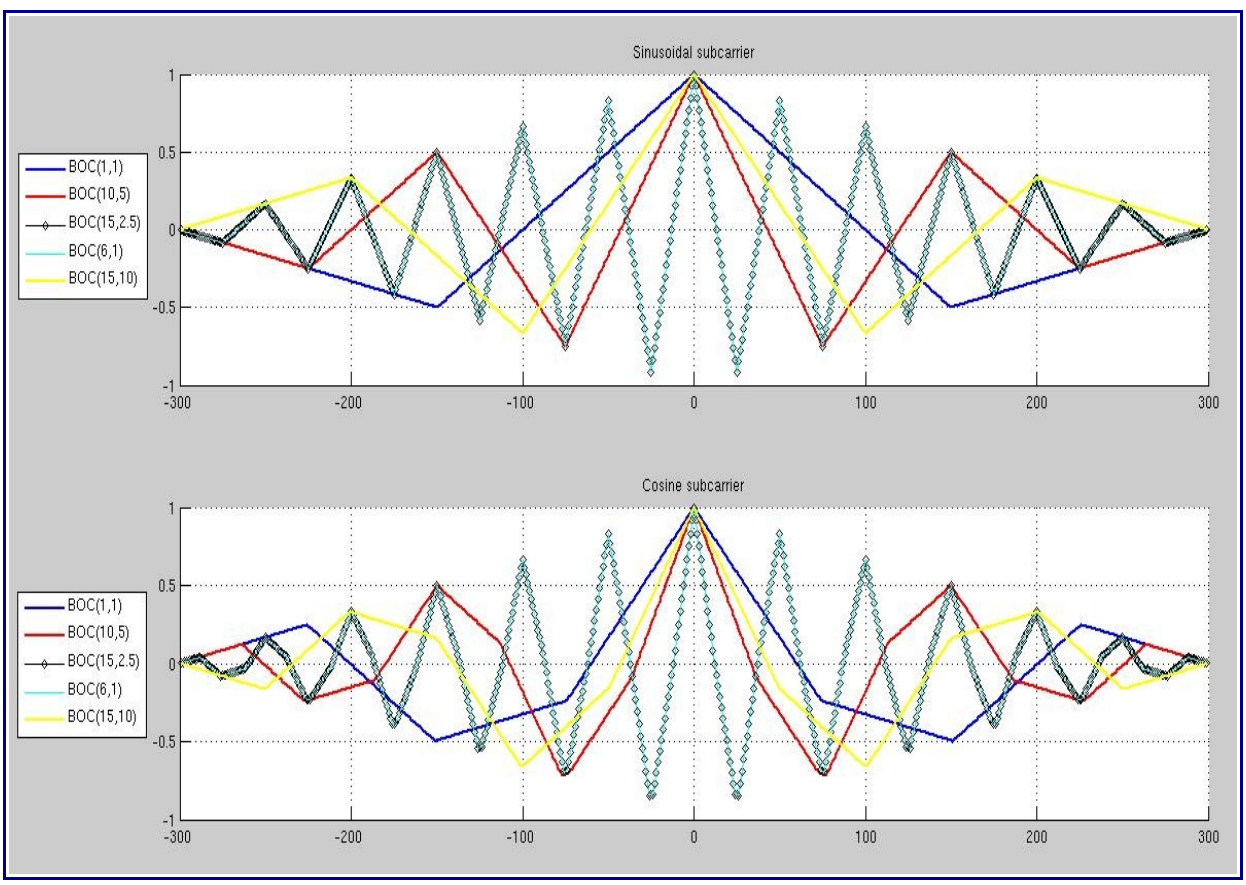


Figure 26: ACF with sine sub-carrier and cosine sub-carrier

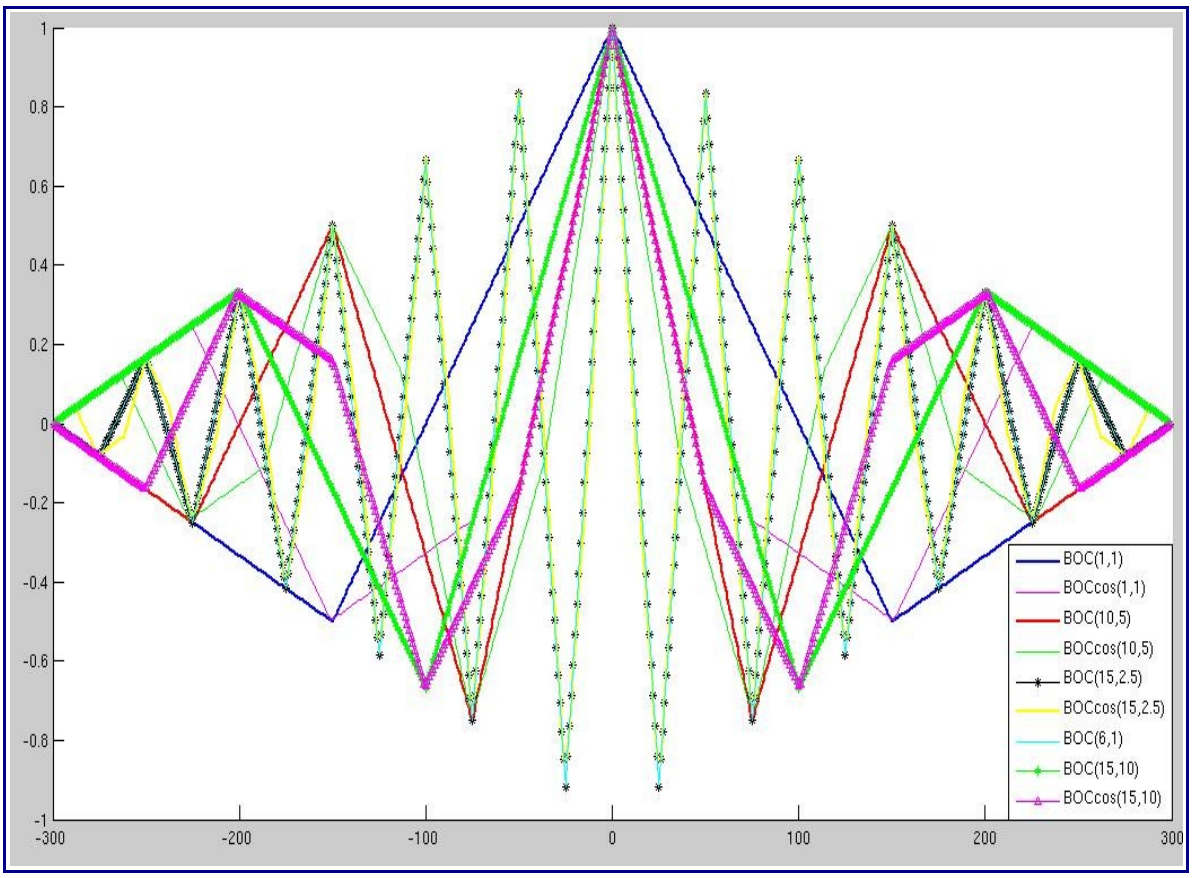


Figure 27: All ACF

In all these figures it can be appreciated how the waveforms for all BOCsin(fs,fc) functions are the expected: "*for an ideal BOC modulation with infinite bandwidth, the autocorrelation function consists of a set of connected line segments with (in general) multiple zero crossings and multiple lobes. The number of negative and positive peaks in the magnitude autocorrelation function is 2n-1, and the peaks are separated in delay by Ts* "[4].

For comparison's sake, it has been plotted as well the BOCcos ACF to see what differences may exist. From the pictures, it can be seen how cosine sub-carrier involves an extrabending which avoids the segments to be straightforward. From [9] it can be appreciated how the results above are faithful to reality. Moreover, it can be also seen that the peak number due to a cosine sub-carrier is *2n+1*.

2.2.4 Coherent Adaptative Sub-carrier Modulation (CASM)

Spectrum limitations for navigation systems require that the various navigation signals broadcast by the Galileo system must be combined and must utilise bandwidth-efficient modulations. At the L1 band, one of the most important questions is how to combine all the OS signals and the PRS signal at the payload level, while maintaining good performance at reception. The **CASM modulation (**also known as "**Interplex modulation**" or "**Modified or Tricode Hexaphase modulation**"), a particular phase-shifted-keyed/phase modulation (PSK/PM), was chosen to transmit these signals because it is a **constant-envelope modulation**, thereby allowing the use of saturated power amplifiers with limited signal distortion. The Interplex modulation was also taken as baseline at the E6 band to transmit the three channels and the services associated on the same carrier frequency. At the E5 band, the modulation must combine two different services on a same constant envelope composite signal, while keeping the simplicity of a BOC implementation [18].

On the **E5 band**, the modulation objective is to multiplex three different services, the OS, the CS and the SoL, included into two BOC signals, while maintaining a constant envelope. AltBOC was chosen to transmit the Galileo E5 signal. In the **L1 band**, the modulation objective is to combine three distinct signals associated to two different services (two OS and one with the PRS) into a phase modulated composite signal that keeps a constant envelope. In this case, the Interplex modulation is preferred because it provides the best overall satellite power efficiencies while verifying all the previous objectives. On the **E6 band** the modulation scheme is similar to the modulation scheme at the L1 band. Indeed, the Interplex modulation is also preferred to combine the three distinct channels associated to the two different services [18].

The formula for this modulation consists in a cosine whose argument contains the carrier frequency, the different channels to be multiplexed and a random phase. Hence, since the cosine function is delimited in the range ±1, the modulated signal will be delimited by the amplitude assigned. This does not mean this signal is constant envelope, but selecting appropriate values it will be easier to achieve that.

So, the formula for the CASM is: $s(t)=\sqrt{2P}\cdot\cos(2\pi\, fc\, t+\theta(t)+\varphi)$

where:

- P is the total average power.

- *fc* the carrier frequency.
- *θ(t)* is the phase modulation.
- *φ* is the random phase.

And in the case of the Galileo Satellite System *θ(t)* is defined as:

$$\theta(t) = \beta_1 s_1(t) + \sum_{n=2}^{N} \beta_n \cdot s_1(t) \cdot s_n(t)$$

where:

- *N* is the number of signals to be multiplexed.
- β_n is the modulation index.
- $s_n(t) = \pm 1$.

The value of the modulation index β_n determines the power allocation for each signal component.

As L1 signal is implemented with CASM, it will be put as an example for the analysis of this modulation. So, in L1 three signals are multiplexed on the same carrier, where one of them will be in the quadrature component, $s_1(t)$, and the other two will be in the in-phase component, $s_2(t)$ and $s_3(t)$. Therefore, the general formula for L1 yields:

$$s(t) = \sqrt{2P} \cdot \cos\left(2\pi\, fc\, t - \frac{\pi}{2} \cdot s_1(t) + \beta_2 \cdot s_1(t) \cdot s_2(t) + \right.$$
$$\left. + \beta_3 \cdot s_1(t) \cdot s_3(t) + \varphi\right)$$

Where β_1 is been assumed $-\pi/2$ because signal $s_1(t)$ has been chosen in quadrature with the two other signals. So, if we develop *s(t)* applying trigonometric expansions:

$$s(t) = \sqrt{2P}\left[\cos(2\pi\, fc\, t + \varphi) \cdot \cos(Q) - \sin(2\pi\, fc\, t + \varphi) \cdot \sin(Q)\right]$$

$$\textit{where} \quad Q = -\frac{\pi}{2} \cdot s_1(t) + \beta_2 \cdot s_1(t) \cdot s_2(t) + \beta_3 \cdot s_1(t) \cdot s_3(t)$$

Now, if we develop the sine and cosine with argument Q and knowing that $s_n(t)$ signals are binary, we obtain:

$$\cos\left(-\frac{\pi}{2}+\beta_2+\beta_3\right)=\sin(\beta_2+\beta_3)=\sin(\beta_2)\cos(\beta_3)+\sin(\beta_3)\cos(\beta_2)$$

$$\sin\left(-\frac{\pi}{2}+\beta_2+\beta_3\right)=-\cos(\beta_2+\beta_3)=-\cos(\beta_2)\cos(\beta_3)+\sin(\beta_2)\sin(\beta_3)$$

Joining all formulas and adding the binary channels as stated before it can be shown that:

$$s(t)=\sqrt{2P}\left[\left[\boldsymbol{s_2(t)}\sin(\beta_2)\cos(\beta_3)+\boldsymbol{s_3(t)}\sin(\beta_3)\cos(\beta_2)\right]\cdot\cos(2\pi\, fs\; t+\varphi)+\right.$$
$$\left.+\left[\boldsymbol{s_1(t)}\cos(\beta_2)\cos(\beta_3)-\boldsymbol{s_1(t)}\,\boldsymbol{s_2(t)}\,\boldsymbol{s_3(t)}\sin(\beta_2)\sin(\beta_3)\right]\cdot\sin(2\pi\, fs\; t+\varphi)\right]$$

If we get the in-phase and quadrature components from this passband expression, it can be expressed as in baseband format:

$$s_b(t)=\sqrt{2P}\left[\left[\boldsymbol{s_2(t)}\sin(\beta_2)\cos(\beta_3)+\boldsymbol{s_3(t)}\sin(\beta_3)\cos(\beta_2)\right]+j\cdot\left[\boldsymbol{s_1(t)}\cos(\beta_2)\cos(\beta_3)-\boldsymbol{s_1(t)}\,\boldsymbol{s_2(t)}\,\boldsymbol{s_3(t)}\sin(\beta_2)\sin(\beta_3)\right]\right]$$

Now, having a look at the power of each component (remembering that the power of a sine or cosine is the amplitude to the square divided by two):

$$P_1=P\cdot\cos^2(\beta_2)\cdot\cos^2(\beta_3) \qquad P_2=P\cdot\sin^2(\beta_2)\cdot\cos^2(\beta_3)$$
$$P_3=P\cdot\sin^2(\beta_3)\cdot\cos^2(\beta_2) \qquad P_{\text{im}}=P\cdot\sin^2(\beta_2)\cdot\sin^2(\beta_3)\quad ,$$

it can be seen that a trade-off must be agreed to avoid that a great amount of available power goes into the intermodulation product (P_{im}) component. Indeed, conditions are needed to stablish the appropriate parameters, so in-phase and quadrature components are meant to have the same power. This leads us to say that *P1* will be the 50% of the power and that both *P2* and *P3* will have a 25% of the power each:

$$P_1=P\cdot\cos^2(\beta_2)\cdot\cos^2(\beta_3)=2\cdot P\cdot\sin^2(\beta_2)\cdot\cos^2(\beta_3)$$
$$P_2=P_3=P\cdot\sin^2(\beta_2)\cdot\cos^2(\beta_3)$$

This system is reduced to solve the following equation: $\cos^2(\beta_2)=2\sin^2(\beta_2)$,

whose solution is: $\beta_2=arctg(\frac{1}{\sqrt{2}})=\pm 0.6155\, radians$

It has to be note the ambiguity introduced by the squares in the sines and cosines function. As we will see in section 4, the channel in the in-phase component is formed by the subtraction of $s_2(t)$ and $s_3(t)$, so we select for β_2 the positive solution and for β_3 the negative solution:

$$\beta_2=-\beta_3=m=0.6155\, radians$$

Consequently, the expression of the signals in baseband is:

$$s_b(t)=\sqrt{2P}\left[\left[\boldsymbol{s_2(t)}\sin(m)\cos(-m)+\boldsymbol{s_3(t)}\sin(-m)\cos(m)\right]+j\cdot\left[\boldsymbol{s_1(t)}\cos(m)\cos(-m)+\boldsymbol{s_1(t)s_2(t)s_3(t)}\sin(m)\sin(-m)\right]\right]$$

Knowing that the sine is an odd function and the cosine an even function, the expression can be showed like:

$$s_b(t)=\sqrt{2P}\left[\left[\boldsymbol{s_2(t)}\sin(m)\cos(m)-\boldsymbol{s_3(t)}\sin(m)\cos(m)\right]+j\cdot\left[\boldsymbol{s_1(t)}\cos^2(m)+\boldsymbol{s_1(t)s_2(t)s_3(t)}\sin^2(m)\right]\right]$$

Regarding the constant envelope issue, if the power for the different channels is analysed adding some restrictions and letting $sin(m)cos(m)=A$, $cos^2(m)=B$ and $sin^2(m)=C$:

$$s_b(t)=\sqrt{2P}\left[\left[A\cdot\boldsymbol{s_2(t)}-A\cdot\boldsymbol{s_3(t)}\right]+j\cdot\left[B\cdot\boldsymbol{s_1(t)}+C\cdot\boldsymbol{s_1(t)s_2(t)s_3(t)}\right]\right]$$

- It is assumed that channel 2 and 3 have the same power.
- It is assumed that channel 1 has the 50% of the relative power and channels 2 and 3 has 25% both.

The first condition expresses that the norm of the I and Q part of the signal $s_b(t)$ must be unity:

$$\sqrt{(A^2-A^2)+(B^2+C^2)}=1$$

The condition of equal power for both I and Q channels lead to:

$$A^2+A^2=B^2$$

Finally, the condition that combined power equals one leads to:

$$A^2+A^2+B^2+C^2=1$$

The system is:

$$\begin{aligned} B+C &= 1 \\ 2\mathrm{A}^2 &= B^2 \\ 2\mathrm{A}^2+B^2+C^2 &= 1 \end{aligned}$$

$$B=1-C \quad \rightarrow \quad 2\cdot(1-C)^2+C^2=1 \quad \rightarrow \quad C=1\,;\, C=\frac{1}{3}$$

Since the solution with *C=1* does not lead to a reasonable result, we get *C=1/3* as a useful solution.

$$\text{Then,} \quad B=\frac{2}{3}\,;\, A=\frac{\sqrt{2}}{3}$$

Hence, the signal can be expressed as:

$$s_b(t)=\sqrt{2P}\left[\left[\frac{\sqrt{2}}{3}\cdot s_2(t)-\frac{\sqrt{2}}{3}\cdot s_3(t)\right]+j\cdot\left[\frac{2}{3}\cdot s_1(t)+\frac{1}{3}\cdot s_1(t)s_2(t)s_3(t)\right]\right] \ ,$$

and now analysing the power for each component:

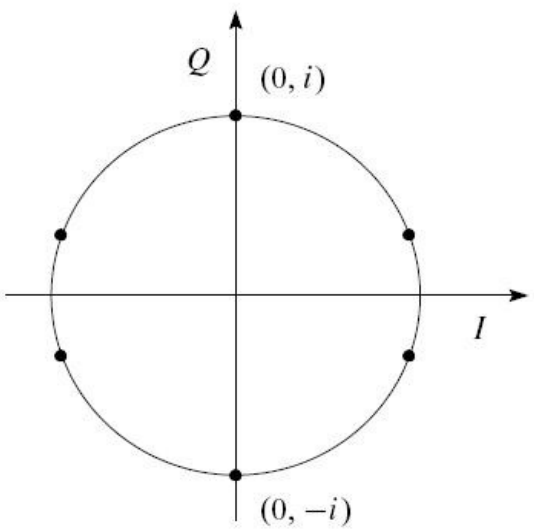
$$\begin{aligned} P_1 &= P\cdot\left(\frac{2}{3}\right)^2=0.44\,P \\ P_2 &= P\cdot\left(\frac{\sqrt{2}}{3}\right)^2=0.22\,P \\ P_3 &= P\cdot\left(\frac{\sqrt{2}}{3}\right)^2=0.22\,P \\ P_{\mathrm{IM}} &= P\cdot\left(\frac{1}{3}\right)^2=0.11\,P \end{aligned}$$

So, this means that only the 88.88% of the total transmitted power is useful, thus showing which is

the trade-off for achieving the constant envelope for signal *s(t)*.

Next table will show the different combinations of the three binary channels $s_1(t)$, $s_2(t)$ and $s_3(t)$, the norm of the signal *s(t),* its real and imaginary part and the constellation for this case of CASM:

$s_1(t)$	$s_2(t)$	$s_3(t)$	*Re {s(t)}*	*Im {s(t)}*	*|s(t)|*
1	1	1	0	1	1
1	1	-1	0.9428	0.3333	1
1	-1	1	-0.9428	0.3333	1
1	-1	-1	0	1	1
-1	1	1	0	-1	1
-1	1	-1	0.9428	-0.3333	1
-1	-1	1	-0.9428	-0.3333	1
-1	-1	-1	0	-1	1



It is observed that the norm of the signal is always 1 and so is represented in the constellation plot of above. It can also be seen from the table that for the three different binary channels, there are six possible phases, thus the name of *tricode hexaphase*. It is also seen that the in-phase component can only assume three different values, so unambiguous mapping from received in-phase signal to transmitted signal is not possible. The table also shows that whenever $s_2(t)$ and $s_3(t)$ (supposed data and pilot) are equal , all power is concentrated in the quadrature component [17].

2.2.5 Double BOC

It must be said that this modulation is not used in any of the Galileo signals. The point to have into account with Double Binary Offset Carrier (**DBOC**) [19] is that it is a new way to write either BOCsin or BOCcos signals using only one formula, that is to say, unifying all BOC signals in one expression. It uses the same parameters that have been explained before in this document about BOC:

- *fs* is the sub-carrier frequency and it is a multiple of 1.023 MHz.
- *fc* is the chip rate and it is a multiple of 1.023 Mcps.
- *n* is the number of half periods within a chip time.

To begin with, we will express the BOCsin signal in baseband like:

$$x(t)=\sum_{m=-\infty}^{+\infty} b_m \sum_{k=1}^{SF} c_{k,m}\, s_{sinBOC}(t-m\cdot T_b-k\cdot T_c)$$

where:

- b_m is the m^{th} bit (symbol).
- $c_{k,m}$ is the k^{th} chip of the m^{th} bit.
- $s_{sinBOC}(t)$ is the waveform of a BOCsin signal (see *figure 7*).
- T_b is the bit (symbol) duration.
- T_c is the chip duration.

Going on with the development and using the convolution operator this equation can be written like:

$$x(t)=s_{sinBOC}(t)*\sum_{m=-\infty}^{\infty}\sum_{k=1}^{SF} b_m c_{k,m}\,\delta(t-m\cdot T_b-k\cdot T_c)=s_{sinBOC}(t)*d(t)$$

where:

- *d(t)* is the spread data symbol.

This spreading symbol will become common in all formulas since the only term that will change is the waveform (sine or cosine).

Then, having a deeper look into $s_{sinBOC}(t)$:

$$s_{sinBOC}(t) = sign(\sin(2\pi f_s t)) \qquad when \quad 0 \leqslant t \leqslant T_c \quad ,$$

this equation can be expressed (using the convolution operator again) like:

$$s_{sinBOC}(t) = p(t) * \sum_{i=0}^{n-1} (-1)^i \delta(t - i \cdot T_s)$$

where:

- $p(t)$ is a rectangular pulse of amplitude 1 and duration T_s.
- n is the number of half-periods within a chip time.
- T_s is the the half of the sub-carrier period.

Once reached this point, the BOCcos signal can be regarded as the same as BOCsin but adding another summation that will split the half-periods into two parts, thus getting the appropriate waveform for the BOCcos signal (see *figure 10*):

$$s_{cosBOC}(t) = p'(t) * \sum_{l=0}^{1} \sum_{i=0}^{n-1} (-1)^{i+l} \delta(t - iT_s - l\frac{T_s}{2})$$

where:

- $p'(t)$ is a rectangular pulse of amplitude 1 and **duration $T_s/2$**.
- l accounts for the splitting of the half period.

This last expression can be also written unwrapping the summation in l since it is only for 0 and 1:

$$s_{cosBOC}(t) = p'(t) * \left[\sum_{i=0}^{n-1} (-1)^i \delta(t - i \cdot T_s) - \sum_{i=0}^{n-1} (-1)^i \delta(t - i \cdot T_s - \frac{T_s}{2}) \right]$$

So, looking at the first $s_{cosBOC}(t)$ expression, the BOCcos can be regarded as a BOCsin modulated signal in which the half period is further split into two parts. Therefore, if this way of splitting is generalised to higher stages, we can express a new type of modulation only varying the counter taken by l:

$$s_{DBOC}(t) = \hat{p}(t) * \sum_{l=0}^{n'-1} \sum_{i=0}^{n-1} (-1)^{i+l} \delta(t - iT_s - l\frac{T_s}{n'})$$

where:

- $\hat{p}(t)$ is a rectangular pulse of amplitude 1 and duration T_s /n'.
- n' can be seen as the order of the second stage (in case of BOCcos it is 2).

Finally, a DBOC signal can be expressed like:

$$x(t) = s_{DBOC}(t) * d(t)$$

The difference between BOC and DBOC is mainly the second order stage parameter n'. When it is increased, the signal mainlobes move further towards the outer sides of the spectrum. A proof of this is simply observed in the BOCsin and BOCcos PSD, where it has been shown that the main lobes for a cosine BOC are a little bit further than the ones from a sine BOC (see underneath explanation in *figure 6*, *figure 6* and next plot at the end of this section) due to the n' parameter, which is 2 for BOCcos and 1 for BOCsin.

To appreciate this spectral feature, the best is to get the PSD function [19]:

$$P_{DBOC}(f) = |S_{DBOC}(f)|^2$$

where $S_{DBOC}(t)$ is the Fourier transform of $s_{DBOC}(t)$:

$$S_{DBOC}(f)=\frac{Ts}{n'}sinc(\frac{Ts}{n'}f)\cdot e^{-j\pi f\frac{T_s}{n'}}\cdot\sum_{l=0}^{n'-1}\sum_{i=0}^{n-1}(-1)^{i+k}e^{-j2\pi fiT_s}\cdot e^{-j2\pi fl\frac{T_s}{n'}}$$

If every exponential is separated in the correct summation, it will be possible to apply the geometric series development:

$$S_{DBOC}(f)=\frac{Ts}{n'}sinc(\frac{Ts}{n'}f)\cdot e^{-j\pi f\frac{T_s}{n'}}\cdot\left[\sum_{l=0}^{n'-1}(-1)^l\cdot e^{j2\pi fl\frac{T_s}{n'}}\cdot\sum_{i=0}^{n-1}(-1)^l e^{j2\pi fiT_s}\right]=$$

$$=\frac{Ts}{n'}sinc(\frac{Ts}{n'}f)\cdot e^{-j\pi f\frac{T_s}{n'}}\cdot\left(\frac{1-(-1)^{n'}\cdot e^{-j2\pi fT_s}}{1-(-e^{-j2\pi f\frac{T_s}{n'}})}\right)\cdot\left(\frac{1-(-1)^n\cdot e^{-j2\pi fnT_s}}{1-(-e^{-j2\pi fT_s})}\right)=$$

$$=\frac{Ts}{n'}sinc(\frac{Ts}{n'}f)\cdot e^{-j\pi f\frac{T_s}{n'}}\cdot\left(\frac{1-(-1)^{n'}\cdot e^{-j2\pi fT_s}}{1+e^{-j2\pi f\frac{T_s}{n'}}}\right)\cdot\left(\frac{1-(-1)^n\cdot e^{-j2\pi fT_c}}{1+e^{-j2\pi fT_s}}\right)$$

Applying now the square absolute value we lose the first exponential term:

$$P_{DBOC}(f)=\left|\frac{Ts}{n'}sinc(\frac{Ts}{n'}f)\cdot\left(\frac{1-(-1)^{n'}\cdot e^{-j2\pi fT_s}}{1+e^{-j2\pi f\frac{T_s}{n'}}}\right)\cdot\left(\frac{1-(-1)^n\cdot e^{-j2\pi fT_c}}{1+e^{-j2\pi fT_s}}\right)\right|^2$$

Developing this last equation we get four different forms of the PSD depending on the parameters n' and n.

- if n is odd and n' is odd: $P_{DBOC}(f)=\left(\frac{\sin(\pi f\frac{T_s}{n'})\cos(\pi fT_c)}{\pi f\cos(\pi f\frac{T_s}{n'})}\right)^2$

- if n is odd and n' is even: $$P_{DBOC}(f)=\left(\frac{\sin(\pi f\frac{T_s}{n'})\sin(\pi f T_s)\cos(\pi f T_c)}{\pi f\cos(\pi f\frac{T_s}{n'})\cos(\pi f T_s)}\right)^2$$

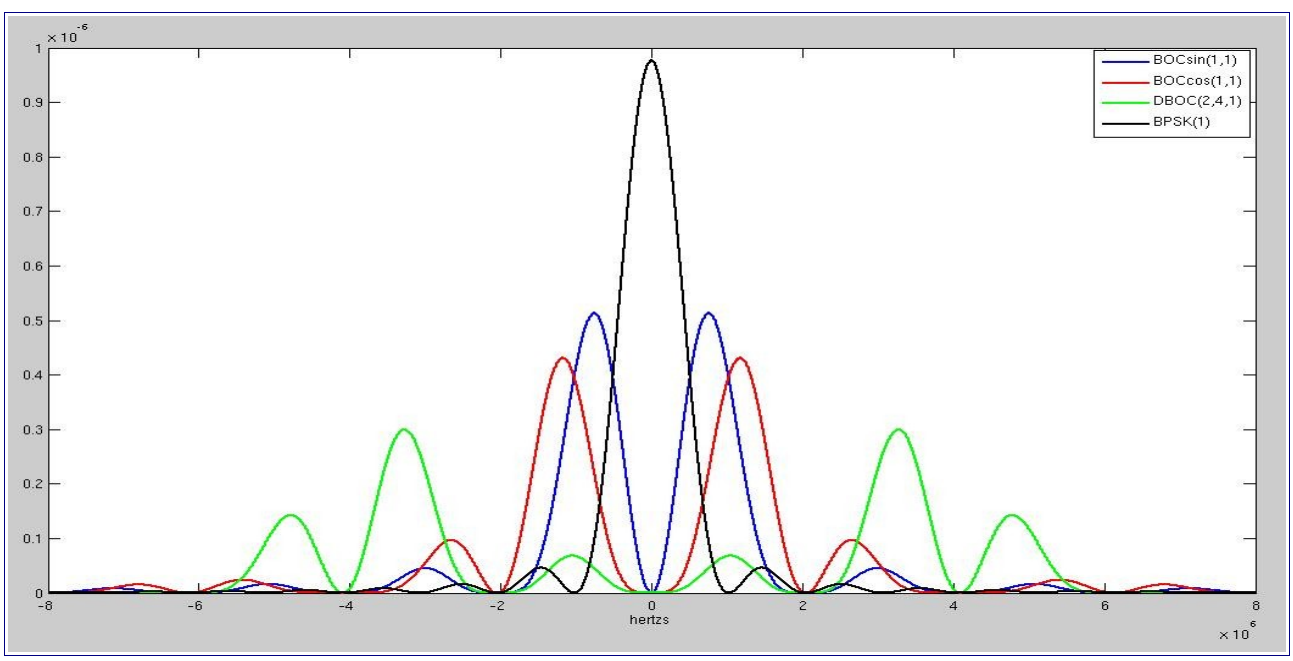
- if n is even and n' is odd: $$P_{DBOC}(f)=\left(\frac{\sin(\pi f\frac{T_s}{n'})\sin(\pi f T_c)}{\pi f\cos(\pi f\frac{T_s}{n'})}\right)^2$$

- if n is even and n' is even: $$P_{DBOC}(f)=\left(\frac{\sin(\pi f\frac{T_s}{n'})\sin(\pi f T_s)\sin(\pi f T_c)}{\pi f\cos(\pi f\frac{T_s}{n'})\cos(\pi f T_s)}\right)^2$$

So, depending on these two parameters we can obtain different modulations:

- n = 1 ; n' = 1 → DBOC = BPSK
- n > 1 ; n' = 1 → DBOC = BOCsin
- n > 1 ; n' = 2 → DBOC = BOCcos
- n >1 ; n' > 2 → High order DBOC

In the next plot we can observe how the DBOC (with n = 2 and n' = 4) is more separated from 0 than the BOCsin(1,1) or BOCcos(1,1), thus giving less interference with the signal in 0.



3. Galileo Spreading Codes

The spreading codes are used to create the Code Division Multiple Access (CDMA) codification (ranging codes) for all Galileo signals. They are constructed by what is called “tiered codes”, consisting of a modulation of two codes: a primary code modulated by a secondary code, obtaining the ranging sequence. Each signal component has assigned one ranging code with different parameters which depend on their length (in time, in chips and number of registers of the Linear Feedback Shift Register (LFSR)), chip rate and symbol rate.

Both codes can be implemented either by “memory codes” or “register-based codes”. Memory codes, also called “optimized pseudo-noise sequences”, are predefined codes which have been constructed expressly to have very good characteristics on correlation and need to be stored in memory because there is no systematic generation possible. Register-based codes are obtained with a LFSR (which is completely defined with a tap polynomial). There are 'r' registers (which dictates the degree of the LFSR) and *r+1* taps (which will produce the sequence).

3.1 Primary Codes

A primary code is generated by the **xor** addition of **two** LFSRs (see figure 28) which are truncated to the desired length (depending on the signal). Both LFSRs have the same number of registers. Having 14 registers means that the LFSR period is a 2^{14}-1 (16383) chip sequence if the tap polynomial is a primitive polynomial. So, for instance, if the longest sequence to be obtained is 10230 chips, that means that the truncation has to be done at 10230.

When a sequence is completed, the registers are reloaded with the start-values. For the first LFSR, the start-values are always **1's**, for the second, the start-values vary and are listed in [1] together with the initial sequences which are also predefined.

All **start-values** codes are given in octal format. If it refers to a tap polynomial, the conversion to binary is as follows: if we have the polynomial 40503, all numbers must be turned into binary separately, that is 100 000 101 000 011, where the most significant bit (MSB) is the one at the left and the least significant bit (LSB) is the one at the right. The indexation for this goes from tap #0, LSB, to tap #14, MSB, (which are always 1 due to the LFSR structure) .

If it refers to a start-value, the conversion to binary is as follows: if we have the start-value 14234, all numbers must be turned into binary separately, that is 001 100 010 011 100, where the first binary value is not used (as we have 14 registers only) and the following one is the MSB, with

indexation 14. The last one (0) has index #1 and it is the LSB.

Regarding **initial sequences**, they come into hexadecimal format and the process is barely the same: all numbers are turned into binary separately. If we have the initial sequence 9D8CF1, the translation is 1001 1101 1000 1100 1111 0001 and it is placed at the beginning of each sequence of the second LFSR.

3.2 Secondary Codes

Secondary codes are all memory codes and are provided in [1]. They are all assigned to a specific signal. For instance, if it is assigned the code $CS20_1$, CS indicates that it is a secondary code, the next number (20) is the number of chips that that code has, and the subscript number (1) is the number of codes, with that length (20), assigned to that signal. For example, for E5aI signal, we have the first code of 20 chips which is in hexadecimal 842E9, that is 1000 0100 0010 1110 1001.

The aim of these codes is to modulate the primary codes. For instance, if we get the signal E5aI, we have a primary chip length (Np) of 10230 and a secondary chip length (Ns) of 20. Moreover, the tiered code length is 20 ms and the chiprate is 10.230 Mcps. So, we should have a total sequence length of 10230 chips x 20 chips = 204600 chips, which coincides with the product 20 ms x 10.230 Mcps = 204600 chips. To obtain the total amount of chips, each chip of the secondary code is multiplied by all chips of the primary code, having in the end 'Ns' times the 'Np' chips (Ns x Np).

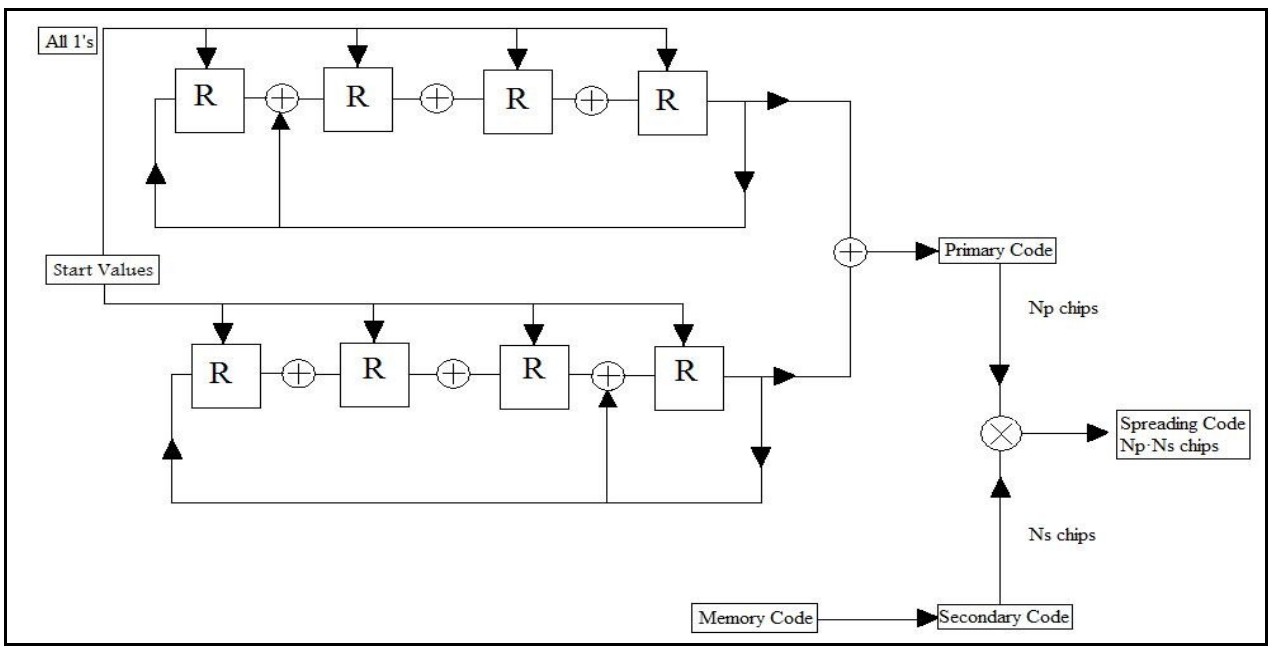


Figure 28: Simplified spreading code implementation scheme

In *figure 28* it can be appreciated the scheme that has been used to achieve the CDMA codification. For the sake of simplicity, it represents a simplified version of the 14th degree polynomial that is

used in the simulation. We can see the two LFSRs which generate the primary code. The first set of registers have start values always in 1, whereas the base register two has different start values which are taken from a list. As commented above, all secondary codes are memory codes and they modulate the primary code as seen in the figure: one chip of the secondary code per all the chips of the primary code, making in the end a *Np·Ns* chips length code.

3.3 Code assignation

The codes have been implemented as faithfully as possible since some information in [1] is ambiguous. So, the implementations of ranging codes for E6 (channel A) and L1 (channel A) have been assumed since the information is restricted for both signals components.

The importance of all these codes falls on decodification (acquiring and tracking a specific satellite), so when it comes to the implementation of the signals is not that important, since what is being done is simulating the signals' PSD. The way in which a ranging code varies is reflected in its chip rate, the higher the chip rate the wider the main lobe. So, in the end, both channels A from E6 and L1 have been assumed just for a complete representation of the Galileo band in Matlab because they are restricted signals.

3.3.1 E5 ranging codes

The ranging codes information for this signal are provided in [1]. They can be implemented either by memory codes or register based codes. Two LFSRs (base register one and base register two) of degree 14 are used. These two registers generate what is called a *Gold Code*[6].

Although we are provided with 50 different secondary codes for each signal component, it has been used only one secondary code for each pilot channel E5aQ and E5bQ. Conversely, the quantity of secondary codes is the exact for data channels E5aI and E5bI, only one for each channel. Moreover, the construction of the ranging codes for these channels entails a reset of 100 times for both LFSRs of the primary code, since there is a secondary code of 100 chips. Therefore, as we are only provided with 50 start values (for base register 2), the 50 start values left have been assumed to be the same, but using them the other way round, that is to say, flipping the order when they are loaded into the registers.

6. Gold Code: xor addition of two maximum length sequences of the same length (2^m-1) which give only three values in its cross-correlation function.

3.3.2 E6 ranging codes

The ranging codes information for this signal is provided in [10]. Each of the ranging codes has been implemented by two LFSRs of 13^{th} degree (channel B) and another pair of 14^{th} degree (channel C), being Gold Codes as well. Moreover, channel C has a secondary code of 50 bits which modulates the primary code. For both channels B and C information is given, however, as channel A is restricted, for simplicity it has been assumed to be channel B multiplied by -1.

3.3.3 L1-E1 ranging codes

On the contrary, E1 ranging codes are memory codes and are already provided in [1]. There are 50 primary codes for each E1-B and E1-C components. It is used only one of the fifty codes provided for the spreading of channel B. The code for channel C is the modulation of a primary code and a secondary code (both memory codes). Channel A, again, is restricted and it has been assumed to be one of the codes of the list for channel B.

4. Signal Generation

The following table defines the signal parameters used for signals:

- 'X' accounting for the respective carrier (E5, E5a, E5b, E6 or E1)
- 'Y' accounting for the respective signal component (A, B, C, I or Q) within the signal 'X'.

Parameter	Explanation	Unit
f_X	Carrier frequency	Hz
P_X	RF-Signal power	W
$L_{X\text{-}Y}$	Ranging code repetition period	chips
$T_{C,X\text{-}Y}$	Ranging code chip length	s
$T_{S,X\text{-}Y}$	Subcarrier period	s
$T_{D,X\text{-}Y}$	Navigation message symbol duration	s
$R_{C,X\text{-}Y}$	$=1/T_{C,X\text{-}Y}$ code chip rate	Hz
$R_{S,X\text{-}Y}$	$=1/T_{S,X\text{-}Y}$ subcarrier frequency	Hz
$R_{D,X\text{-}Y}$	$=1/T_{D,X\text{-}Y}$ navigation message symbol rate	Hz
$S_X(t)$	Signal pass-band representation	N/A
$C_{X\text{-}Y}(t)$	Binary (NRZ modulated) ranging code	N/A
$D_{X\text{-}Y}(t)$	Binary (NRZ modulated) navigation message signal	N/A
$sc_{X\text{-}Y}(t)$	Binary (NRZ modulated) subcarrier	N/A
$e_{X\text{-}Y}(t)$	Binary NRZ modulated navigation signal component including code, sub-carrier (if available) and navigation message data (if available); $(=c_{X\text{-}Y}(t)\ sc_{X\text{-}Y}(t)\ D_{X\text{-}Y}(t))$	N/A
$s_X(t)$	Normalised (unit mean power) baseband signal $=s_{X\text{-}I}(t)+j\ s_{X\text{-}Q}(t)$	N/A
$c_{X\text{-}Y,k}$	k_{th} chip of the ranging code	N/A
$d_{X\text{-}Y,k}$	k_{th} symbol of the navigation message	N/A
$DC_{X\text{-}Y}$	$=T_{D,XY}/T_{C,XY}$ number of code chips per symbol	N/A
$\|i\|_L$	i modulo L	N/A
$[i]_{DC}$	Integer part of i/DC	N/A
$rect_T(t)$	Function "rectangle" which is equal to 1 for $0<t<T$ and equal to 0 elsewhere	N/A

4.1 E5 Signal

The Galileo E5-signal consists of the signals E5a, E5b (and modulation product signals) and is transmitted in the frequency band 1164 - 1215 MHz allocated to Radio Navigation Satellite Systems (RNSS) with a worldwide co-primary status. The E5-signal shares the band with the co-primary Aeronautical Radionavigation Service (ARNS) (ITU-R Radio Regulations). Moreover, it shares the band with other RNSS-signals provided by European Geostationary Navigation Overlay Service (EGNOS), GPS-L5, GLONASS, etc. as well as signals of the ARNS (DME, TACAN) [1].

As it can be seen in [7], the AltBOC modulation scheme will be used by Galileo satellites to broadcast new navigation signals in the E5 band (1164-1215 MHz). The signal E5 is modulated and multiplexed in AltBOC(15,10). The AltBOC modulated signals are one of the most promising innovations of the Galileo system. The AltBOC derives from the family of BOC modulations. This modulation scheme allows a constant envelope modulated signal and can also be considered as a *multiplexing technique*. In fact, the AltBOC modulated signal features a split spectrum with two main lobes. In this way the E5 band can be used as two separate sidebands, conventionally denoted as E5A and E5B transmitting four different channels (E5a-I, E5a-Q, E5b-I, E5b-Q). Each data and pilot channels are the in-phase and the quadrature components, since all signals are provided with its complex envelope (baseband) format.

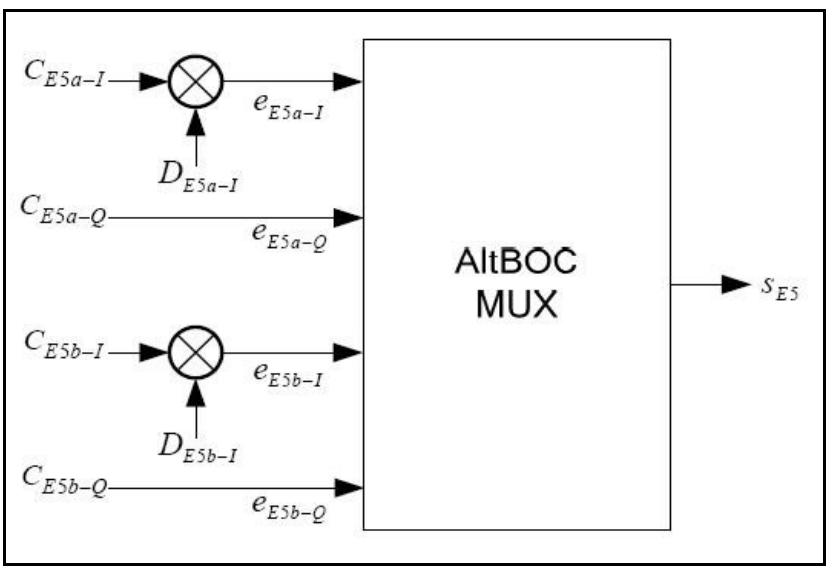


Figure 29: Signal E5 generation

As it can be appreciated in *figure 29*, there are all channels which are to be multiplexed in AltBOC. Data channels D_{E5aI} and D_{E5bI} will transmit at a rate of 50 symbols per second (sps) and 250 sps respectively.

$$e_{E5a-I}(t) = \sum_{i=-\infty}^{+\infty} \left[c_{E5a-I,|i|_{LE5a-I}} d_{E5a-I,[i]_{DCE5a-I}} rect_{T_{C,E5a-I}}(t - iT_{C,E5a-I}) \right]$$

$$e_{E5a-Q}(t) = \sum_{i=-\infty}^{+\infty} \left[c_{E5a-Q,|i|_{LE5a-Q}} rect_{T_{C,E5a-Q}}(t - iT_{C,E5a-Q}) \right]$$

$$e_{E5b-I}(t) = \sum_{i=-\infty}^{+\infty} \left[c_{E5b-I,|i|_{LE5b-I}} d_{E5b-I,[i]_{DCE5b-I}} rect_{T_{C,E5b-I}}(t - iT_{C,E5b-I}) \right]$$

$$e_{E5b-Q}(t) = \sum_{i=-\infty}^{+\infty} \left[c_{E5b-Q,|i|_{LE5b-Q}} rect_{T_{C,E5b-Q}}(t - iT_{C,E5b-Q}) \right]$$

Figure 30: Signal binary components

Accordingly, a single data channel (equivalent to a BPSK signal) and a pilot channel (another BPSK signal) will be transmitted in each sideband. The AltBOC modulated signal is then equivalent to two separate QPSK modulations (8-PSK see *figure 31.a*), placed respectively around the E5a and E5b centre frequency.

The Galileo E5a-signal is an inherent element of the E5-signal and consists of a data-component transmitted in the in-phase component and a pilot-component transmitted in the quadrature component. The E5a-signal provides the Freely accessible NAVigation message (F/NAV) message supporting Galileo Open Service and overlaps (in the spectrum) with the GPS-L5-signal.

The Galileo E5b-signal is an inherent element of the E5-signal and consists of a data-component transmitted in the in-phase component and a pilot-component transmitted in the quadrature component. The E5b-signal provides the Integrity NAVigation message (I/NAV) message and supports Safety of Life service, Galileo system integrity and Open Service.

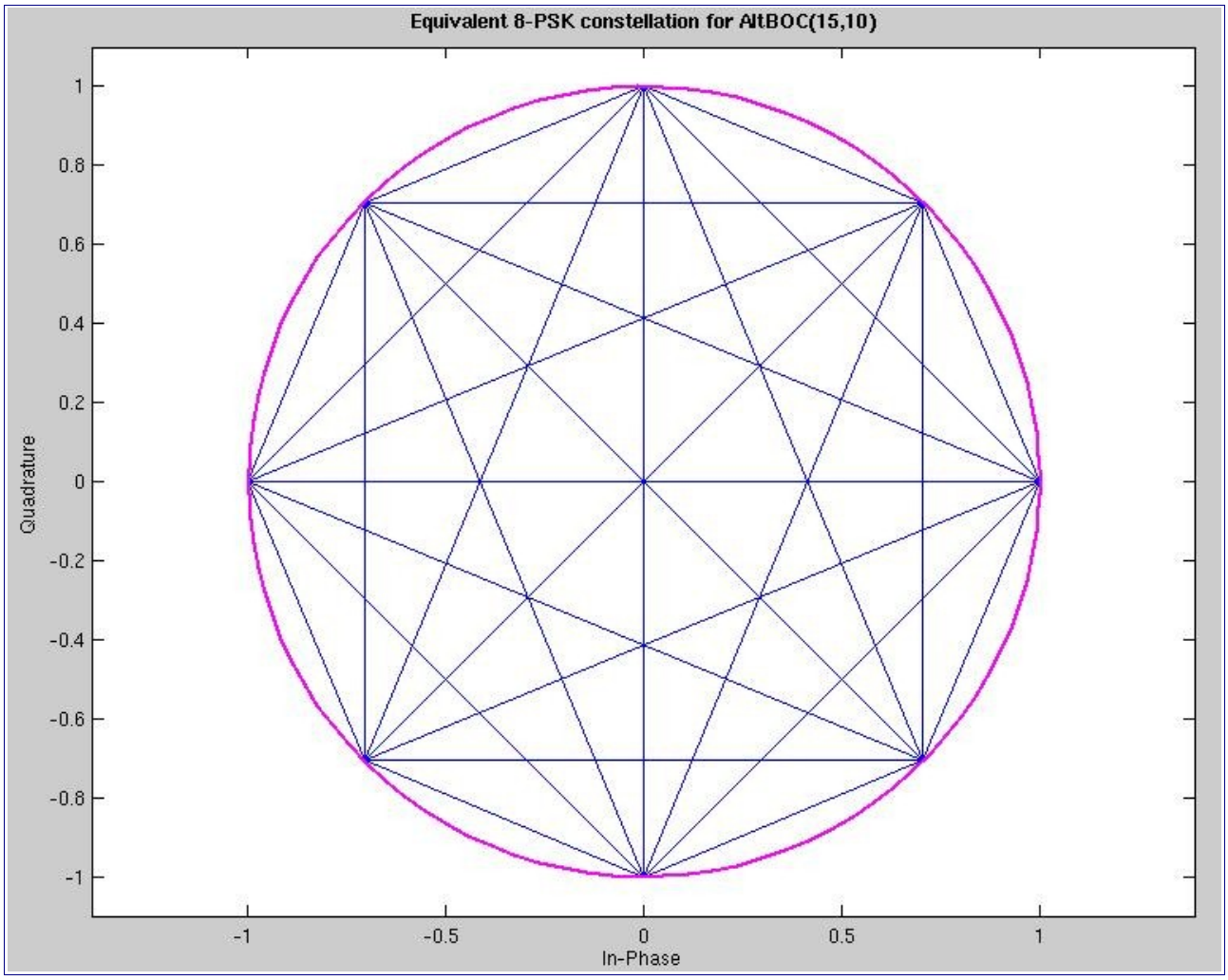


Figure 31.a: Constellation of E5 signal

The output of the multiplexer provides the signal E5 in the format that can be observed in *figure 31.b*.

$$s_{E5}(t)=\frac{1}{2\sqrt{(2)}}\left(e_{E5a-I}(t)+j\ e_{E5a-Q}(t)\right)\left[sc_{E5-S}(t)-j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]+$$

$$\frac{1}{2\sqrt{(2)}}\left(e_{E5b-I}(t)+j\ e_{E5b-Q}(t)\right)\left[sc_{E5-S}(t)+j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]+$$

$$\frac{1}{2\sqrt{(2)}}\left(\overline{e}_{E5a-I}(t)+j\ \overline{e}_{E5a-Q}(t)\right)\left[sc_{E5-P}(t)-j\ sc_{E5-P}(t-\frac{T_{s,E5}}{4})\right]+$$

$$\frac{1}{2\sqrt{(2)}}\left(\overline{e}_{E5b-I}(t)+j\ \overline{e}_{E5b-Q}(t)\right)\left[sc_{E5-P}(t)+j\ sc_{E5-P}(t-\frac{T_{s,E5}}{4})\right]$$

Figure 31.b: E5 modulation type

The dashed signal components represent product signals according to *figure 32*. Each of the dashed components is the multiplication of all the others channels except itself:

$$\bar{e}_{E5a-I} = e_{E5a-Q}\, e_{E5b-I}\, e_{E5b-Q} \qquad \bar{e}_{E5b-I} = e_{E5b-Q}\, e_{E5a-I}\, e_{E5a-Q}$$

$$\bar{e}_{E5a-Q} = e_{E5a-I}\, e_{E5b-I}\, e_{E5b-Q} \qquad \bar{e}_{E5b-Q} = e_{E5b-I}\, e_{E5a-I}\, e_{E5a-Q}$$

Figure 32: Product signals

And the parameters $sc_{E5\text{-}P}$ and $sc_{E5\text{-}S}$ represent the four-valued sub-carrier functions for the single signals and the product signals respectively as shown in *figure 33*.

$$sc_{E5-S}(t) = \sum_{i=-\infty}^{+\infty} AS_{|i|_8}\, rect_{T_{S,E5}/8}\left(t - \frac{iT_{s,E5}}{8}\right)$$

$$sc_{E5-P}(t) = \sum_{i=-\infty}^{+\infty} AP_{|i|_8}\, rect_{T_{S,E5}/8}\left(t - \frac{iT_{s,E5}}{8}\right)$$

i	0	1	2	3	4	5	6	7
ASi	$\frac{\sqrt{(2)}+1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{\sqrt{(2)}+1}{2}$	$-\frac{\sqrt{(2)}+1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{\sqrt{(2)}+1}{2}$
APi	$\frac{-\sqrt{(2)}+1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{\sqrt{(2)}-1}{2}$	$\frac{\sqrt{(2)}-1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{-\sqrt{(2)}+1}{2}$

Figure 33: sub-carrier functions and parameters

A representation of a Matlab plot of these two signals can be seen in the next figure:

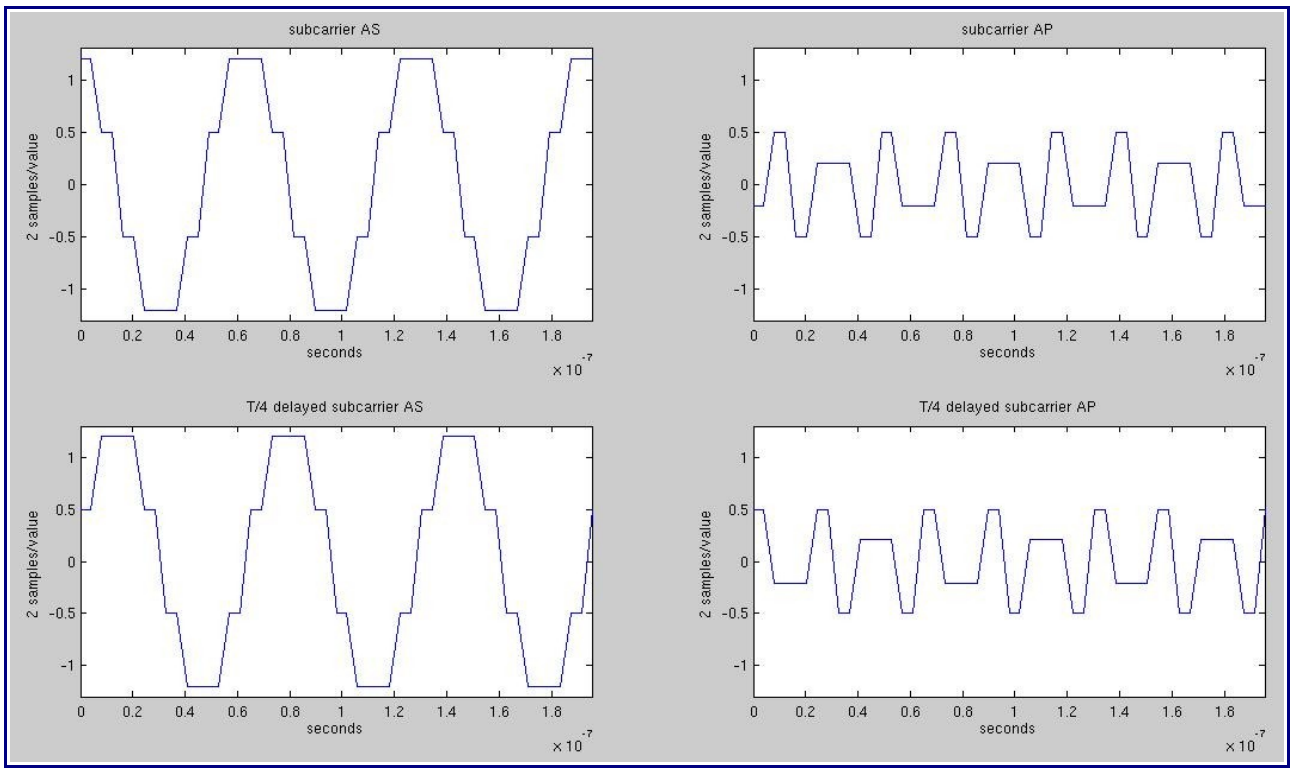


Figure 34: sub-carriers representation

The inclined slopes are due to hardware limitations. These plots were taken with a sampling frequency of 240·fo (fo=1.023 MHz or Mcps), which means that a lot of memory is used to store all the data. So I decided to do it with 2 samples per value. At the beginning, the sampling frequency needed not to be so high because the sub-carrier rate and chip rate were 15·fo and 10·fo respectively, and a sampling frequency of 90·fo would have been enough. However, the fact that these sub-carriers are changing their value every eight of sub-carrier period (2Ts), makes it impossible to maintain a sampling frequency so low. Therefore, one solution I found was to increase the sampling frequency up to 240·fo.

As E5 bandwidth is very large, it is interesting to take advantage of it using large band signals, like the one used in this part which comprises two adjacent bands E5A and E5B. Furthermore, making use of the AltBOC modulation give several advantages such as **it is only needed one single chain to transmit the four components**, it is a **constant modulus envelope signal** (see *figure 31.a*), it is a broad reception signal and both sidebands can be processed independently by the user receiver. Nonetheless, it also has bad points such as the difficult implementation and intermodulation products appear due to the dashed components.

Some results of the simulation in Matlab can be seen in the following plots:

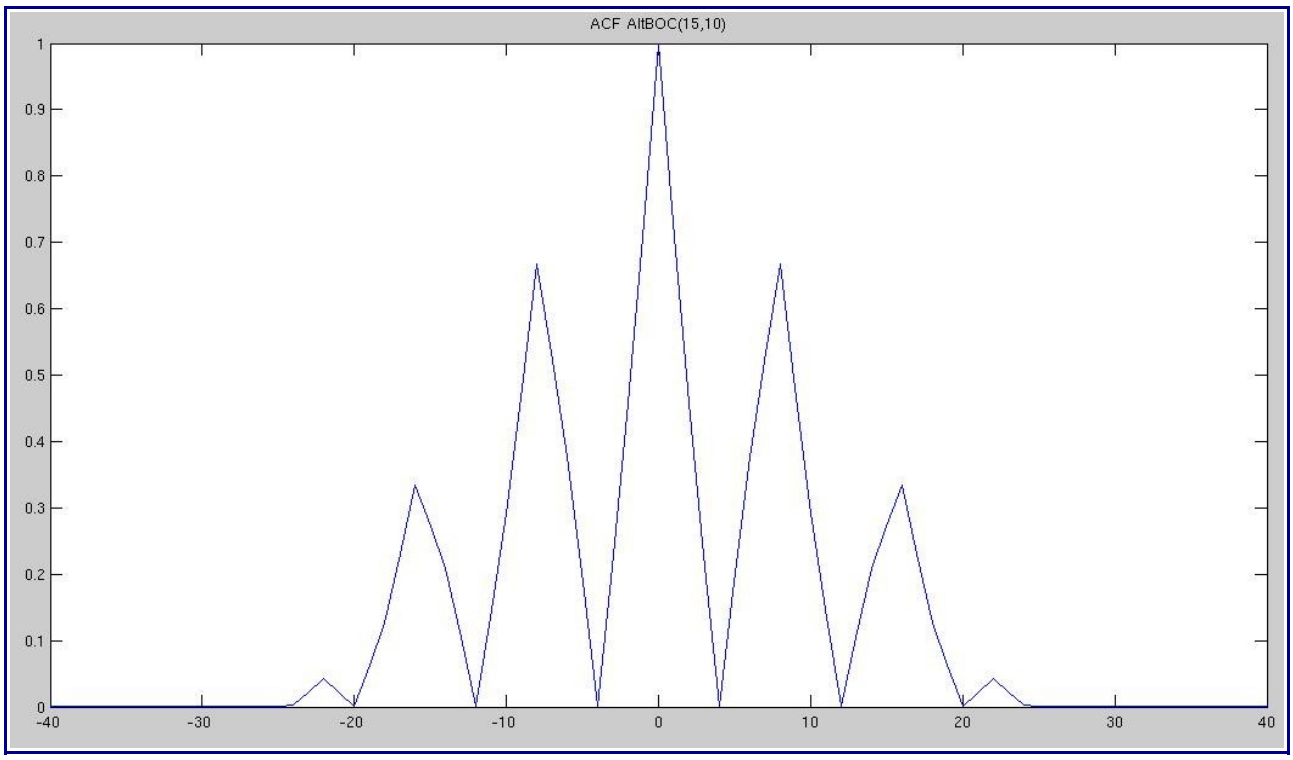


Figure 35: ACF of AltBOC(15,10) with parameter functions (81 samples out of 9820799)

As *figure 35* shows, the E5 signal ACF normalised absolute value is formed by seven peaks. This plot does not match with what it is expected in [4], since what we have here it is not only one BOC modulation, but two of them and they are not square-wave signals. Moreover, the way they are multiplexed may vary the result of the ACF. That is it, if we do the same with square-wave sub-carriers, the result is the same, though the peaks are more defined and narrower (see *figure 36*). However, it does match with what it is shown in [7], although the plot provided is not in absolute value (but it has the same number of peaks).

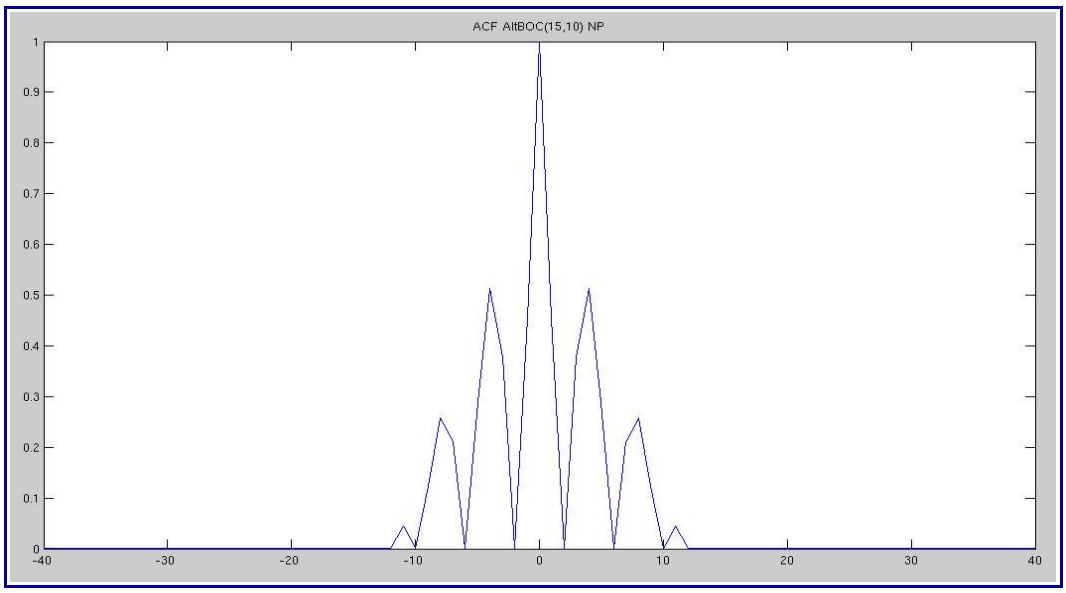


Figure 36: ACF of AltBOC(15,10) with no parameter functions (81 samples out of 9820799)

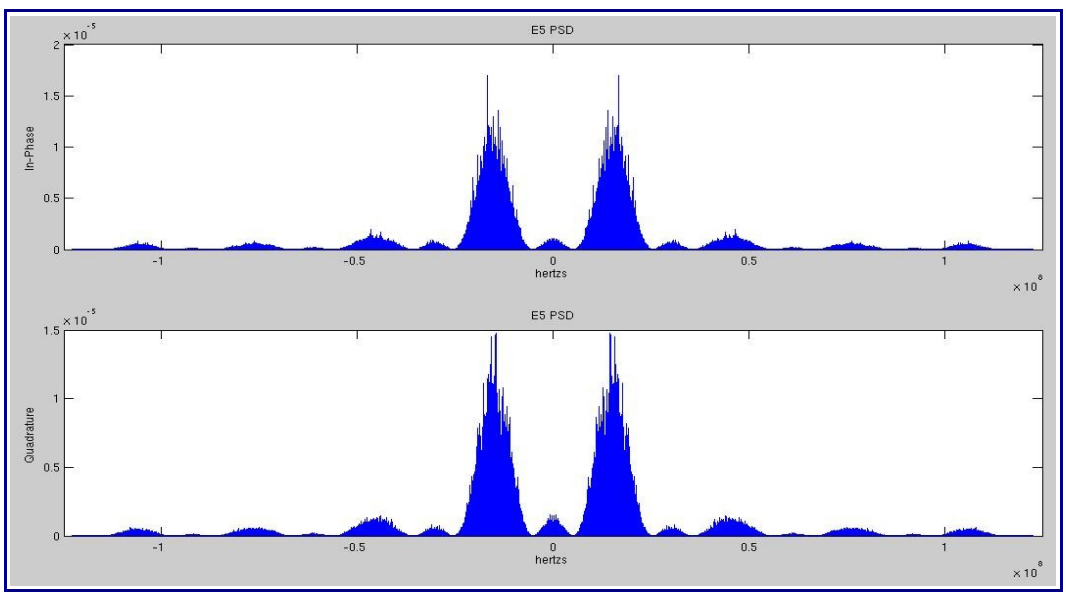


Figure 37: E5 PSD separately represented in I & Q

Figure 37 shows the four channels existing in E5 signal PSD. Note how the two main lobes are centred in 15·fo, as it is the sub-carrier function. Doing so, the split of the spectrum in two separate sidebands allows E5 band to be used as two separate sidebands.

The shape of this spectrum is quite peculiar (it has not the same shape than a normal BOC) due to the way the E5 signal is multiplexed (and modulated). In *figure 31.b* the expression of the signal is composed by 4 addends. Each of the addends can be divided in a real and imaginary part, since there is the data (real) and the pilot (imaginary) multiplied by the same sub-carrier function (but one of the two sub-carriers is delayed), one in the real axe and the other one in the imaginary axe. So, the first two addends correspond to the first (low amplitude), second (big one), third (low amplitude) and last lobe (very low) of both the real and imaginary components as can be seen in *figure 38* and the subsequent mathematical expressions.

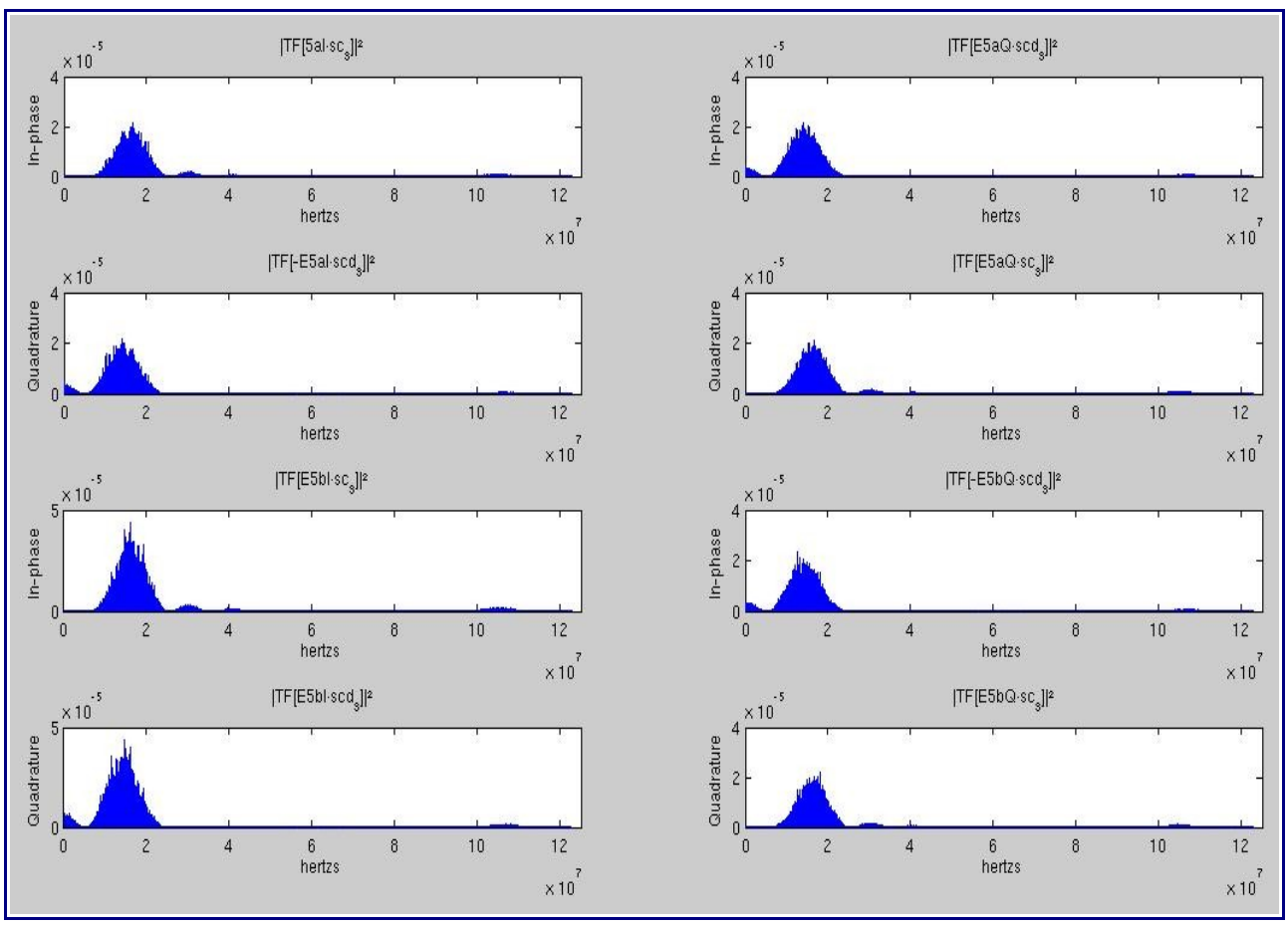


Figure 38: PSD of the first and second addends of the E5 signal

1st *addend*

$$\Re: \quad e_{E5a-I(t)} \cdot sc_{E5-S}(t) \; + \; e_{E5a-Q(t)} \cdot sc_{E5-S}(t - T_{SE5}/4)$$

$$\Im: \quad -e_{E5a-I(t)} \cdot sc_{E5-S}(t - T_{SE5}/4) \; + \; e_{E5a-Q(t)} \cdot sc_{E5-S}(t)$$

2nd *addend*

$$\Re: \quad e_{E5b-I(t)} \cdot sc_{E5-S}(t) \; - \; e_{E5b-Q(t)} \cdot sc_{E5-S}(t - T_{SE5}/4)$$

$$\Im: \quad e_{E5b-I(t)} \cdot sc_{E5-S}(t - T_{SE5}/4) \; + \; e_{E5b-Q(t)} \cdot sc_{E5-S}(t)$$

3rd *addend*

$$\Re: \quad \overline{e}_{E5a-I(t)} \cdot sc_{E5-P}(t) \; + \; \overline{e}_{E5a-Q(t)} \cdot sc_{E5-P}(t - T_{SE5}/4)$$

$$\Im: \quad -\overline{e}_{E5a-I(t)} \cdot sc_{E5-P}(t - T_{SE5}/4) \; + \; \overline{e}_{E5a-Q(t)} \cdot sc_{E5-P}(t)$$

4th *addend*

$$\Re: \quad \overline{e}_{E5b-I(t)} \cdot sc_{E5-P}(t) \; - \; \overline{e}_{E5b-Q(t)} \cdot sc_{E5-P}(t - T_{SE5}/4)$$

$$\Im: \quad \overline{e}_{E5b-I(t)} \cdot sc_{E5-P}(t - T_{SE5}/4) \; + \; \overline{e}_{E5b-Q(t)} \cdot sc_{E5-P}(t)$$

So, as it can be seen in the expression above, every addend has one real and imaginary part, and each real and imaginary part of each addend has a data and pilot component (I & Q) modulated by the corresponding parametric sub-carrier. Therefore, as can be appreciated in *figure 37*, the first two addends form the big lobes and the last two addends form the little lobes.

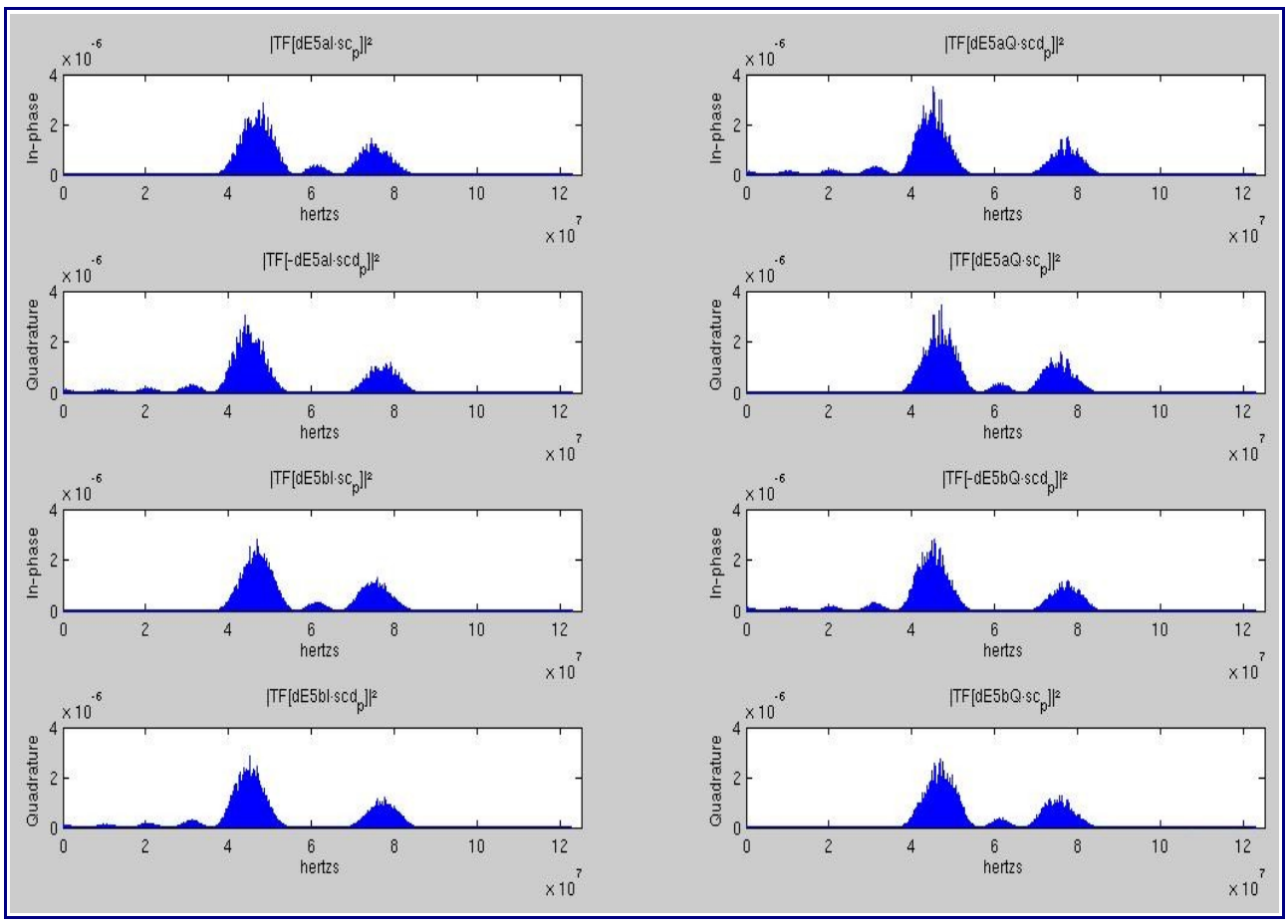


Figure 39: PSD of the third and fourth addend of E5 signal

The other two addends, which are composed by a combination of three of the four binary signals, are the ones which represent the rest of the lobes of lower amplitude (they are ten times lower approximately) that are located between the first main lobe and the last lobe (*figure 39)*. The third and fourth addends can be also expressed separately in the real and imaginary part.

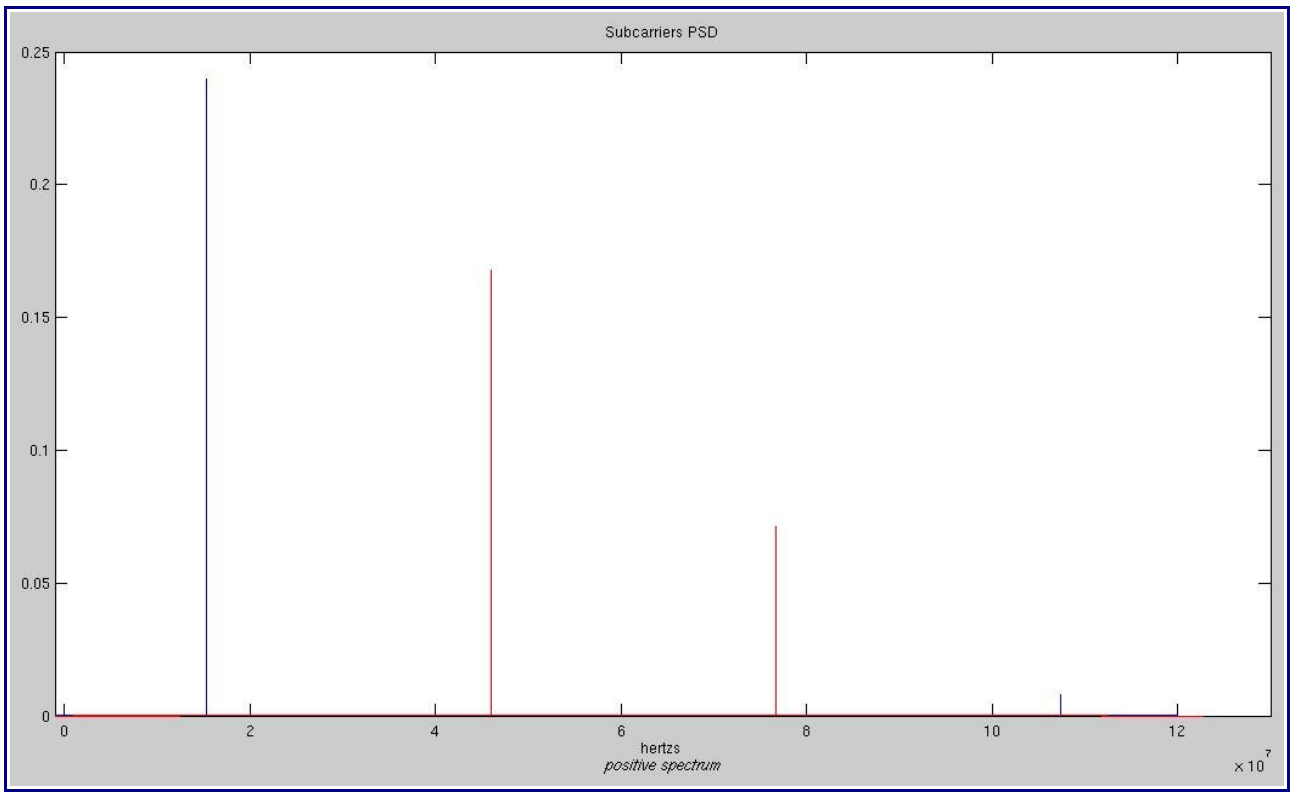


Figure 40.a: PSD of sub-carriers with parameters (only positive spectrum)

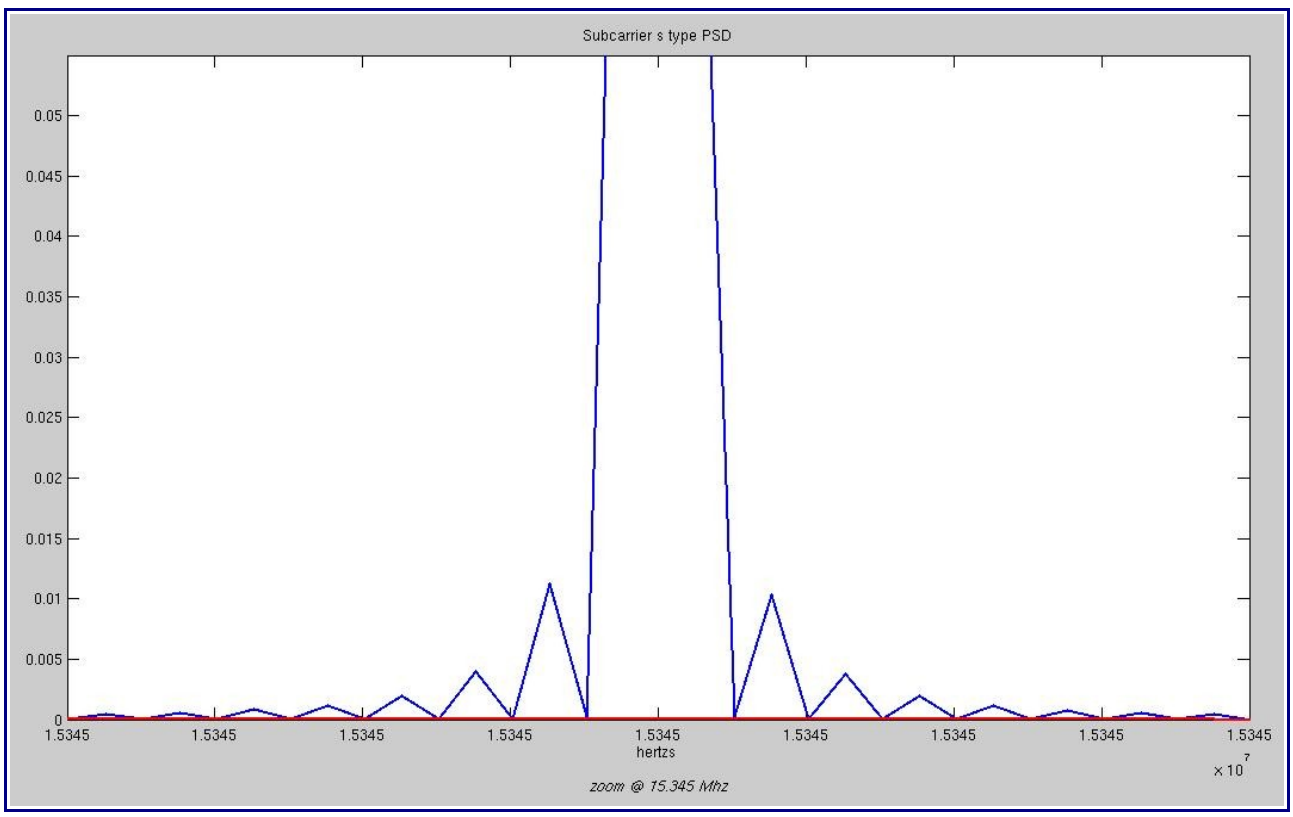


Figure 40.b: Zoom of figure 41.a at 15.345 MHz

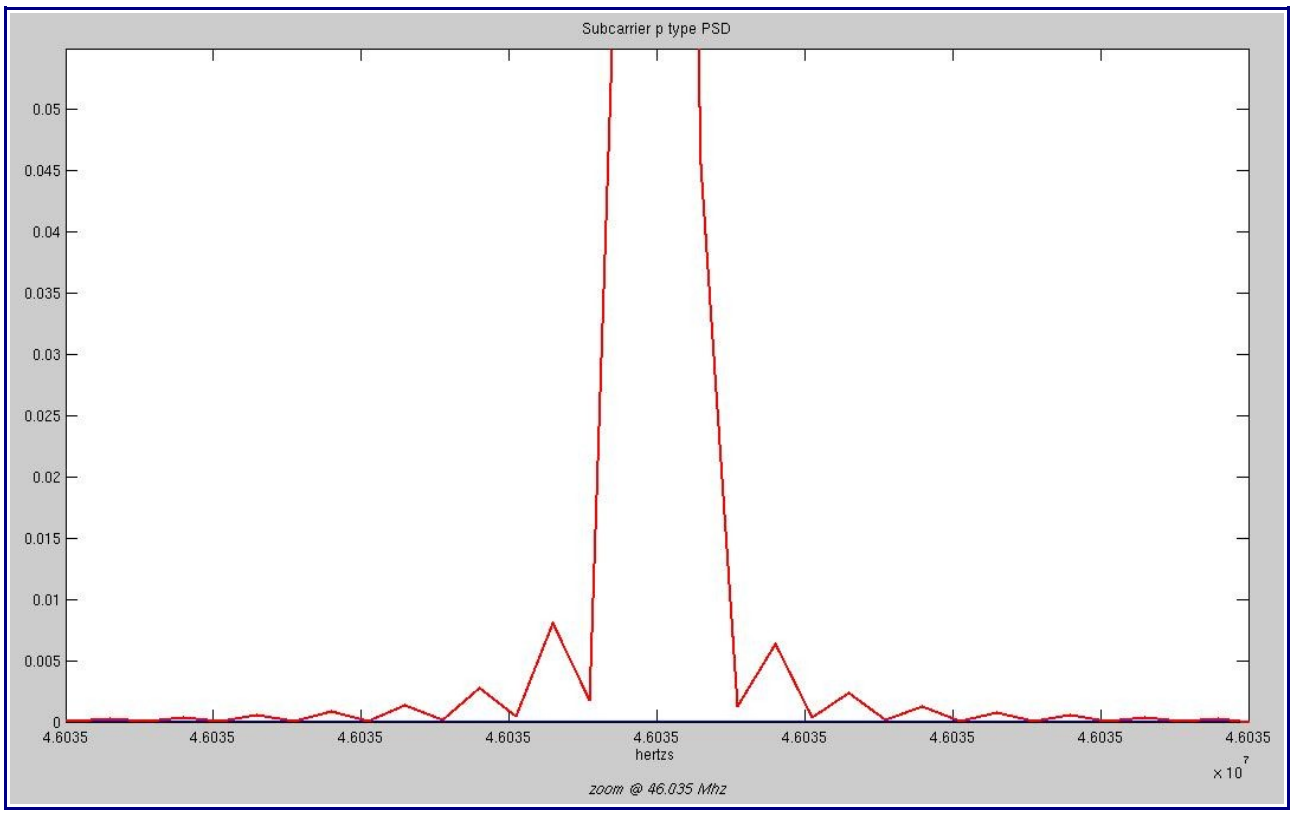


Figure 40.c: Zoom of figure 40.a at 46.035 MHz

If we have a look to the Fourier transform of the sub-carriers in *figure 40.a*, it can be appreciated how the deltas are separated every odd multiple of 15.345 MHz from 0 to 107.415 MHz (15.345, 46.035, 76.725 and 107.415 MHz). For instance, the main harmonics corresponding to the ***s*** type sub-carrier are the ones situated at 15.345 MHz and 107.415 MHz, whereas for the ***p*** type sub-carrier are the ones situated at 46.035 MHz and 76.725 MHz. Therefore, as it has been said in ***2.2.1***, the resulting spectra will be a main lobe (combination of all channels) convoluted with the different deltas at different frequencies.

Add to this, in *figures 40.b* and *40.c* it can also be appreciated how little ripples appear due to the so-called abrupt change of slope of the sub-carrier waveforms in time (*figure 34)*.

Retaking signal E5, if we rewrite the two first addends from expression in *figure 31.b* and as it has been shown in section 2.2.2, we can appreciate, more intuitively, how the four different channels are modulated and located in the spectrum:

$$\underbrace{2\cdot\sqrt{2}\cdot s_{E5}(t)}_{\text{Only two first addends}} = (e_{E5a-I}(t)+j\ e_{E5a-Q}(t))\cdot\left[sc_{E5-S}(t)-j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]+$$

$$\left(e_{E5b-I}(t)+j\ e_{E5b-Q}(t)\right)\cdot\left[sc_{E5-S}(t)+j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]=$$

$$=e_{E5a-I}(t)\cdot\left[sc_{E5-S}(t)-j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]+e_{E5a-Q}(t)\cdot\left[sc_{E5-S}(t-\frac{T_{s,E5}}{4})+j\ sc_{E5-S}(t)\right]+$$

$$e_{E5b-I}(t)\cdot\left[sc_{E5-S}(t)+j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]+e_{E5b-Q}(t)\cdot\left[-sc_{E5-S}(t-\frac{T_{s,E5}}{4})+j\ sc_{E5-S}(t)\right]$$

$$\textit{where:}\quad \left[sc_{E5-S}(t)+j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]=pcss(t)$$

then:

$$\underbrace{2\cdot\sqrt{2}\cdot s_{E5}(t)}_{\text{Only two first addends}} = e_{E5a-I}(t)\cdot p\check{c}ss(t)+j\cdot e_{E5a-Q}(t)\cdot\left[sc_{E5-S}(t)-j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]+$$

$$e_{E5b-I}(t)\cdot pcss(t)+j\cdot e_{E5b-Q}(t)\cdot\left[sc_{E5-S}(t)+j\ sc_{E5-S}(t-\frac{T_{s,E5}}{4})\right]=$$

$$=e_{E5a-I}(t)\cdot p\check{c}ss(t)+j\cdot e_{E5a-Q}(t)\cdot p\check{c}ss(t)+e_{E5b-I}(t)\cdot pcss(t)+j\cdot e_{E5a-Q}(t)\cdot pcss(t)=$$

$$=e_{E5a-I}(t)\cdot p\check{c}ss(t)+e_{E5b-I}(t)\cdot pcss(t)+j\cdot\left\{e_{E5a-Q}(t)\cdot p\check{c}ss(t)+e_{E5b-Q}(t)\cdot pcss(t)\right\}$$

With this last expression, it can be clearly seen how both data channels E5aI and E5bI are in the in-phase component and how the pilot channels, E5aQ and E5bQ, are in the quadrature component. If we do the same with the other two addends left, we will obtain:

$$\underbrace{2\cdot\sqrt{2}\cdot s_{E5}(t)}_{\text{Only two second addends}} = \bar{e}_{E5a-I}(t)\cdot p\check{c}sp(t)+\bar{e}_{E5b-I}(t)\cdot pcsp(t)+j\cdot\left\{\bar{e}_{E5a-Q}(t)\cdot p\check{c}sp(t)+\bar{e}_{E5b-Q}(t)\cdot pcsp(t)\right\}$$

$$\textit{where:}\quad \left[sc_{E5-P}(t)+j\ sc_{E5-P}(t-\frac{T_{s,E5}}{4})\right]=pcsp(t)$$

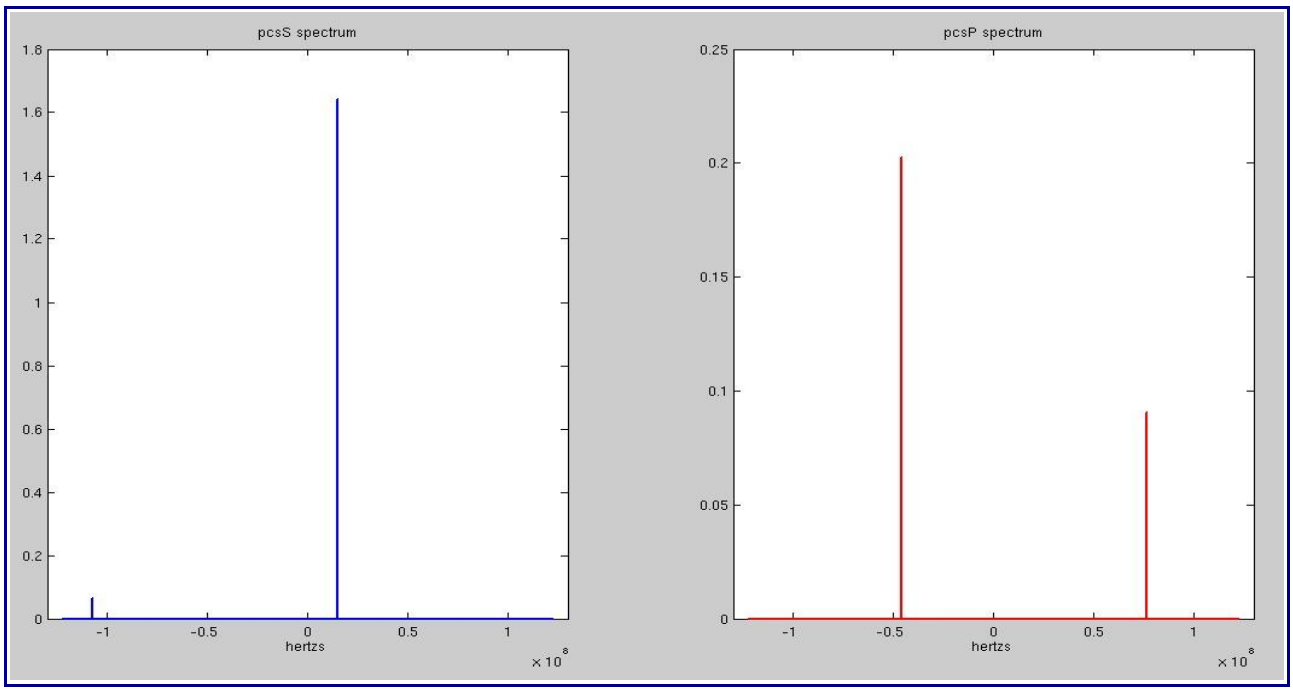


Figure 41: PSD of pcss(t) and pcsp(t) functions separately

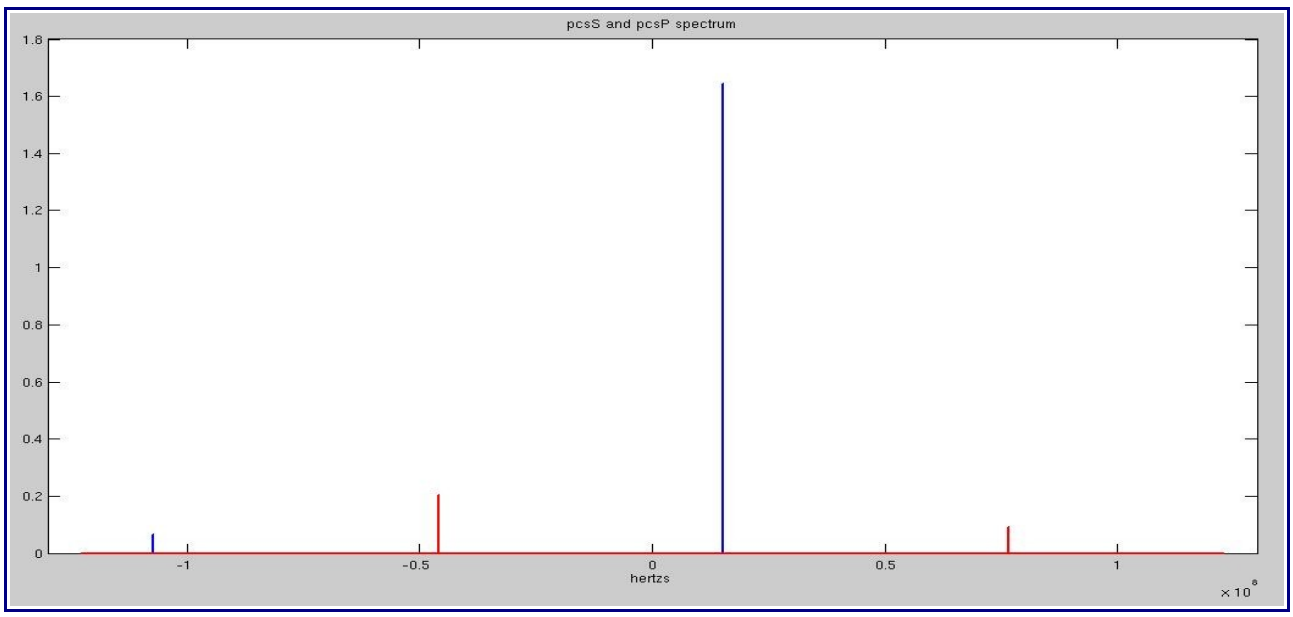


Figure 42.a: PSD of pcss(t) and pcsp(t) functions together

In both last pictures it can be seen how the blue spectrum is the one used for data channels and pilot channels (E5aI, E5bI, E5aQ and E5bQ) and the red spectrum is the one assigned to the dashed signals (the main harmonic is situated at 15.345 MHz).

In this case, if we zoom *figure 42.a* we do not see the ripple that appears in *figure 40.b*. The transformed function is not the same, it is composed by two sub-carriers, one of them in the real

part and the other one, which is the same sub-carrier but delayed, in the imaginary part. It looks like a complex exponential, although it is not, and its Fourier transform corresponds to a pair of deltas spaced in frequency, this might be one of the reasons why the ripples disappears in this transformation.

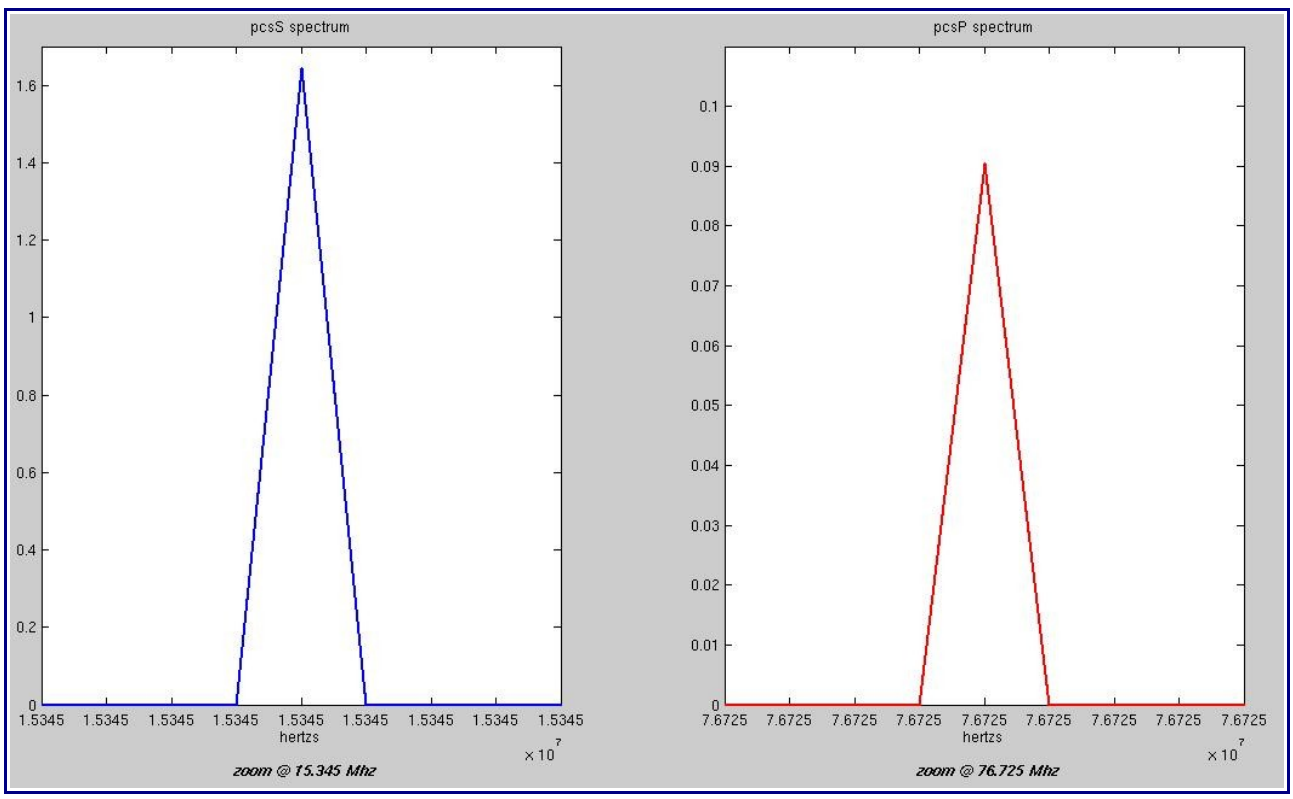


Figure 42.b: Zoom for figure 41

Finally, if all four addends are joined in the same equation:

$$s_{E5}(t) = \frac{1}{2\cdot\sqrt{2}}\cdot e_{E5a-I}(t)\cdot p\check{c}ss(t)+e_{E5b-I}(t)\cdot pcss(t)+\overline{e}_{E5a-I}(t)\cdot p\check{c}sp(t)+\overline{e}_{E5b-I}(t)\cdot pcsp(t) \; +$$

$$\frac{1}{2\cdot\sqrt{2}}\cdot j\cdot\left\{e_{E5a-Q}(t)\cdot p\check{c}ss(t)+e_{E5b-Q}(t)\cdot pcss(t)+\overline{e}_{E5a-Q}(t)\cdot p\check{c}sp(t)+\overline{e}_{E5b-Q}(t)\cdot pcsp(t)\right\}$$

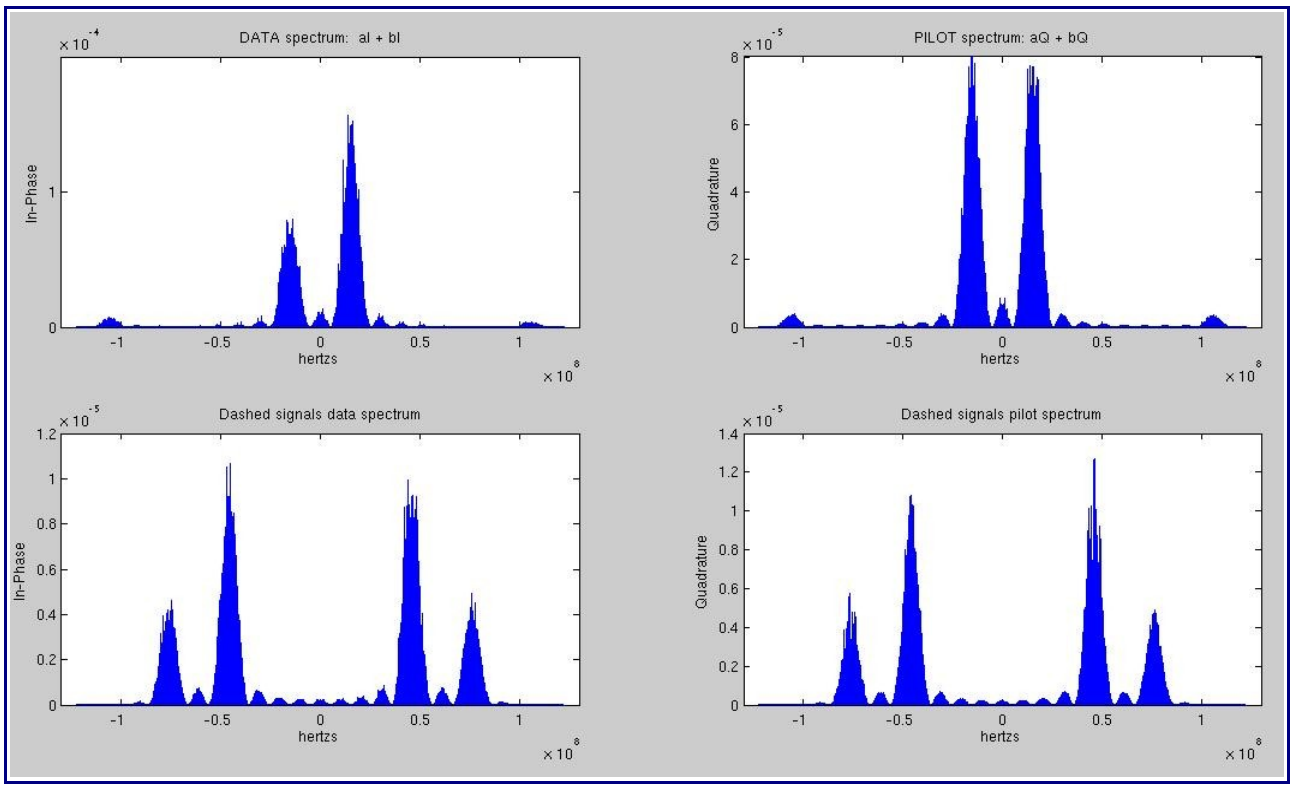


Figure 43: PSD of the 4 different channels (normal and dashed) separately

In this figure it can be seen how the last equation is represented in frequency. It shows all channels (E5aI, E5bI, E5aQ, E5bQ and the dashed channels) in their correct position.

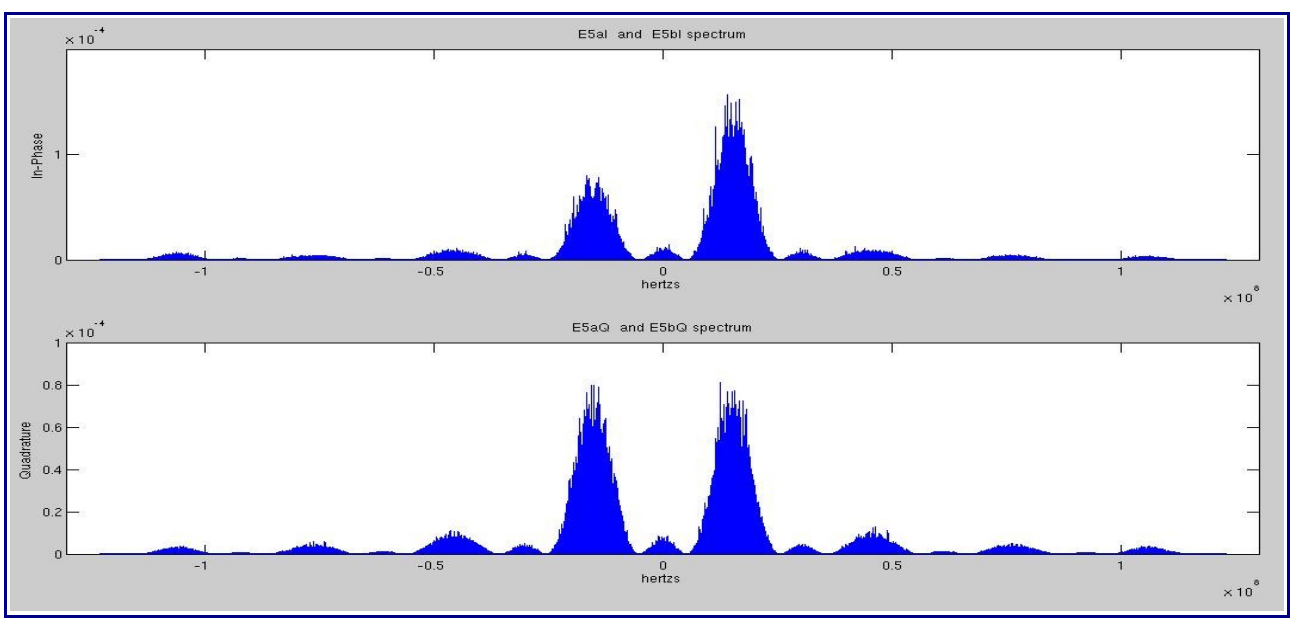


Figure 44: PSD of signal E5 showing in-phase and quadrature components with the 4 different channels

To end with this signal, we will show a series of plots regarding the constant envelope of this signal, such as the cumulative distribution function (CDF), the complementary cumulative distribution function (CCDF) and the peak-to-peak average power ratio (PAPR), thus demonstrating its constant envelope.

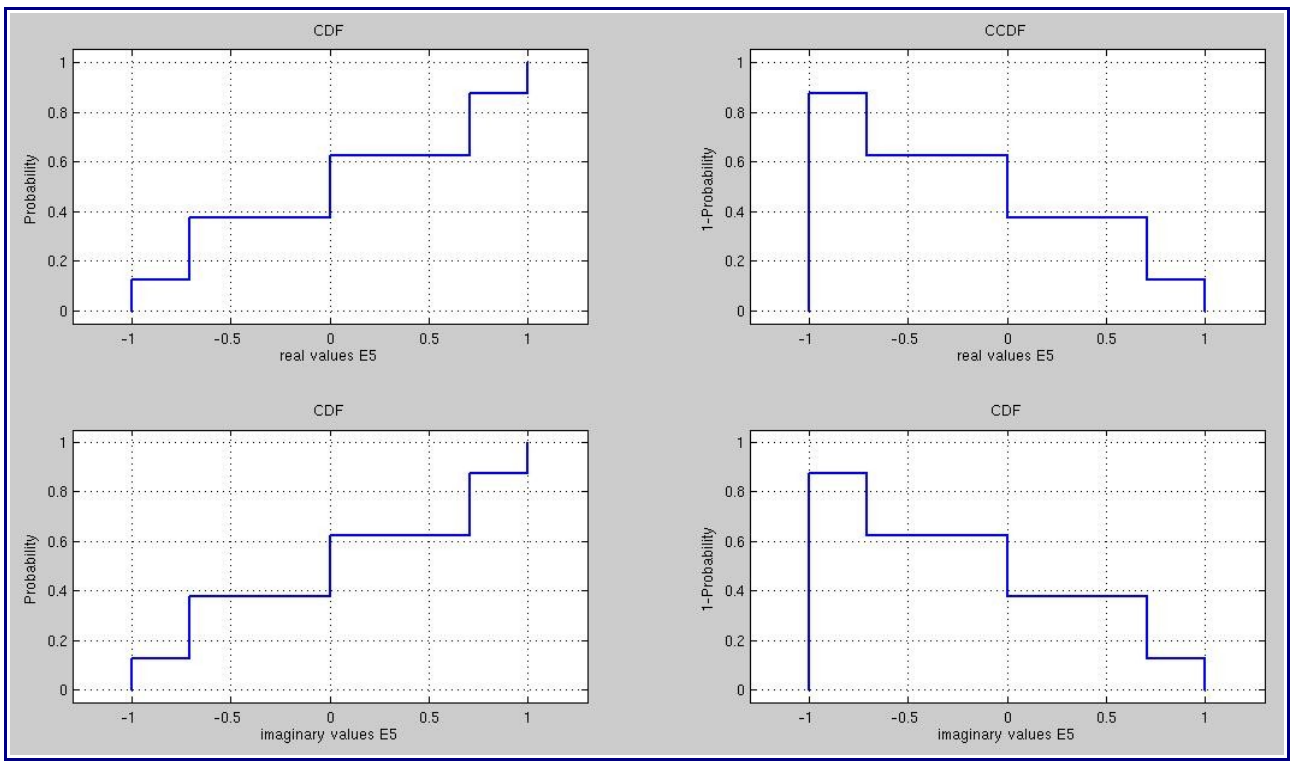


Figure 45: CDF and CCDF for E5 signal values separated in real and imaginary parts

In *figure 45* it can be seen the CDF and CCDF of the amplitude for E5 signal in real and imaginary parts. As it shows, all values are between -1 and +1 and getting *figure 31.a*, which shows a constellation with symbol placed around a circle, it is clear that its envelope is constant. Moreover, looking at *figure 46*, the PAPR is 1, that means that the power peak value and the average power are always the same, and consequently the CCDF is zero.

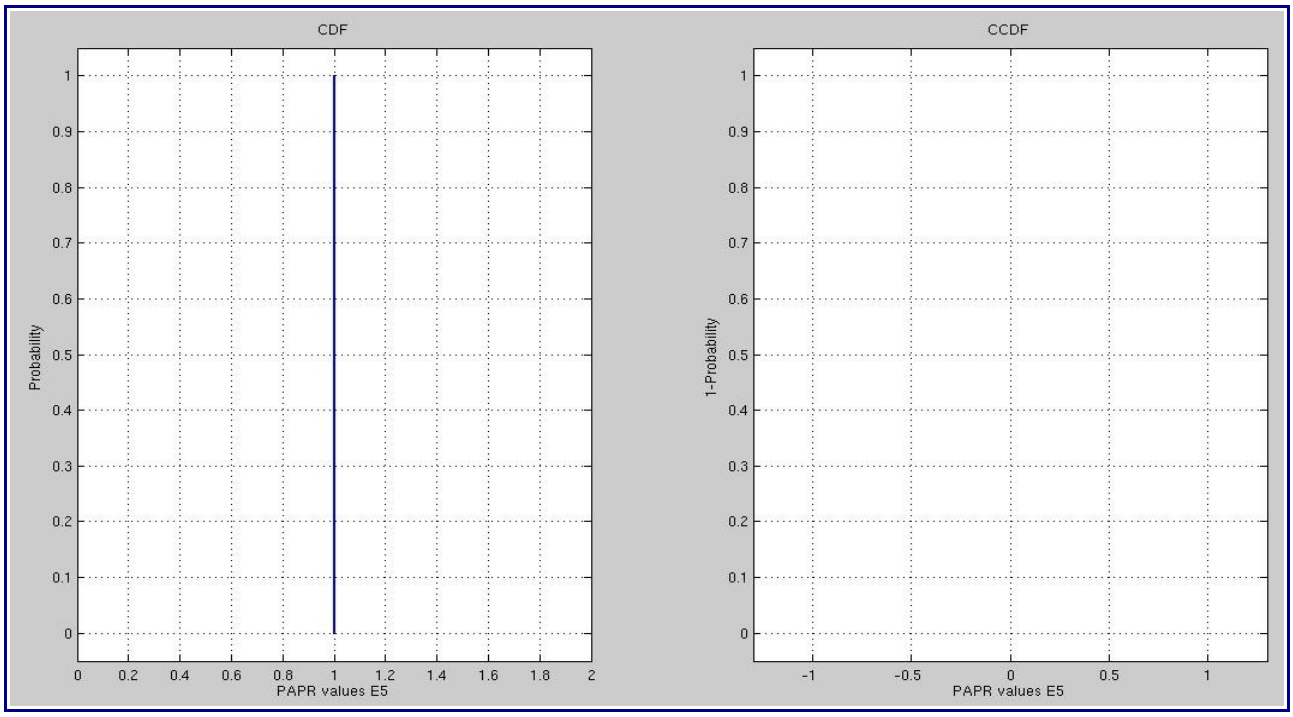


Figure 46: CDF and CCDF for E5 signal PAPR

4.2 E6 Signal

The Galileo E6-signal consists of the signal components E6-B and E6-C and is transmitted in the frequency band 1215 - 1300 MHz allocated on a worldwide co-primary basis (ITU-R Radio Regulations), sharing with radar systems of the radio navigation and radiolocation service. The signal components E6-B and E6-C are data-component and pilot-component respectively. The E6-signal provides the Commercial NAVigation message (C-NAV) message and supports Commercial Service [1].

The E6 signal is modulated by a BPSK at $5 \cdot fo$ Mcps and a symbol rate of 1000 Sps. Since this band is not used by either GPS or GLONASS there were not so many restrictions to select the modulation type [3].

So, the format of the E6 signal can be seen in the next three figures:

$$S_{E6}(t) = \frac{1}{\sqrt{(2)}}[e_{E6-B}(t) - e_{E6-C}(t)]$$

Figure 47: E6 signal

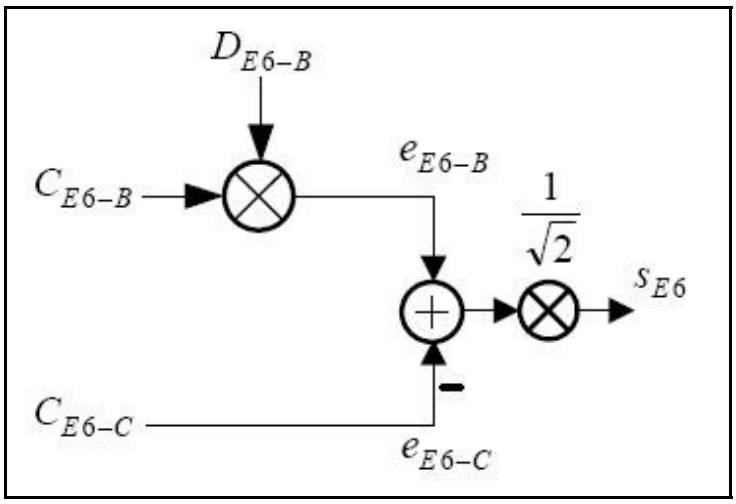


Figure 48: Multiplexing E6

$$e_{E6-B}(t)=\sum_{i=-\infty}^{+\infty}\left[c_{E6-B,|i|_{LE6-B}}\, d_{E6-B,[i]_{DCE6-B}}\, rect_{T_{C,E6-B}}(t-iT_{C,E6-B})\right]$$

$$e_{E6-C}(t)=\sum_{i=-\infty}^{+\infty}\left[c_{E6-C,|i|_{LE6-C}}\, rect_{T_{C,E6-C}}(t-iT_{C,E6-C})\right]$$

Figure 49: E6 components

E6P band (channel A) is restricted and no information is provided for this signal. It is only said that its modulation is a BOCcos(10,5). In order to simulate this signal it has been added all the necessary parameters with suitable values which make the simulation simpler. Some results of the Matlab simulation are shown in the next plots.

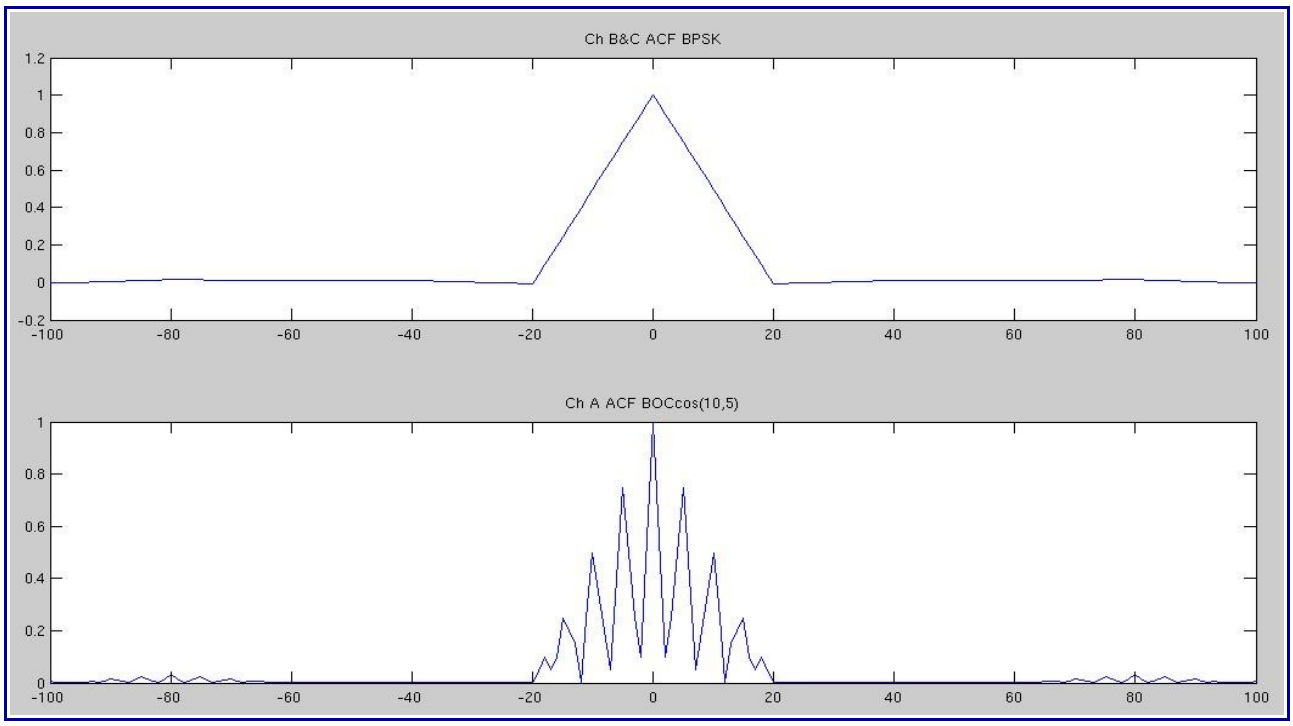


Figure 50: ACF of E6 channels B&C (BPSK) and A (BOCcos(10,5)) (201 samples out of 2045999)

It can be seen in this plot that the waveform of the ACF of both signals is the expected, since for a BPSK we have a triangle function, and for a BOCcos, the number of peaks is the same than for a BOCsin plus two: *2n +1*.

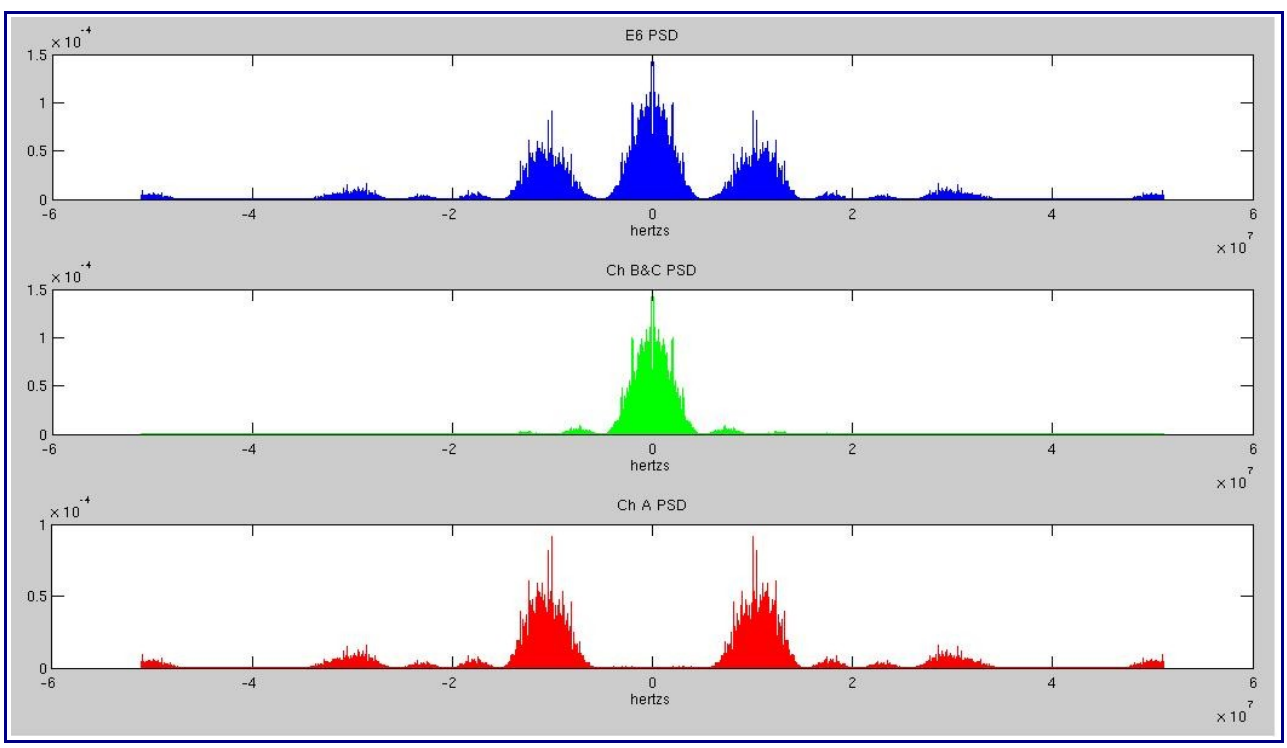


Figure 51: PSD of both bands E6P and E6C

It can be clearly seen clearly how both spectra are well represented. BPSK is the green one and BOCcos is the red one.

In the next plot it is represented the spectrum of the BPSK modulated at 30 MHz to have a clear vision.

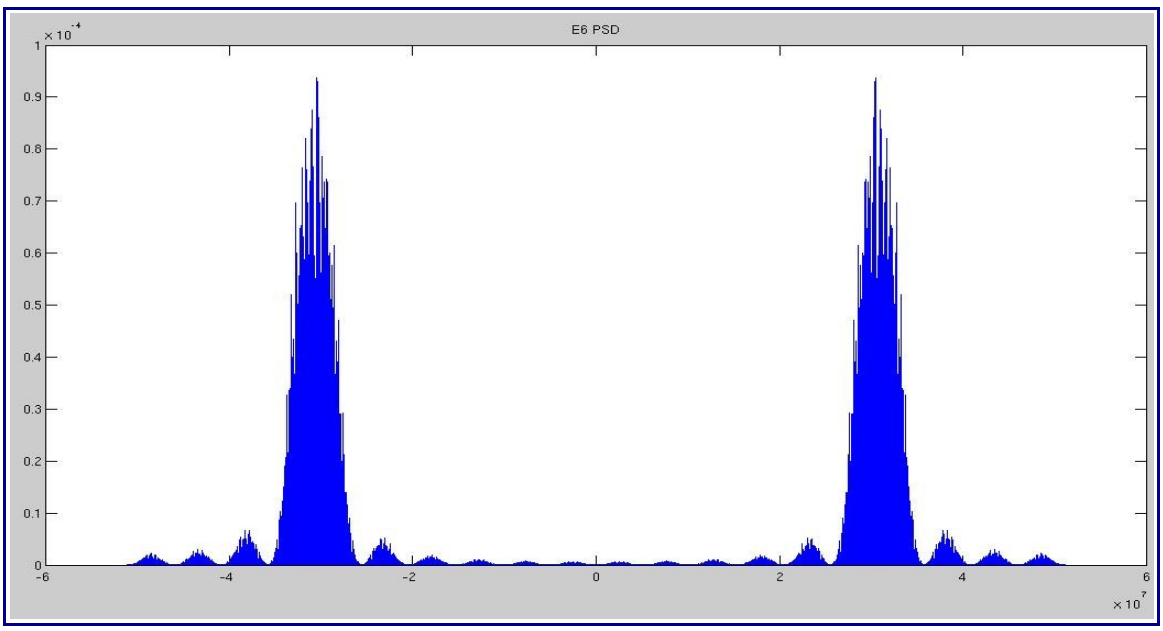


Figure 52: PSD of E6 B&C (BPSK) band modulated @ 30MHz

This signal has been simulated and implemented following [1], however, in that document there is neither information about channel A nor for the multiplexing technique it uses. Therefore, taking reference [11], which is not as updated as [1], we can find a multiplexing scheme that coincides with another references (such as [3]) and says that E6 signal is multiplexed with Coherent Adaptative sub-carrier Modulation (CASM) technique, which ensures, again, a constant envelope transmitting signal as seen in *section 2.2.4*. Add to this, it gives a detailed expression of E6 signal with which we can check the constant modulus envelope of this signal as well. Moreover, a new plot of the PSD for E6 signal will be given to compare it with the one above.

The expression provided by [11] is given in passband, then, in-phase and quadrature components are extracted so that the analysis can be done in baseband.

$$S_{E6}(t)=\left\{e_{E6-A}(t)\cdot\cos(m)-e_{E6-C}(t)\cdot\sin(m)\right\}+j\cdot\left\{e_{E6-B}(t)\cdot\cos(m)+e_{E6-A}(t)\cdot e_{E6-B}(t)\cdot e_{E6-C}(t)\cdot\sin(m)\right\}$$

$$where \quad m=0.6155\ radians$$

Figure 53: E6 signal expression with modulation index m.

With this expression containing all three channels we obtain this plot for its PSD:

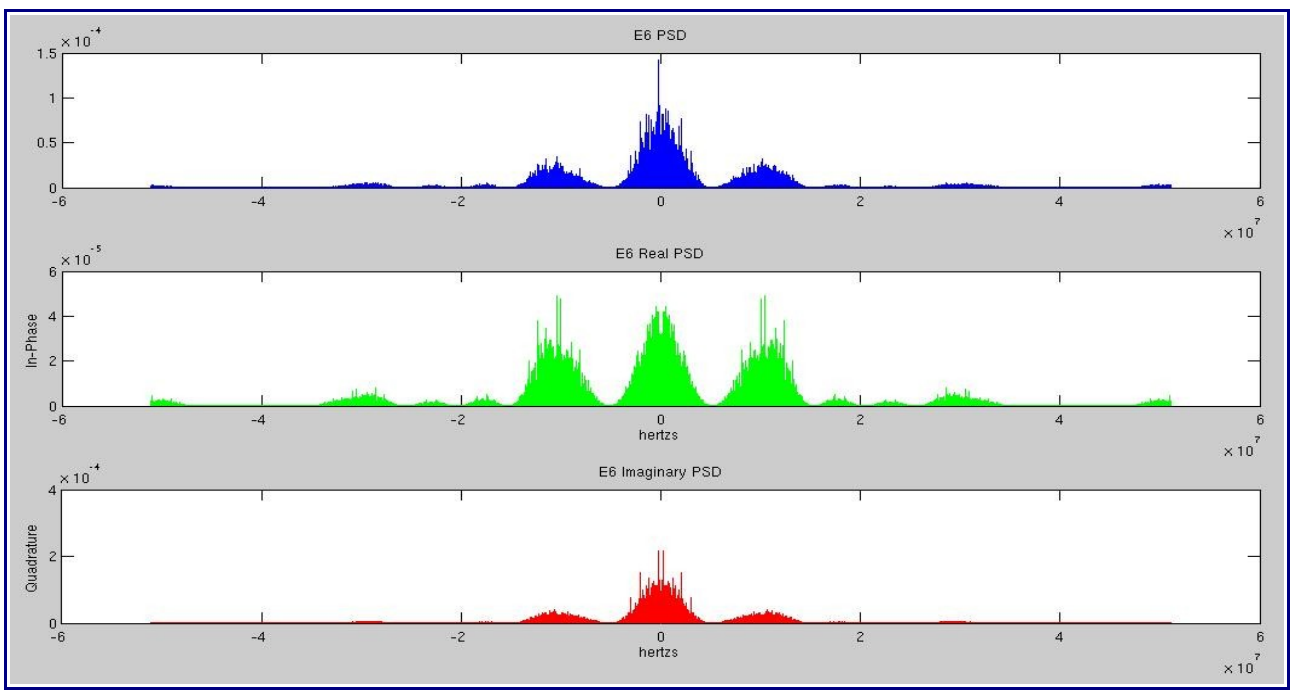


Figure 54: E6 signal PSD (blue), representing in-phase (green) and quadrature (red) components in frequency

From E6 expression it can be seen clearly the channel A (BOCcos) in green and channel C (BPSK) in green as well (the centred lobe), and then channel B (BPSK) in red (centred lobe) and the multiplication of the three channels added to channel B (side lobes in red) to achieve the constant modulus (as it has been done in E5 and it will be done in E1-L1 as well). The modulation index, *m*, is used to control the amplitude of the different channels, giving more power to a certain channel assigning the *cos(m)*, since for that value of *m,* the cosine is bigger than the sine (*cos(m)*=0.8165 ; *sin(m)*=0.5774).

Regarding the constant envelope, the same plots have been obtained as for E5 signal: CDF, CCDF, and constellation.

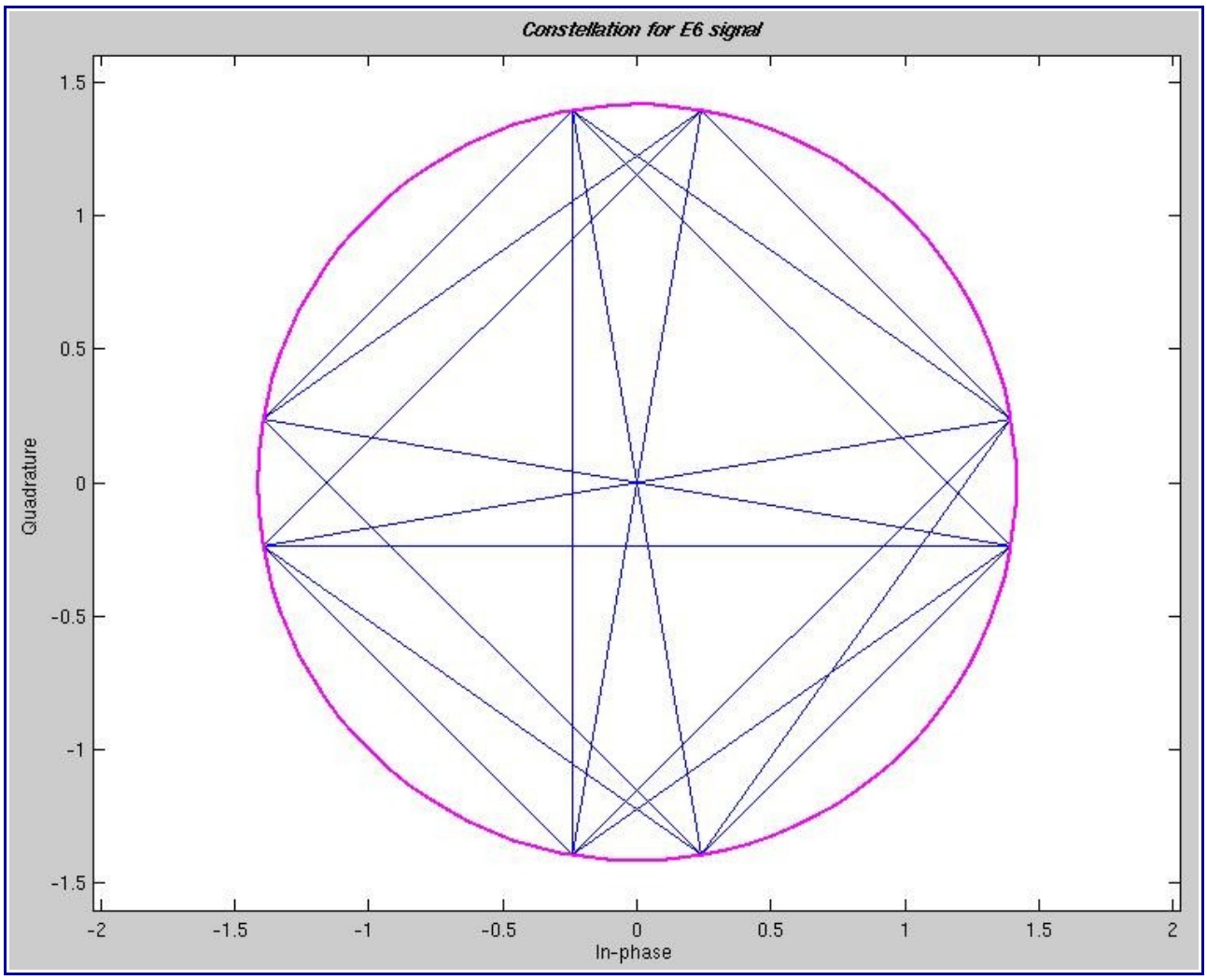


Figure 55: Constellation for E6 signal

Ch A	Ch B	Ch C	Re {s(t)}	Im {s(t)}	|sE6(t)|
1	1	1	0.2391	1.3938	2
1	1	-1	1.3938	0.2391	2
1	-1	1	0.2391	-1.3938	2
1	-1	-1	1.3938	-0.2391	2
-1	1	1	-1.3938	0.2391	2
-1	1	-1	-0.2391	1.3938	2
-1	-1	1	-1.3938	-0.2391	2
-1	-1	-1	-0.2391	-1.3938	2

As it is said in [11], this modulation is a modified Hexaphase, a QPSK signal resulting from the combination of two channels in phase modulated with the third channel and, as said before, the modulation index is used to set the relative power between the three channels. That is what we can observe in *figure 55* and in the table beneath it, the set of symbols placed in a circle of radius $r=\sqrt{2}$, thus reaffirming again the constant envelope. It can also be observed in the table the different combinations that the channels yield, thus giving the eight different symbols in the constellation plot (like the study of CASM in section 2.2.4).

It can be observed in the following plot, *figure 57*, that the values for this signal are located between $-\sqrt{2}$ and $+\sqrt{2}$, and that both real and imaginary values are distributed exactly in the same way for both in-phase and quadrature components. Finally, in *figure 58*, it is shown the CDF and CCDF for the PAPR. Again, there is only a line in the unit value, thus confirming the constant envelope behaviour.

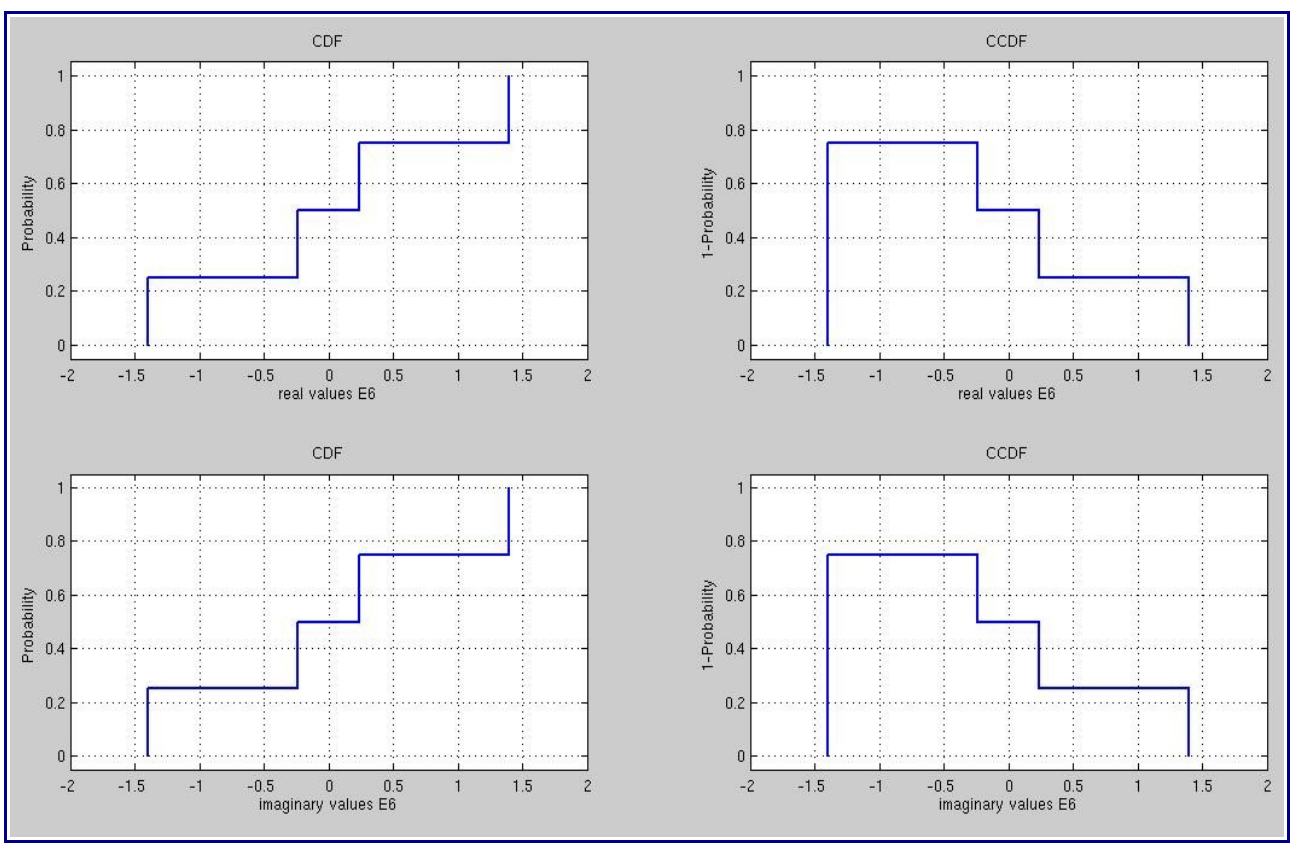


Figure 56: CDF and CCDF for E6 signal values separated in real and imaginary parts

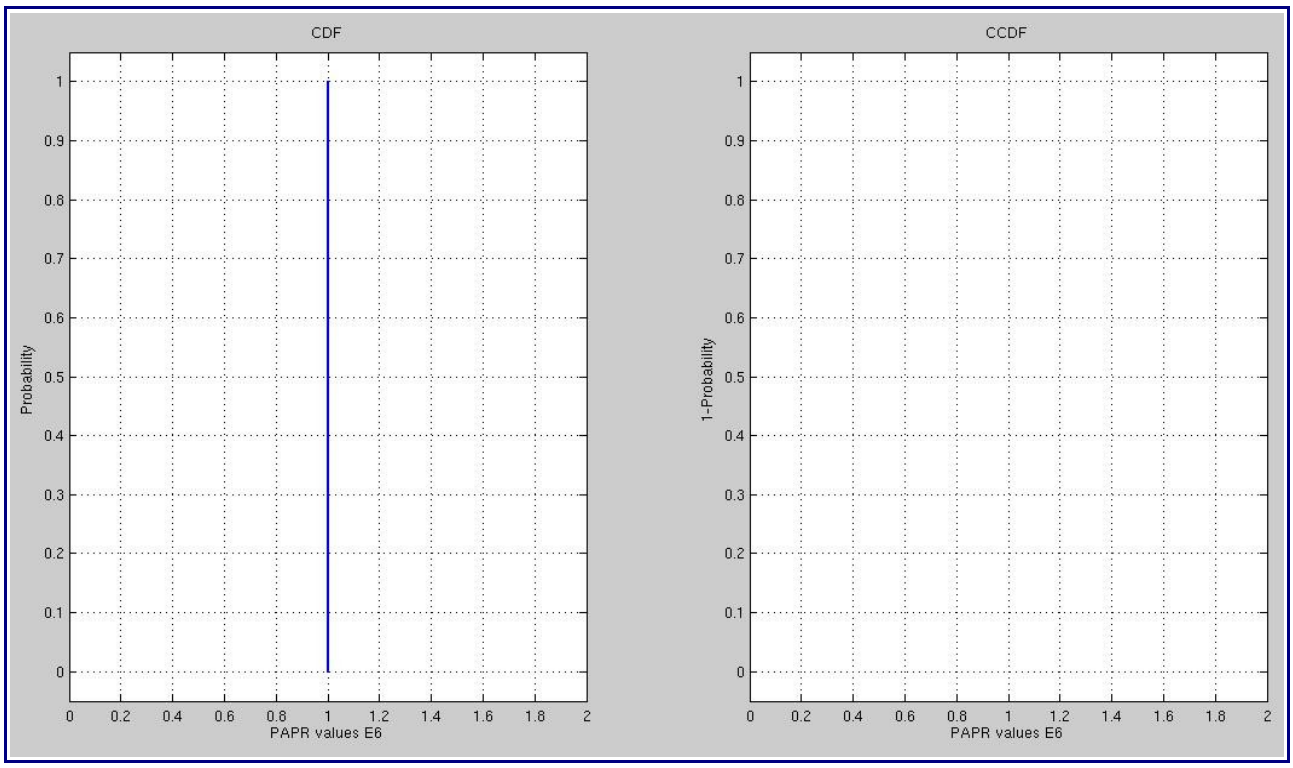


Figure 57: CDF and CCDF for E6 signal PAPR

4.3 E1 Signal

The Galileo E1-signal comprises the signal components E1-B and E1-C and is transmitted in the frequency band 1559 - 1610 MHz allocated to RNSS and ARNS on a worldwide co-primary basis (ITU-R Radio Regulations). The signal components E1-B and E1-C are data-component and pilot-component respectively. The E1-signal provides the I/NAV message and supports Safety of Life service, Galileo system integrity and Open Service [1].

The E1 signal is modulated by a CBOC(6,1,1/11) (Composite BOC) at *fo* and a symbol rate of 250 Sps and multiplexed with scheme shown in figure 59. This signal modulates with two different sub-carriers: $sc\_a = fo$ and $sc\_b = 6 \cdot fo$. Furthermore, it can be seen in *figure 59* that both pilot and data components are modulated onto the same sub-carrier component, with a power sharing of 50%.

CBOC linearly combines BOC(1,1) and BOC(6,1) sub-carriers (both components being present at all times). These sub-carriers, as the balanced sum of two squared-wave sub-carriers, will have four different levels [6].

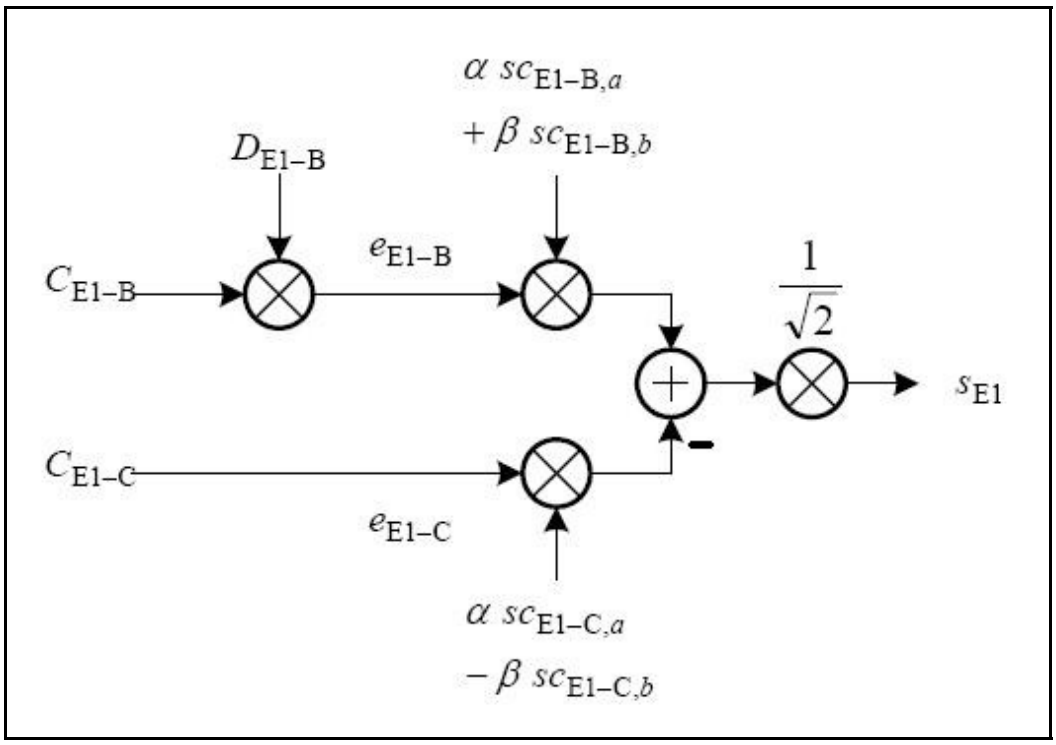


Figure 58: Multiplexing scheme for E1 CBOC signal

The power of BOC(6,1) component has to represent 1/11 of the total OS signal power. Among other parameters, then, the actual implementation of the MBOC (Multiplexed BOC) will depend upon the power share between the pilot and data channels, as well as the percentage of BOC(6,1) sub-carrier used on each of these channels.

$$S_{E1}(t)=\frac{1}{\sqrt{(2)}}(e_{E1-B}(t)(\alpha\, sc_{E1-B,a}(t)+\beta\, sc_{E1-B,b}(t))-e_{E1-C}(t)(\alpha\, sc_{E1-C,a}(t)-\beta\, sc_{E1-C,b}(t)))$$

$$with \qquad sc_x(t)=sign(\sin(2\pi R_{s,X}t))$$

$$\alpha=\sqrt{\frac{10}{11}} \qquad \beta=\sqrt{\frac{1}{11}}$$

Figure 59: Resulting E1 signal

The parameters α and β are chosen so that the combined power of the $sc_{E1\text{-}B,b}$ and the $sc_{E1\text{-}C,b}$ sub-carrier components equals 1/11 of the total power of $e_{E1\text{-}B}$ plus $e_{E1\text{-}C}$. Note that the sign of the BOC(6,1) sub-carrier is different between the data and pilot channels. This is necessary to satisfy the MBOC constraint (removal of cross-terms appearing from the cross-correlation between the BOC(1,1) and BOC(6,1) sub-carriers) [6].

$$e_{E1-B}(t) = \sum_{i=-\infty}^{+\infty} \left[c_{E1-B,|i|_{LL1-B}} d_{E1-B,[i]_{DCE1-B}} rect_{T_{C,E1-B}}(t - iT_{C,E1-B}) \right]$$

$$e_{E1-C}(t) = \sum_{i=-\infty}^{+\infty} \left[c_{E1-C,|i|_{LL1-C}} rect_{T_{C,E1-C}}(t - iT_{C,E1-C}) \right]$$

Figure 60: E1 signal components

4.4 L1 Signal

The Galileo L1-signal consists of the signal components (or channels) L1A, L1B and L1C and is transmitted in the frequency band 1559 – 1610 MHz allocated to RNSS and ARNS on a worldwide coprimary basis (ITUR Radio Regulations) [2].

L1F is an open access signal transmitted in the L1 band comprising a data channel and a pilot channel (the L1B and L1C signal components respectively). It has unencrypted ranging codes and navigation data, which is accessible to all users. The L1F navigation data stream corresponds to a I/ NAV message type and contains integrity messages as well as encrypted commercial data.

The L1P signal is a restricted access signal transmitted in L1A signal channel. Its ranging codes and navigation data are encrypted using a governmental encryption algorithm. The L1P navigation data stream corresponds to a G/NAV (signal mapped to the Public Regulated Service) message type.

L1A channel is modulated with BOCcos(15,2.5) and multiplexes together with E1 signal above using the CASM multiplexer shown in *figure 61*. Again, this signal channel is restricted and suitable parameters were used so that the simulation could be done.

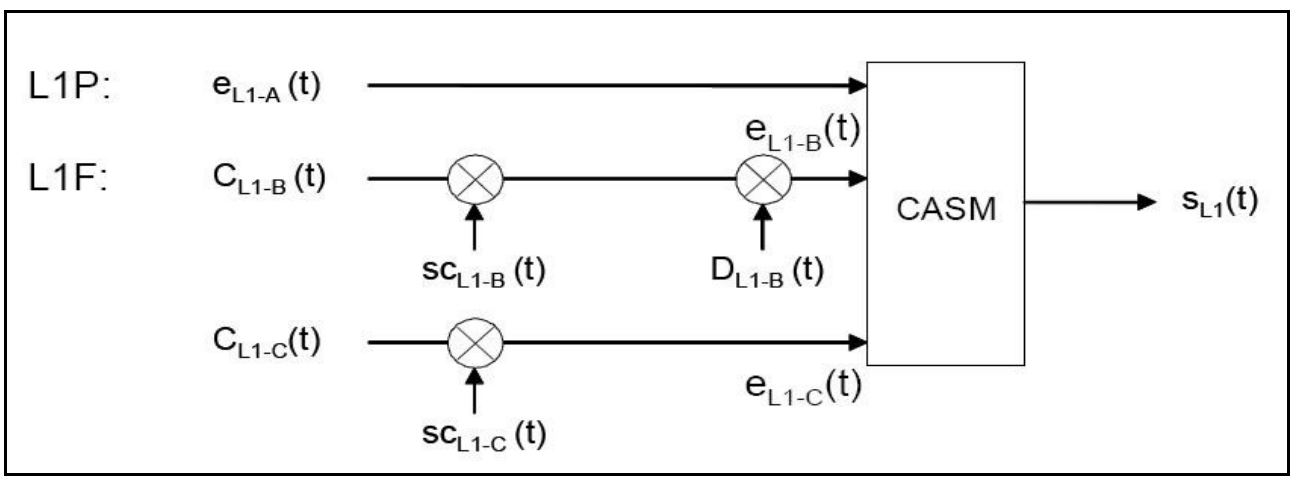


Figure 61: Multiplexing scheme for L1

And the CASM expression resulting from the multiplexing above is:

$$s_{L1}(t)=\frac{1}{3}\{\left[\sqrt{(2)}\cdot e_{L1-B}(t)-\sqrt{(2)}\cdot e_{L1-C}(t)\right]+j\left[2\cdot e_{L1-A}(t)+e_{L1-A}(t)\cdot e_{L1-B}(t)\cdot e_{L1-C}(t)\right]\}$$

The **constant envelope** is maintained by adding to the desired channels A, B and C an additional signal, which is the product of all desired binary signals (the second term of the imaginary part of expression above).

To avoid possible confusions, E1 and L1A signals have been treated separately because there was different information in [1] and [2] and both of them were useful. E1 is within the L1 band and it would correspond to band L1F or channels B and C.

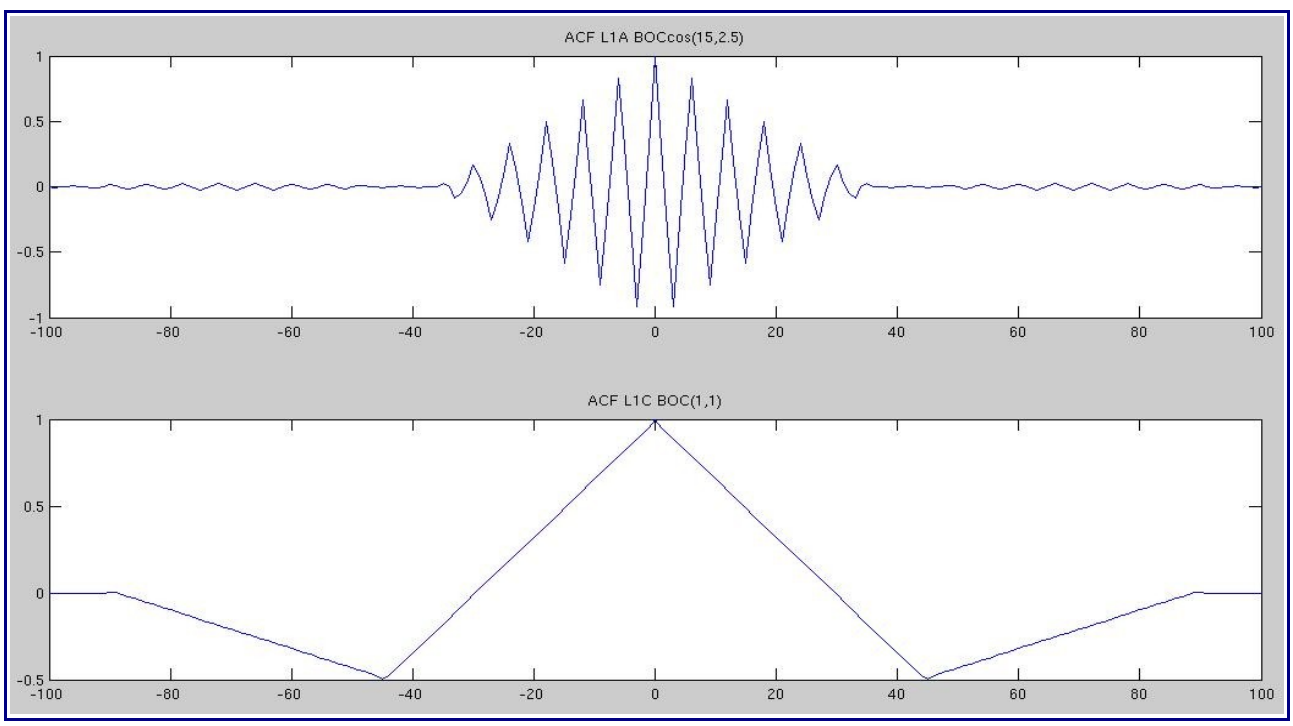


Figure 62: ACF of L1A (BOCcos(15,2.5)) and E1 (BOC(6,1,1/11)) (201 samples out of 1473119)

In *figure 62* both ACF results coincide with what it was expected (see ACF plots in 2.2.3). For the BOCcos(15,2.5), n = 12, so it should be 23+2 = 25 peaks since it is a BOC with cosine sub-carrier. For the BOC(1,1) is the simplest representation of an ACF function and it is correct: n = 2 and number of peaks equal to 3.

Regarding PSD, it can be seen in the next figure, as in E6 signal, both spectra from different channels and how they are located in the correct frequency. BOCcos is in the sub-carrier frequency and E1 signal (in green) is just in *fo*.

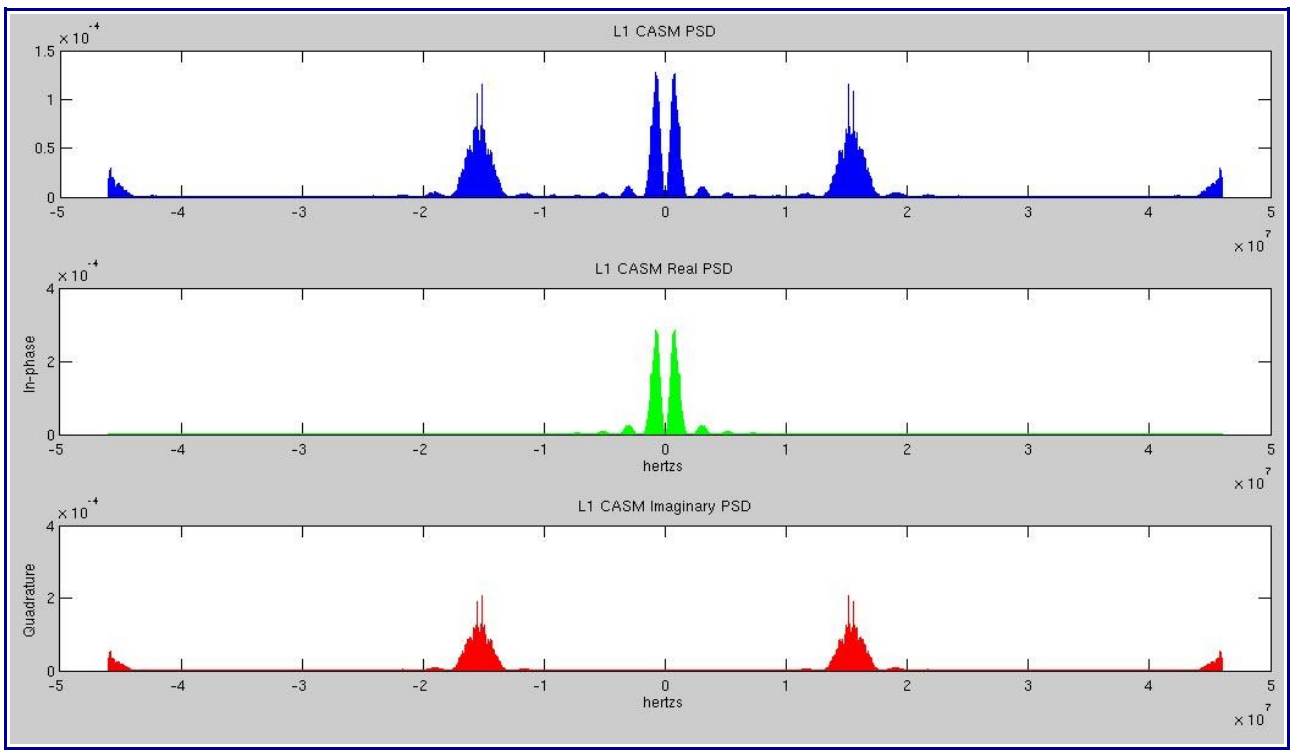


Figure 63: PSD of L1 Signal

With regard to constant envelope issue, this signal shows the same behaviour as the ones above. Its multiplexing technique is the same than E6, a modified Hexaphase modulation. The plots of the constellation (*figure 64)*, CDF and CCDF (*figure 65 and 66)* will be presented and the same behaviour will be observed for having constant modulus.

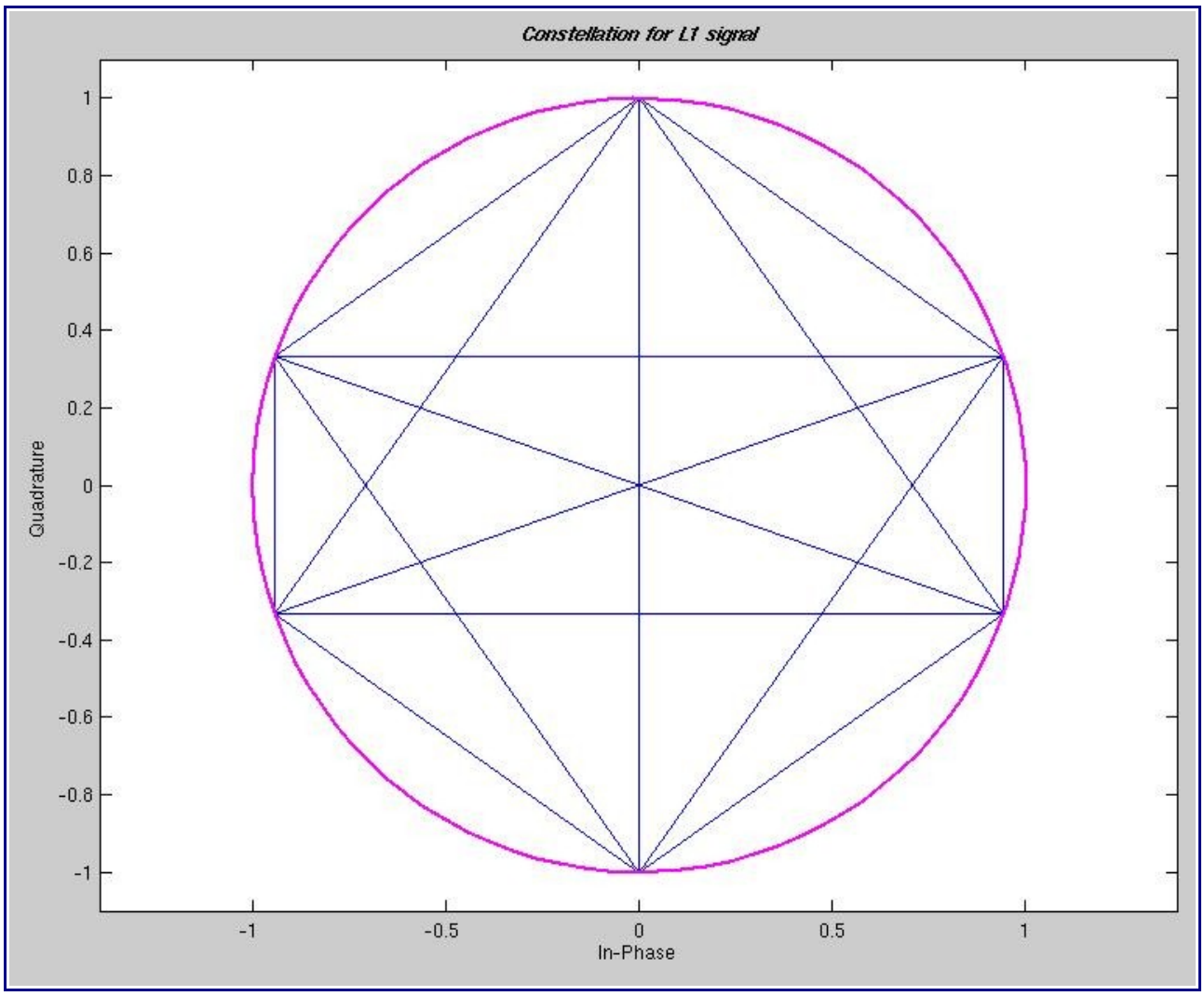


Figure 64: Constellation for L1 signal

As shown in [3], the constellation is formed by 6 symbols and this is achieved by adding an intermodulation product to assure constant envelope: the multiplication of the three channels that are added to the channel in quadrature in L1 signal expression.

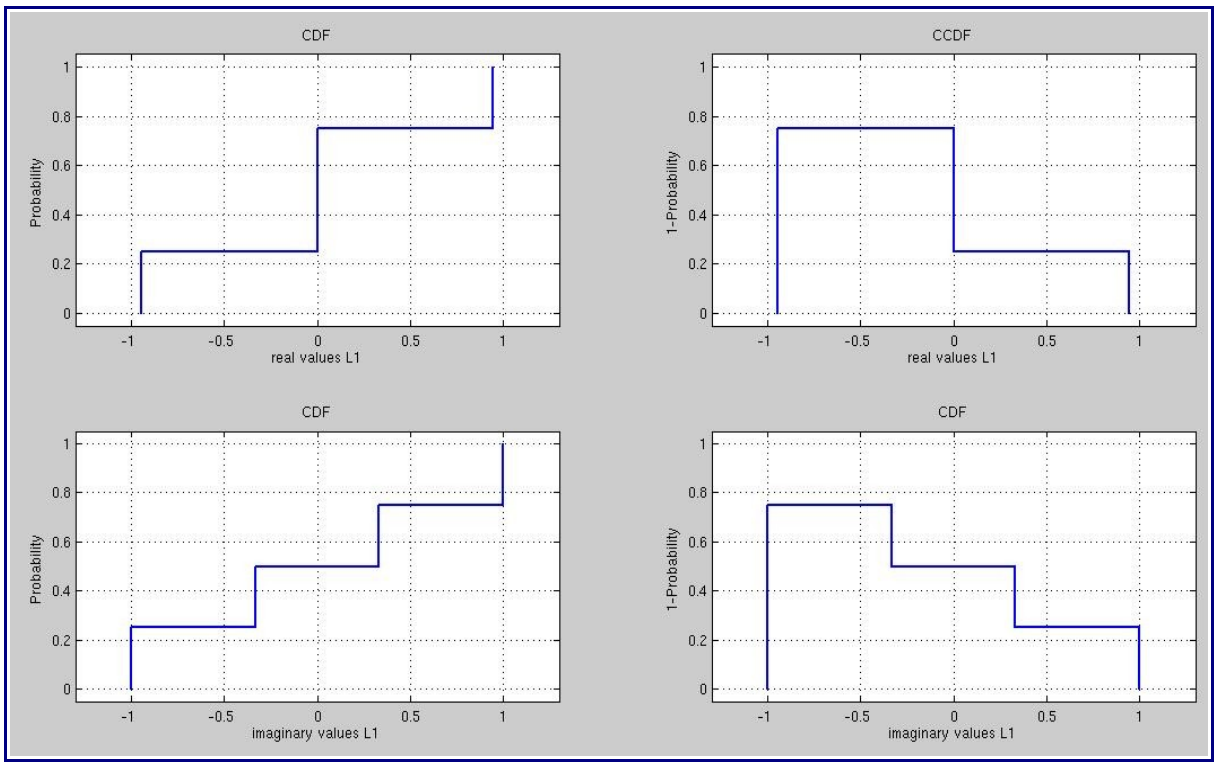


Figure 65: CDF and CCDF for L1 signal values separated in real and imaginary parts

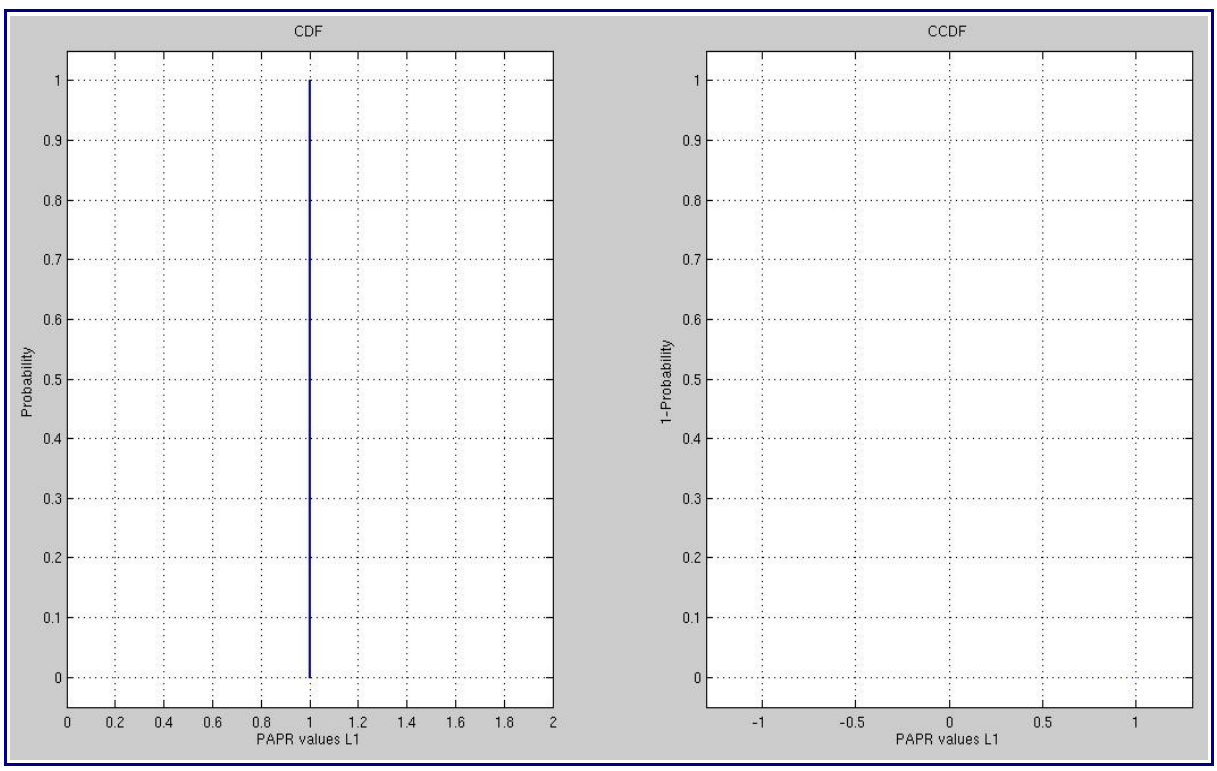


Figure 66: CDF and CCDF for L1signal PAPR

The importance of being a constant envelope signal is, mostly, due to the amplifiers in reception. For instance, if the input power is constant, its consume is also constant, thus obtaining a consumption prediction; The analog-to-digital converters (ADC) perform better if the input signal approaches the dynamic range and it is easier to design an automatic gain control (AGC) if the input power is constant; As amplifiers have a linear zone of work, non-constant envelope signals might have undesired peaks and make amplifiers go into the non-linear zone thus producing distortions, moreover, these peaks might saturate the amplifiers; Real amplifiers might have a memory effect, and, as its amplification depends on the input power, abrupt variations in the envelope might change the amplification constant. So, if we expect a constant signal we can use a cheaper amplifier; Since the SNR depends on the received power and the noise power, it is much easier to calculate the SNR with constant envelopes; These have only been some of the reasons why is so important that a signal becomes constant envelope like.

Finally, and to end point 4, it will be shown the plot of all Galileo band. In this picture the signals do not represent the real allocation since the axe for frequency is not scaled, however, they have been placed as they are in the reality, respecting distance with each other. All signals can be observed in their totality in the respective points of the work.

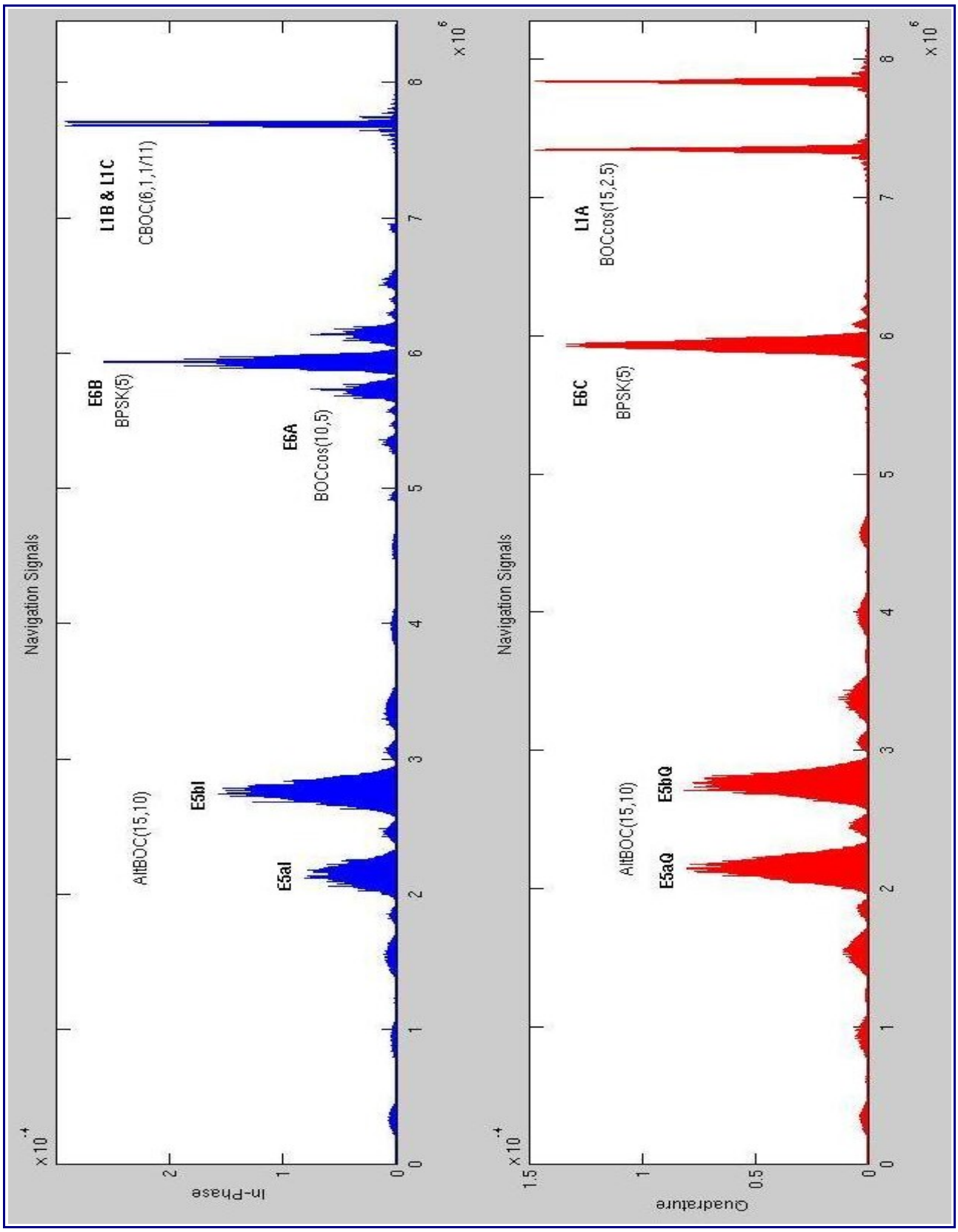


Figure 67: Galileo Band

5. Matlab aspects

The simulation of all signals has been implemented in Matlab language. At the beginning, I had to learn how Matlab behaves and works. I had only used this program once some years ago and for a short period of time, moreover, all the things I did were not related to those which I have had to deal with while doing this project about Galileo (Fourier transforms, autocorrelations, LFSR...). The work with Matlab in this project can be divided into two fields: *ranging codes* and *navigation signals*.

Although the **ranging codes** could have been done with memory codes, I opted for implementing a LFSR behaviour because I found it to be more academical. So, for doing this, I took the codes (start-values, initial sequences, tap polynomials and memory codes) provided in reference [1] and copied them in a Matlab script so that the simulation resulted as faithful as possible. Once all the data were in the scripts, that information had to be transformed from characters (octal and hexadecimal) to integers so that they could be used by the LFSR code. Afterwards, two LFSR codes had to be implemented to create the memory code. While the first one only needed the tap polynomials and the number of chips, the second one needed, apart from these two arguments, the start-values and the initial sequences. Finally, the sequence of 1's and 0's is converted to +1's and -1s to get the right amplitude. Despite the different ways that exist to implement a LFSR, such as in [1], I opted to do it as I was taught in college because it was the way I am most acquainted with. The result of doing it in one or other scheme is the same, the only difference is the order of the sequence, that is to say, if we have a sequence like 0 0 1 1 1 0 1 resulting from one scheme, if we use another scheme we obtain 0 0 1 0 1 1 1. It can be seen that there are the same number of zeros and ones but they are in a different order.

With regard to **navigation signals**, the main problem was how to implement the signal. It has to be taken into account that Matlab is a numeric program and not a symbolic program. Despite you can also work with symbolic variables, at the end of the day what we are using is vectors of numbers. At the beginning I could not even plot a sine. I did not know that a time line had to be created so that one can put the function in question in function of a variable. In addition, the signal has to be sampled and it has to match the time line so that you obtain what you really want to see in the plot. After this, it came the *Fourier transform*. Matlab uses the algorithm of the Fast Fourier Transform (FFT), however, to transform a function and then plot it properly, you have also to create the new axe for the transformed domain. Moreover, if we take the Fourier transform of a function that is

defined for negative values in time, its transform will be not wrong but bad placed. To avoid this, the command *"fftshift ( )"* has to be used twice: *fftshift (fft (fftshift (**x**)))*. Doing so, we obtain a shifted version of the Fourier transform of a shifted version of the signal in time, avoiding the error of placement in frequency and also an error in phase. As we mentioned before, Matlab is a numeric program that works with vectors, thus treating all variables as vectors (or matrices). Therefore, if we type "*a = 5* ", that variable will be stored as a matrix of 1x1 element. Then, vectors can be introduced as row or column and when it comes to multiply two vectors, their dimensions have to match in order to be multiplied correctly. For instance, if we have a vector of 1x5 and it has to be multiplied by another vector, that vector will have to be a 5x1 (and it has to be transposed if necessary). The same happens with matrices, if a power of a matrix has to be done, they must be squared matrices. For this reason, if we want to multiply (or divide, put to the square, square root...) an element as if it were a number (not a vector), we have to use the character "." before the sign of the operation (e.g. a = b.*c.*d) which makes the operation element by element. Add to this, when adding or subtracting two variables (vectors), they must be the same length, and this is a very important point that has influenced remarkably in the implementation of the navigation signals. For example, letting r_a and r_b be two bit rates for channels A and B respectively, and $r_a = 5 \cdot r_b$, we cannot transmit the same amount of bits in the two channels if these signals have to be added in any point of the script. That is, depending on how these channels are multiplexed or/and modulated, if they have to be transmitted in the same term, then they have to be the same length. Therefore, and in this case, there will be five bits for channel A for each bit of channel B.

The procedure to create the signals is as follows:

- ranging codes
- data stream
- modulating and multiplexing
- plotting

In the first point the CDMA encoding is obtained for each channel. Depending on the channel, it is a memory code or a register code. Once the sequence is stored, it is oversampled so that it matches the sampling frequency and the chip rate and, lastly, the ranging code is repeated as many times as needed to fill the amount of bits that have to be split by the CDMA code. Note in the code provided

in the appendix that as all signals have to be the same length, pilot channels have been adapted to data channels and data channels have also been adapted so that they have the same duration (amount of bits that each channel transmit).

In the second point the data is created. Depending on the signal, the commands *sign ( )* and *randn ( )* have been used to obtain a random vector of ±1. In other cases, as it is only a vector of ±1, and to obtain a better plot of the PSD, the data have been selected beforehand. After that, the data are also oversampled.

In the third point the signal is composed. In this point the modulation of the signal indicates how the channels have to be joined with the ranging codes and the sub-carriers. The most complex multiplexing is the one regarding E5 signal, because it has the most complex expression and because the sampling frequency is the highest and thus it is the signal that uses more memory. Once the signal is achieved, the autocorrelation is done. To do this point I have used the Matlab command *xcorr ( )* with the option *coeff* since it is better for ACF (it gives normalized values). Moreover, I added a second variable (*lag* in most scripts) that collects the lag (offset) of the ACF so that when it is plot, the function has the zero in the middle. After this, the Fourier transform is done with the function *shcenteredFFT ( )* where we have to pass the function to be transformed and the sampling frequency so that the frequency axe can be done. This function is a modified function that I found on a MathWorks forum to understand better how the Fourier transform is used in Matlab.

In the fourth and last point the plot of the signal is displayed. Although it seems the easiest point to do, until one does not learn how it works, it takes a long time to print exactly what you want to see and which are the options to write in the command *plot ( )*. Apart from this, there are a lot of different aspects within the plotting that have to be taken into account and they are not as simple as they seem, such as the subplot, the xlabel, the ylabel, the axis, the xlim, the ylim, the title, adding legends, adding comments, drawing in the same plot more than one signal, what colours to use, what kind of dots or markers are used to plot the waveform, the thickness of the line in which the waveform is plotted, etc.

On top of this, there is the memory issue. Matlab stores all the variables generated when a script is executed and, of course, the longer the vectors the larger is the space used in memory. So, the problem with the memory comes, above all, if the sampling frequency used is very high. For instance, if we take the signal E5 sampling frequency (around 240 MHz), it means that there are 240 million of points per second of a signal. So, the program runs quickly out of RAM memory if

that issue is not taken into account. The same happens if there are a lot of plots with large signals. There are some techniques to avoid this happens, however, the most important thing is to keep the sample frequency as low as possible (in this case). For example, it is better to use the same variable all script long although it suffers some changes, just overlapping its content until the final signal is achieved, deleting the unused variables and also preallocating space better than making growing vectors in a loop. Another way is to use nested functions, which they do not copy the variables passed as arguments and no memory is used. Another hardware solution would be to increase the virtual memory or swap memory (in Linux). The issue is that sometimes the size of a variable, in bytes, is too large to be stored in memory, not because the available space, but because it surpasses the total amount permitted for a single variable. In other words, this means that there is memory available, but there is no contiguous piece of memory that is large enough to hold the specified variable. It is also recommended to turn off the unused applications so that they do not waste RAM memory. Moreover, the command *xcorr ( )* returns a signal that has 2·L-1 samples, where L is the number of samples of the function to be correlated, thus doubling the size of the variable. So, its Fourier transform will also have 2·L-1 samples because the *fft ( )* command returns the transformed signal with the same number of points. Hence, when it comes to work with spectral densities and autocorrelations, the amount of memory is reduced dramatically.

Finally, as an a complementary material for navigation signals, there are also several plots having to do with modulations' (BPSK, BOC and AltBOC) ACF and PSD. The aim of those plots is to give a closer insight of how these modulations are like, both in time and frequency.

The first example (not provided in the appendix since it has nothing to do with the project) that I did to get familiarised with modulations and Fourier transforms and so on was a spread spectrum BPSK signal. Simply, it consisted in modulating data as binary phased and then including the CDMA code and check how the spectrum changed. It also implemented the Fourier transform and the corresponding plots of the ACF and PSD to see the results. This example was very useful to me as a sparring for the real thing I had to do.

When it comes to BOC or AltBOC the procedures that I used to simulate them were basically getting the sub-carrier function in time and find its spectrum only using the function *shcenteredFFT ( )*, to avoid problems with memory, and then plotting the absolute value to the square. Doing so, we obtained what we really wanted but without being so theoretical, since the PSD of a signal is defined as the Fourier transform of its autocorrelation function. In any case, the results differed

basically in the amplitude of the signal. If both signals were compared (PSD and absolute value to the square) it could be appreciated how both had the same harmonics in the same frequencies but the one with the absolute value had higher amplitude than the other one.

6. Future Lines

The study done could be extended by the following points:

- Since this work has been done from the point of view of the satellite (transmitter), how would it be to do the same but in the ground segment control (receiver)? How would be the signals in the receiver? As seen in [1], [4], [9], [6], [11] and [20], how multipath and attenuations due to atmosphere and other sources (a more complicated signal propagation model must be added which accounts for signal amplification at the satellite, signal losses in free space, atmosphere, antennas, and the front-end, even Doppler effects) would distort the signal coming from the space? In what way the modulations selected are good or bad for those inconveniences? As seen in [4], [6], [9] and [20], how these signals are tracked by receivers? What kind of receiver is better? What reception schemes exist for each band? How is the performance by the power amplifiers taking into account that all signals are constant envelope? Regarding the received signal, how all those errors affect the satellite performance (in accuracy for example)? Study of possible errors in the phase of the signals, etc.

- As seen in [4] and [17], from another point of view more realistic, how would be the signals simulated taking into account a limited bandwidth? How would be the effects of loss of power and shaping of the spectrum when bandlimiting? How different the ACF and PSD would be from the ideal ones?

- As seen in [11] and [5], it could be studied how the compatibility works with other satellites, GPS and GLONASS. Which are the constraints and trades-off that different parts have agreed so that all three satellite systems are interoperable in space? How the spectra in all those frequencies are assigned? Interferences between satellite systems? Methods to mitigate those interferences (filters, modulations, etc.).

- Which are the **operational** satellite systems (BEIDOU -China-, DORIS -France-, GLONASS -Russia-, GPS -USA-)? Which are the current satellite systems **in development** (COMPASS -China-, GALILEO -Europe-, IRNSS -India-, QZZS -Japan-)? Which are the Global Navigation Satellite Systems **to be updated** or renewed (EGNOS, GAGAN, GPS·C,

LAAS, MSAS, WAAS, Star Fire)? What will be improved in the current GNSS. Compare all/some of them against GALILEO.

- A deep comparison between services provided by GPS and GALILEO (maybe with GLONASS as well) and which are their characteristics. In what do they resemble and in what do they differ? Which are the features (performance) of each, e.g. accuracy, spectral efficiency, etc? Which gives a better performance? Why?

- Last but not least, and introduction of a new modulation **Double Binary Offset Carrier (DBOC)** and check which would be the changes with respect to simple BOC. Since there is not a unified analysis of the BOC-like signals and all the formulas are provided for particular cases of ACF and PSD BOC modulations, this "new" modulation gives the opportunity to unify not only BPSK, BOCsin and BOCcos, but also a wider and more flexible range of split-spectrum modulations in a single framework. For more information see [19].

Key of Terms

- **ACF**: Autocorrelation Function
- **AltBOC**: Alternate Binary Offset Carrier
- **ARNS**: Aeronautical Radio Navigation Service
- **BOC**: Binary Offset Carrier
- **BOCcos**: Binary Offset Carrier with cosine sub-carrier
- **BOCsin**: Binary Offset Carrier with sine sub-carrier
- **BPSK**: Binary Phase Shift Keying
- **CASM**: Coherent Adaptative Sub-carrier Modulation
- **CBOC**: Composite Binary Offset Carrier
- **CCDF**: Complementary Cumulative Distribution Function
- **CDF**: Cumulative Distribution Function
- **CDMA**: Code Division Mutlple Access
- **C/NAV**: Commercial navigation message
- **DBOC**: Double Binary Offset Carrier
- **EGNOS**: European Geostationary Navigation Overlay Service
- **F/NAV**: Freely accessible navigation message
- **GLONASS**: GLObal'naya Navigatsionnaya Sputnikovaya Sistema
- **GPS**: Global Positioning System
- **I/NAV**: Integrity navigation message
- **JTIDS**: Joint Tactical Information Distribution System
- **LFSR**: Linear Feedback Shift Register
- **LSB**: Least Significant Bit
- **MBOC**: Multiplexed Binary Offset Carrier
- **MIDS**: Multifunctional Information Distribution System
- **MSB**: Most Significant Bit
- **NRZ**: No-Return to Zero
- **PAPR**: Peak-to-Average Power Ratio
- **PSK-R**: Phase Shift Keying-Rectangular
- **PSD**: Power Spectral Density
- **RNSS**: Radio Navigation Satellite System

Acknowledgements

I would like to thank Dr. Máirtín O'Droma for allocating this project to me as it has given me the chance to learn a lot regarding the world of Matlab and Communications apart from having introduced me into the Galileo Satellite System.

A special sense of gratitude for my parents and brother, Carme, Rafel and David, who have made this Erasmus experience possible by assisting me in every way they could.

And finally, I give thanks to all my friends in Barcelona and all the friends and housemates I have met in Limerick and that, in their way, have also contributed to this project by supporting me whenever I needed it.

Appendix A

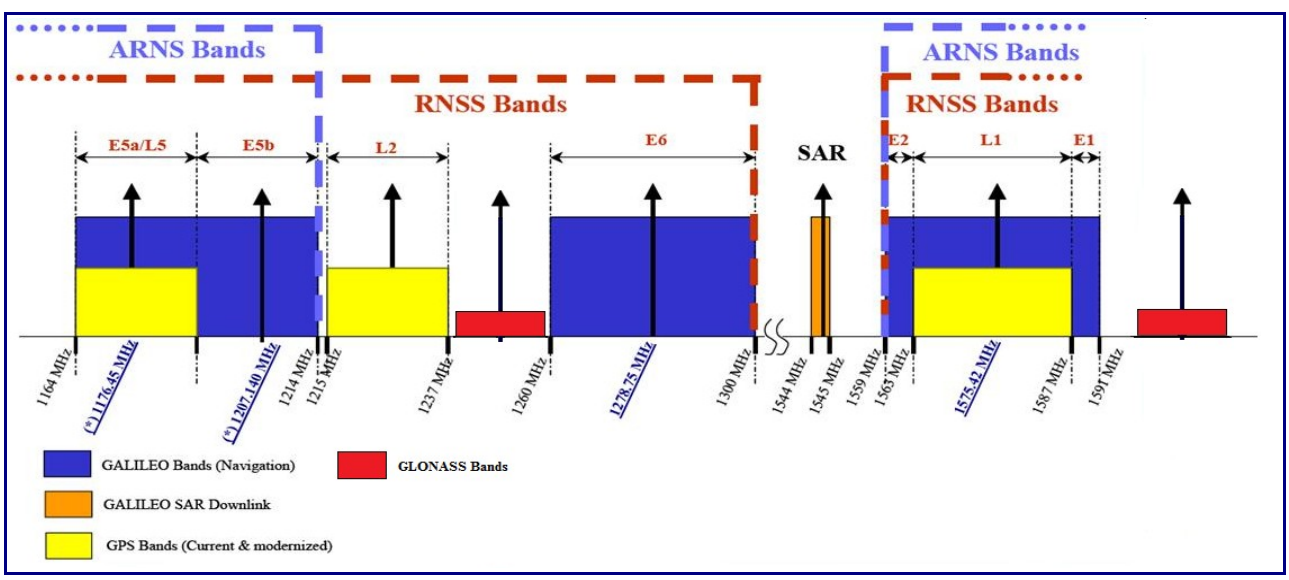
GALILEO: Introducing innovative modulation techniques

Abstract

This paper will present the different modulations and multiplexing techniques used in the Galileo Satellite System, namely ***Binary Offset Carrier (BOC)****,* ***Alternate BOC (AltBOC)*** *and* ***Coherent Adaptative Sub-Carrier Modulation (CASM)****. While the first one can be considered the origin of all BOC-like signals, the second one can be regarded as both modulation and multiplexing technique, and the last one as a multiplexing technique to achieve a constant envelope signal that will be used in BOC-like signals. Finally, it will be introduced an unifying type modulation to gather all BOC-like signals into one formula:* ***Double BOC (DBOC).***

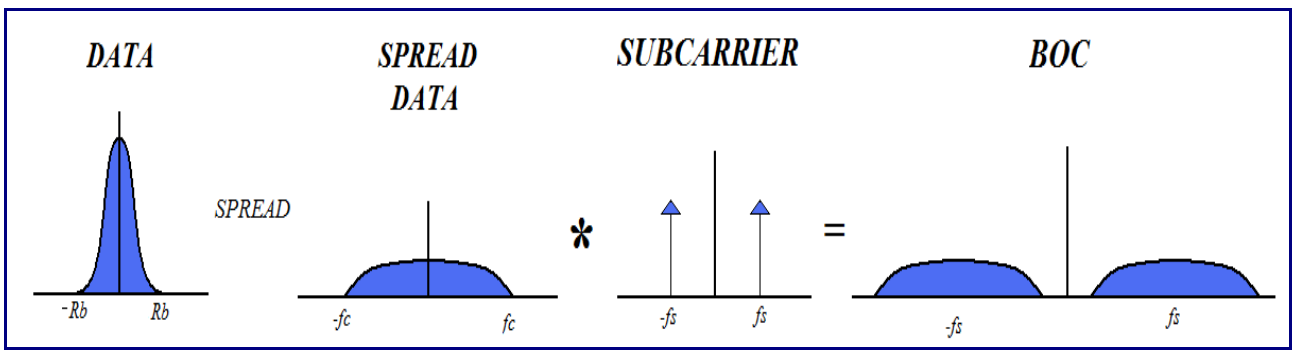
Introduction

Galileo is the European global navigation satellite system (GNSS) providing a highly accurate, guaranteed and global positioning service under civilian control. It is inter-operable with Global Positioning System (GPS) and GLONASS, the two other current global satellite navigation systems (see figure in this page*)*. American GPS has run successfully for many years but its positioning precision of civil signal is not high enough for user demands. Compared to GPS, Galileo can provide many services such as Open Service (OS), Public Regulated Service (PRS), Commercial Service (CS), Safety of Life (SOL) service and Safe and Rescue (SAR) service. European Galileo will come true that users will be satisfied with positioning precision of Galileo civil signal (signals will offer a guaranteed accuracy down to 1 metre, with value-added services achieving a real-time accuracy of 10 cm). For that, a new modulation technology will be taken by Galileo, **Binary Offset Carrier (BOC)**, which, in the scope of emerging radionavigation satellite systems, is of special interest [9].



Binary Offset Carrier (BOC)

BOC describes a class of spread-spectrum modulations recently introduced for the next generation of global navigation satellite systems. The design strategies of these BOC signals have so far focused on the spectral properties of these signals. The new generation GNSS, modernized GPS, and the European Galileo system will use BOC (*or BOC-based*) signals on different carriers and with different parameters. The main reasons for creating BOC signals were, on one hand, the need to improve traditional GNSS signals properties for better resistance to multipath interferences of all kinds and receiver noise, and on the other hand, the need for improved spectral sharing of the allocated bandwidth with existing signals (single carrier GPS) or future signals of the same class [15].



Since Binary Phase Shift Keying (BPSK) modulation is often used in the spread spectrum satellite communication systems, the splitting spectrum character of the BOC-modulated signals can achieve the spectrum isolation with the BPSK signals in the same band. So, BOC modulation will be widely applied gradually as the crowding of the communication band [16].

BOC modulations offer two independent design parameters: sub-carrier frequency *fs* (in MHz) and spreading code rate *fc* (in Mcps). These two parameters provide freedom to concentrate signal power within specific parts of the allocated band to reduce interference with the reception of other signals [7]. The result of the sub-carrier modulation is to split the classical BPSK spectrum in two symmetrical components with no remaining power on carrier frequency [17]. The product is a symmetric split spectrum with two main lobes shifted from the carrier frequency by the amount equal to the sub-carrier frequency (see figure in this page).

When it comes to the reception of the signal, the redundancy in the upper and lower sidebands of BOC modulations offers practical advantages in receiver processing for signal acquisition, code tracking, carrier tracking, and data demodulation [7]. In most ways, receiver processing of BOC modulations is similar to receiver processing of PSK-R (Phase Shift Keying with Rectangular symbol) modulations. However, BOC modulations offer some unique

opportunities for variations in receiver processing that can provide advantages in implementation and performance. The best receiver performance is obtained by processing both sidebands coherently as a single signal, not only because all the signal energy is combined coherently, but also because the unique spectral shape of the dual-sideband modulation leads to better ranging performance. However, since each of the two spectral sidebands redundantly contains all the information needed for ranging and data demodulation, the distinct sidebands can be processed separately if desired. The receiver treats the BOC modulation like a PSK-R modulation centred at one of the two subcarrier frequencies. If desired, filtering can be used to select only the desired sideband and prevent aliasing at lower sampling rates. Thus, processing one sideband is identical to processing a conventional GPS signal with a PSK-R modulation. Another motivation for sideband processing of BOC modulations is interference avoidance. Simple circuitry in a receiver can detect when partial-band interference is encountered that obscures only one sideband of the BOC modulation, and can reconfigure the receiver to process only the sideband not being interfered with. This approach can provide substantial immunity to some types of interference without the complexity of interference mitigation circuitry [4].

The BOC signal consists in a three or two (if dataless) sets of sequences, namely pesudo-random-noise (PRN), square sub-carrier and data (if any). Indeed, BOC modulation uses also a spread spectrum to achieve the direct spectrum spreading on the transmitter, then on the receiver a local PRN, which is synchronised with the one in the transmitter, is used to despread the received signal.

Apart from giving a single code to each signal, the chip rate also sets the width of the sidebands of this modulation, thus giving a wider lobe the bigger the chip rate it is. The sub-carrier is a square waveform which comes from applying the sign function to a sine or cosine function:

$$sc(t) = sign(\sin(2\pi\, fs\, t)) \;\downarrow\; F \;\downarrow\; SC(f) = -j \sum_{k=-\infty}^{+\infty} (-1)^k \, sinc\left(\frac{2k+1}{2}\right) \delta\left(f - \frac{2k+1}{T}\right)$$

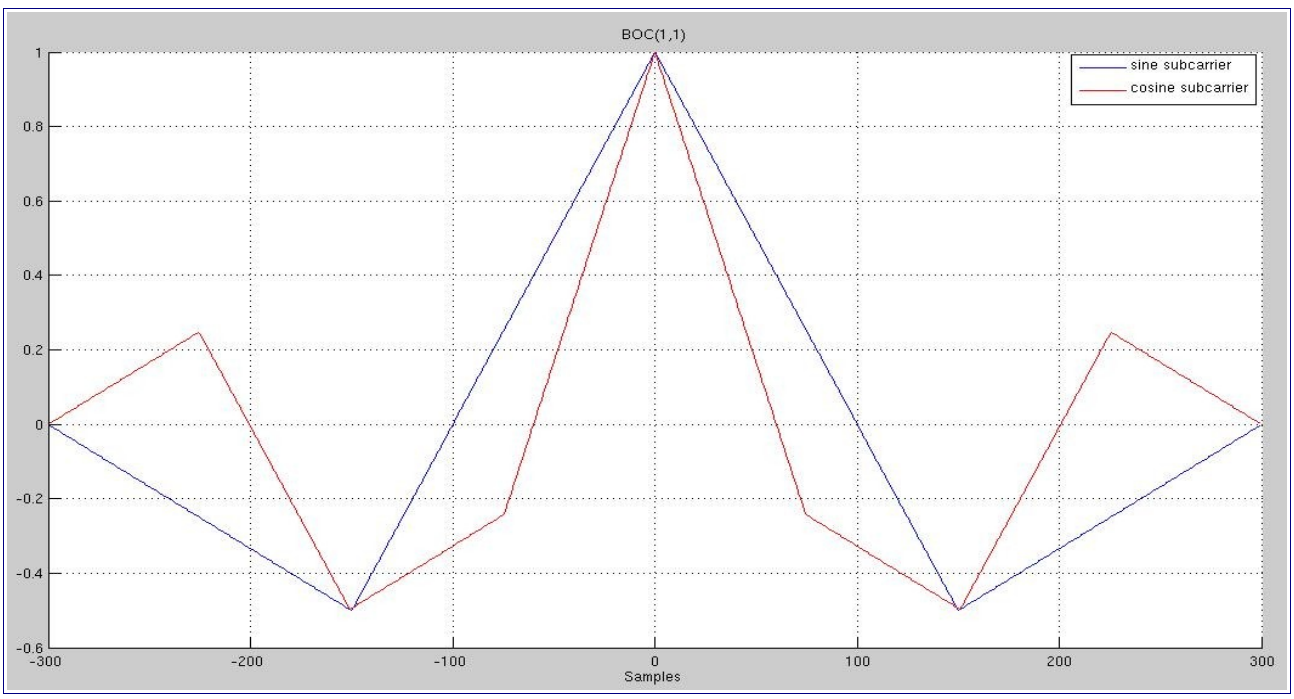
$$sc(t) = sign(\cos(2\pi\, fs\, t)) \;\downarrow\; F \;\downarrow\; SC(f) = \sum_{k=-\infty}^{+\infty} sinc\left(\frac{2k+1}{2}\right) \delta\left(f - \frac{2k+1}{T}\right)$$

The aim of the square sub-carrier is to allocate the sidebands on the correct position off the centre frequency. If we get the subcarrier and have a look to its spectrum, we will see that it is composed by a train of deltas balanced by a sinc function (s*ee equations above*). However, the most important harmonics are the ones situated at ± *fs* (as shown in the last page figure). Which sub-carrier is better? It depends on the crowding of the band, but generally sine sub-carrier spread more power on main lobes while cosine

sub-carrier concentrates more power on outer sides of spectrum, and therefore, the interference with GPS signals is reduced.

Having a look at the time domain, the ACF of BOC signals has a profile with more peaks that may be tracked. For BOC signals it is important to make sure the channel is tracking the main peak of the correlation pattern. So additional correlators are needed for measuring the correlation profile at half a sub-carrier phase from prompt correlator at either side. If one of the output values of these so-called very early and very late correlators is higher than the punctual correlation, the channel is tracking a side peak and corrective action is taken [17].

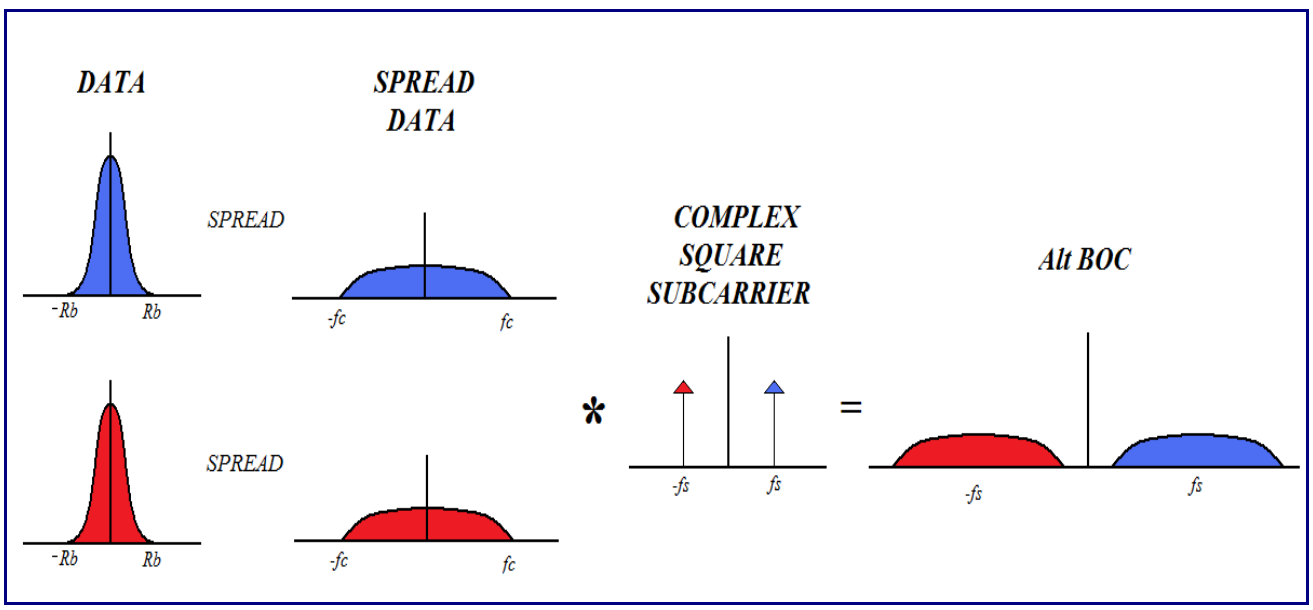
The number of peaks depend on the parameters *fs* and *fc*. Parameter *fs* is defined as $fs = \frac{1}{2\,Ts}$ and parameter *fc* is defined as $fc = \frac{1}{nTs}$, where $2Ts$ is the period of the sub-carrier and *n* is the number of half-periods of the sub-carrier existing within a chip. So, if we get a BOC(1,1), it is obtained $Ts = 0.5$ and $n = 2$. This means that there will be two half-periods of the square sub-carrier within a chip time. Getting different values for *fs* and *fc* leads to other values for *n*. The higher *n* it is, the more split a chip will be. Then, the number of peaks for the ACF will be *2n – 1* for a square-sine sub-carrier and *2n + 1* for a square-cosine sub-carrier (see figure in this page).



Alternate BOC (AltBOC)

The AltBOC modulated signals are one of the most promising innovations of the Galileo system. The innovative features of the AltBOC signals allow to foresee unprecedented performance for future Galileo receivers in the presence of the typical error sources, well-known in GNSS. In detail, the AltBOC modulation scheme will be used by Galileo satellites to broadcast new navigation signals in the E5 band (1164-1215 MHz). The AltBOC modulation derives from the family of BOC modulations. This modulation scheme allows us to obtain a constant envelope modulated signal and can also be considered as a multiplexing technique. The data channels and the pilot channels are transmitted respectively, as in-phase components and quadrature phase components of the modulated signal. Accordingly, a single data channel (equivalent to a BPSK signal) and a pilot channel (another BPSK signal) will be transmitted in each sideband.

Future Galileo receivers will be able to acquire the signals transmitted from one satellite choosing to receive only one or both of the sidebands (E5a and E5b) and then taking advantage of the correlation properties of up to four codes. In fact a receiver will be able to distinguish the four channels since four different quasi-orthogonal PRN codes will be used by each Galileo satellite. It must also be remarked that the AltBOC modulated signal features interesting correlation properties, depending on the chosen type of receiver architecture. In fact, two different categories of architectures can be used for the reception of the E5 signals: the BPSK-like or the true AltBOC architectures [7].



As figure of above shows, the AltBOC modulation allows the introduction of different channels onto the same sub-carrier. That is possible because the sub-carrier type used is different. In this case, the sub-carrier function is a “complex” rectangular exponential that only

shifts the spectrum up or down (only the conjugate has to be applied to the complex exponential) of the centre frequency. Hence, the spectrum of this sub-carrier will be a delta shifted *fs* hertzs up or down depending on the sign of the imaginary part:

$$sc(t) = sign(\cos(2\pi\, fs\, t)) + j\, sign(\sin(2\pi\, fs\, t)) = e^{j\alpha}$$
$$\downarrow$$
$$F$$
$$\downarrow$$
$$\delta(f - fs)$$

However, it has to be remarked that the four Galileo channels within the E5 band are multiplexed by a modified version of the AltBOC modulation here presented. AltBOC in itself does not provide a constant envelope signal if pilot channels are also in the signal of interest. Therefore, the procedure taken to achieve the important feature of constant envelope is to add a second set of signals to the data and pilot channels which will make the signal constant modulus. These added signals consist in products of the different four channels whose interaction provokes intermodulation products, thus losing a little part of the useful power as a counterpart to achieve the constant modulus.

Coherent Adaptative Sub-Carrier Modulation (CASM)

CASM is a multichannel modulation scheme (phase-shifted-keyed / phase modulation (PSK/PM)) also known as Tricode Hexaphase Modulation or Interplex Modulation which combines multiple signals into a phase modulated composite signal and it is used to ensure that the signal transmitted from the satellite has a constant power envelope, that is to say, the total transmitted power does not vary over time. Thus the transmitted information is not contained in the signal amplitude and the transmitted signal amplitude becomes less critical [17]. In Galileo this scheme is used explicitly in bands E6 (channels A, B and C) and E1 (channels A, B and C) and implicitly in E5 band (channels E5a-I, E5a-Q, E5b-I and E5b-Q).

The formula [18] for this modulation is $s(t) = \sqrt{2P} \cdot \cos(2\pi\, fc\, t + \theta(t) + \varphi)$, where:

- P is the total average power.
- fc the carrier frequency.
- $\theta(t)$ is the phase modulation.
- φ is a random phase.

And in the case of the Galileo Satellite System $\theta(t)$ is defined as:

$$\theta(t) = \beta_1 s_1(t) + \sum_{n=2}^{N} \beta_n \cdot s_1(t) \cdot s_n(t)$$, where:

- N is the number of signals to be multiplexed.
- B_n is the modulation index.

- $s_n(t)=\pm 1$.

So, applying this formula to either E6 band or E1 band, three **binary** channels (A, B and C) are *interplexed*:

$$s(t)=\sqrt{2P}\cdot\cos(2\pi\, fs\; t+\beta_1\cdot s_1(t)+ \\ +\beta_2\cdot s_1(t)\cdot s_2(t)+\beta_3\cdot s_1(t)\cdot s_3(t)+\varphi)$$

As $s_1(t)$ is to be located in the quadrature component with the two other signals the first modulation index is set as $\beta_1=-\frac{\pi}{2}$. By applying trigonometric expansions it can be shown that:

$$s(t)=\sqrt{2P}[\cos(2\pi\, fc\; t+\varphi)\cdot\cos(Q)- \\ -\sin(2\pi\, fc\; t+\varphi)\cdot\sin(Q)]$$

where

$$Q=-\frac{\pi}{2}\cdot s_1(t)+\beta_2\cdot s_1(t)\cdot s_2(t)+\beta_3\cdot s_1(t)\cdot s_3(t)$$

And developing *cos(Q)* and *sin(Q)* having into account that $s_1(t)$, $s_2(t)$ and $s_3(t)$ are binary signals:

$$\cos(-\frac{\pi}{2}+\beta_2+\beta_3) \;=\; \sin(\beta_2+\beta_3) \;= \\ =\; \sin(\beta_2)\cos(\beta_3)+\sin(\beta_3)\cos(\beta_2)$$

$$\sin(-\frac{\pi}{2}+\beta_2+\beta_3) \;=\; -\cos(\beta_2+\beta_3) \;= \\ =-\cos(\beta_2)\cos(\beta_3)+\sin(\beta_2)\sin(\beta_3)$$

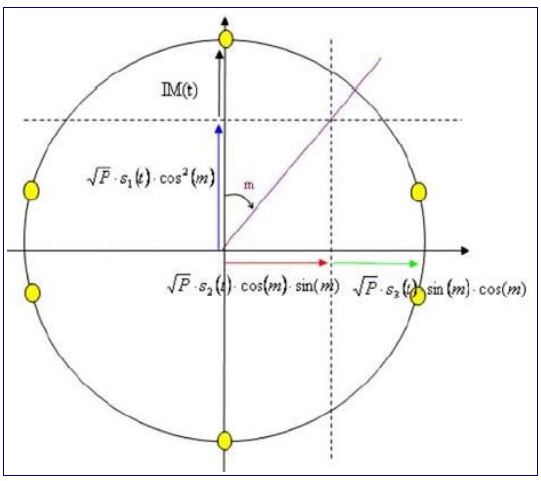
it is finally obtained:

$$s(t)=\sqrt{2P}[\cos(2\pi\, fs\; t+\varphi)\cdot[\sin(\beta_2)\cos(\beta_3)+\sin(\beta_3)\cos(\beta_2)]+ \\ +\sin(2\pi\, fs\; t+\varphi)[-\cos(\beta_2)\cos(\beta_3)+\sin(\beta_2)\sin(\beta_3)]]$$

Once this form is achieved, different channels are introduced as needed:

$$\frac{s(t)}{(2P)^{1/2}}=[\boldsymbol{s_2(t)}\sin(\beta_2)\cos(\beta_3)+\boldsymbol{s_3(t)}\sin(\beta_3)\cos(\beta_2)]\cos(2\pi\, fs\; t+\varphi)+ \\ [\boldsymbol{s_1(t)}\cos(\beta_2)\cos(\beta_3)-\boldsymbol{s_1(t)\,s_2(t)\,s_3(t)}\sin(\beta_2)\sin(\beta_3)]\sin(2\pi\, fs\; t+\varphi)$$

Having a look at this final expression, it can be noticed that the first three terms correspond to the desired useful signal terms *s1(t)*, *s2(t)* and *s3(t)*, and the fourth one is an undesired intermodulation term which consumes a small fraction of the total transmitted power available for the three desired signals.



So, having a look at the L1 signal (constellation above) signal expression, it can be seen the analogies between them and the signal s(t) studied in this paper, where s2(t) and s3(t) would be channels B and C and s1(t) would be channel A (in quadrature).

Double BOC

Since most of the provided formulas in references and papers about BOC-like signals

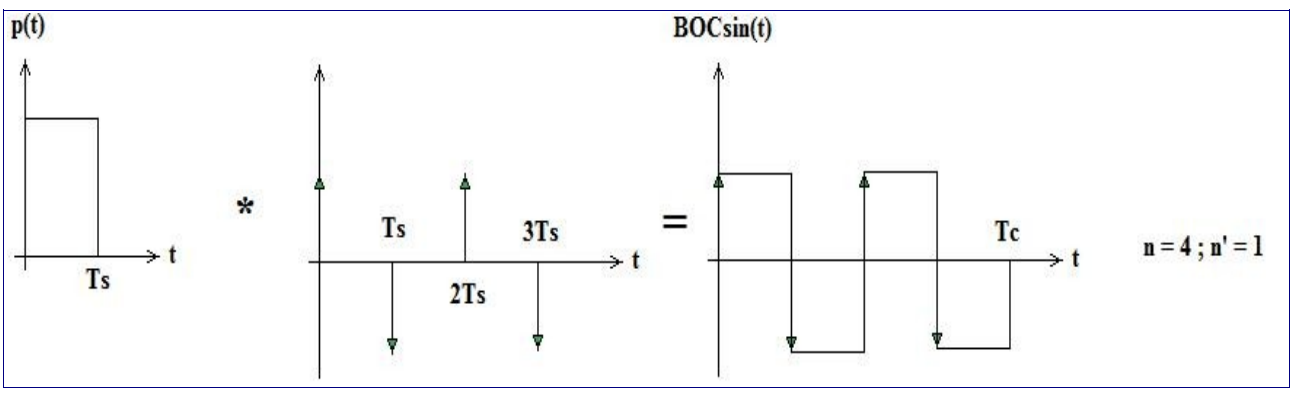
are regarded as particular cases of the different parameters (*fs*, *fc*, *n*, etc.) for both time and frequency domains, it is a good idea to unify all these examples in a formula which depending on what parameters are applied, it can be obtained any of the modulations studied in this work: BPSK, BOCsin, BOCcos, AltBOC [21] and high order DBOC.

To do so, the typical BOC formula has not to be seen as a product of data, spreading code and square sub-carrier. Conversely, it has to be looked as if it were a convolution between a period of the square sub-carrier within a chip time convoluted by the spread data symbol (already including bits (symbols) and chips modulated amongst themselves).

According to this last paragraph, we can express a BOCsin signal like this:

$$x(t)=s_{sinBOC}(t)*\sum_{m=-\infty}^{\infty}\sum_{k=1}^{SF}b_m c_{k,m}\,\delta(t-m\cdot T_b-k\cdot T_c)=$$

$$=s_{sinBOC}(t)*d(t)$$



where:

- b_m is the m^{th} bit (symbol).
- $c_{k,m}$ is the k^{th} chip of the m^{th} bit.
- $s_{sinBOC}(t)$ is the waveform of a BOCsin signal.
- T_b is the bit (symbol) duration.
- T_c is the chip duration.

Making a further insight into the $s_{sinBOC}(t)$ signal, which is only defined in a chip time (T_c), it can be expressed like:

$$s_{sinBOC}(t)=sign(\sin(2\pi f_s t)) \qquad when \;\; 0\leqslant t\leqslant T_c$$

, and then be rewritten as:

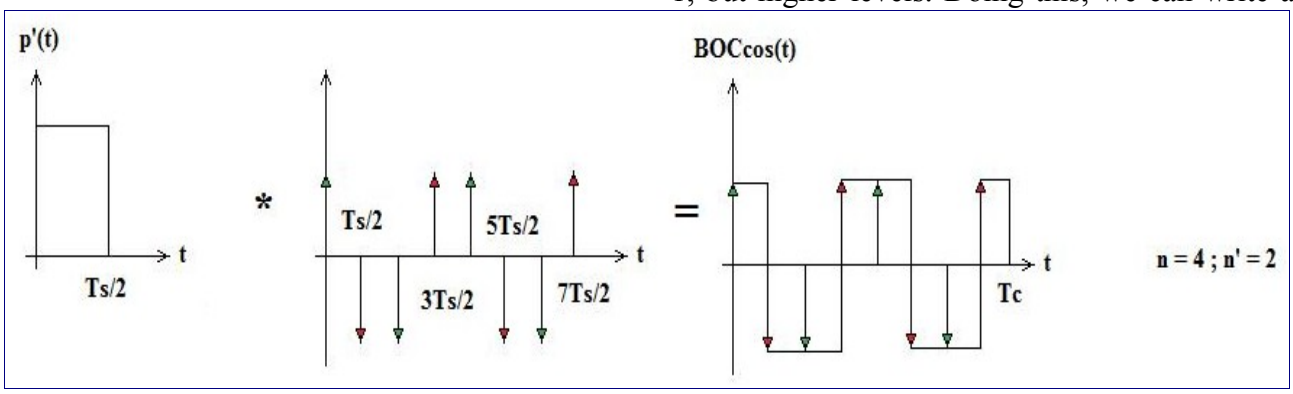
$$s_{sinBOC}(t)=p(t)*\sum_{i=0}^{n-1}(-1)^i\,\delta(t-i\cdot T_s)$$

where:

- *p(t)* is a rectangular pulse of amplitude 1 and duration T_s.
- *n* is the number of half-periods within a chip time.
- T_s is the the half of the sub-carrier period.

The same can be done for a BOCcos signal:

$$s_{cosBOC}(t) = p'(t) * \sum_{l=0}^{1} \sum_{i=0}^{n-1} (-1)^{i+l} \delta(t - iT_s - l\frac{T_s}{2})$$



where:

- *p'(t)* is a rectangular pulse of amplitude 1 and **duration T_s/2**.
- *l* accounts for the splitting of the half period.

If this last expression is developed by the summation in *l*, it will be checked out that a BOCcos is only a BOCsin modulated signal where its half periods have been split into two parts and where the second part of each split half period is multiplied by -1, thus achieving the appropriate waveform of the *sign(cos(·))*.

Taking this step further, it can be generalised for cases where the parameter *l* goes not only until 1, but higher levels. Doing this, we can write a general formula that will involve the modulations studied before:

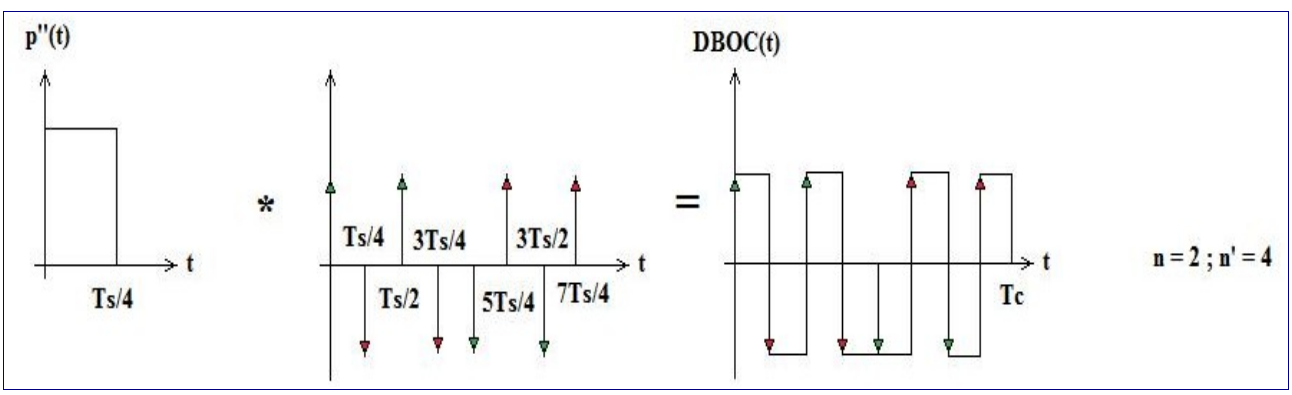
$$s_{DBOC}(t) = \hat{p}(t) * \sum_{l=0}^{n'-1} \sum_{i=0}^{n-1} (-1)^{i+l} \delta(t - iT_s - l\frac{T_s}{n'})$$

where:

- $\hat{p}(t)$ is a rectangular pulse of amplitude 1 and duration T_s /n'.
- n' can be seen as the order of the second stage (in case of BOCcos it is 2).

obtaining:

$$x(t) = s_{DBOC}(t) * d(t)$$



The advantage of this modulation is that the

higher it is n' the more separated the main lobes are located from the centre sub-carrier frequency, thus avoiding interferences with GPS in L1 band for example. However, the counterpart to be assumed is to use more bandwidth. It also provides a more comfortable way to write BOC-like signals and have them all gathered into one single formula, since giving different values to n and n' several modulations can be obtained:

- n = 1 ; n' = 1 → DBOC = BPSK
- n > 1 ; n' = 1 → DBOC = BOCsin
- n > 1 ; n' = 2 → DBOC = BOCcos
- n >1 ; n' > 2 → High order DBOC

Conclusions

This paper introduced three modulations (**BOC**, **AltBOC** and **CASM**) which are currently used in the GNSS. Although it has been shown that both AltBOC and CASM can also be interpreted as a multiplexing scheme, they have been described as single modulations. It has been shown the advantages of each of them regarding Galileo signals: the spectral isolation and receiver processing that BOC signals can provide, the way in which more than one channel can be introduced in the same sub-carrier with AltBOC, and the very necessary feature of constant envelope that CASM modulation provides by adding intermodulation products to the desired signal. In particular, for this last case, some mathematical calculation has been provided for the better understanding of the modulation. Finally, a new way of expressing BOC signals has been introduced, **DBOC**, thus giving its pros and cons and analysing the different types that can be obtained depending on its modulation parameters.

References:

[1] European Space Agency/European GNSS Supervisory Authority. "Galileo Open Service. Signal in Space Interface Document. OS SIS ICD , Draft 1", 2008.
URL (2009-04-09):
http://www.gsa.europa.eu/go/galileo/os-sis-icd/galileo-open-service-signal-in-space-interface-control-document#Register

[2] Galileo Joint Undertaking. "*L1 band part of Galileo Signal in Space ICD (SIS ICD)*", pp 12,15; 2005.
URL (2009-04-09):
http://www2.matimop.org.il/1/foreign/Galileo%20Signal.pdf

[3] Ester Armengou Miret. "*Galileo Signal in Space Design*", pp 10,11; May 2005.
URL (2009-04-09):
http://www.galileoic.org/la/files/SignalPresentation_MasterPolito_9thMay2005.pdf

[4] John W. Betz. "*Binary Offset Carrier*

Modulation for Radionavigation", *Journal of The Institute of Navigation* Vol. 48, No. 4, pp 227-229, 230-231, 235-236, 244-245; March 2002.

[5] José-Ángel Ávila Rodríguez, Guenter W. Hein, Stefan Wallner, Jean-Luc Issler, Lionel Ries, Laurent Lestarquit, Antoine De Latour, Jeremie Godet, Frederic Bastide, Tony Pratt and John Owen. "*The MBOC modulation. A Final Touch for the Galileo Frequency and Signal Plan*", *Inside GNNS*, p 55; September/October 2007.

[6] Olivier Julien, Christophe Macabiau, Jean-Luc Issler and Lionel Ries. "*Two for One. Tracking Galileo CBOC Signal with TMBOC*", *Inside GNSS*, pp 50-51, 53-54; Spring 2007.

[7] Davide Margaria, Fabio Dovis and Paolo Mulassano. "*Galileo AltBOC Signal Multiresolution Acquisition Strategy*", *IEEE A&E SYSTEMS MAGAZINE*, pp 4-5; November 2008.

[8] John G. Proakis, Masoud Salehi. "*Communication Systems Engineering*". Prentice-Hall, pp 180-183 ; 1994.

[9] Li Qiang, Li Huifeng, Liu Bing. "*A new algorithm of BOC multipath errors simulation*", *Second International Conference on Space Information Technology. Proc. of SPIE Vol. 6795, 67950M, doi 10.1117/12.773751,* pp 3-5; 2007.

[10] Grace Xingxin Gao, Jim Spilker, Todd Walter, Per Enge. Standfor University, USA. "*Code Generation Scheme and Property Analysis of Broadcast Galileo L1 and E6 Signals*", p 7.
URL (2009-04-09):
http://waas.stanford.edu/~wwu/papers/gps/PDF/GaoIONGNSS06.pdf

[11] Guenter W. Hein, Jeremie Godet, Jean-Luc Issler, Jean-Christophe Martin, Philippe Erhard, Rafael Lucas-Rodriguez and Tony Pratt. "*Status of Galileo Frequency and Signal Design*", *Proceedings of the 15th International Technical Meeting of the Satellite Division of the Institute of Navigation ION GPS 2002,* pp 5, 10-12; September 24-27, 2002.

[12] ESA Publications Division. *"The First Galileo Satellites. Galileo In-Orbit Validation Element"*, p17; November 2005.
URL (2009-04-09):
http://www.mi.gov.pl/1/files/0/3141/galileo_brochure.pdf

[13] European Space Agency. "*Galileo Full Operation Capability (FOC). Ground Mission Segment. High Level Requirements Document. Appendix 2-A*", p 6; 1 July 2008.

[14] E. Sayrol, A. Gasull, A. Moreno, J. Salavedra, F. Vallverdú. "*Senyals i sistemes*

analògics. Una introducció pràctica", UPC Editions, p 91; 2001.

[15] B.Muth, P. Oonincx and C.Tiberius. "*A time domain fingerprint for BOC(m,n) Signals", EURASIP Journal on Advances in Signal Processing. Volume 2007, Article ID 56104,* p 1; 10 April 2007.

[16] Li Yang, YongxinFeng, ChengshengPan, YumingBo. "*The research of side-band acquisition for BOC modulated signals*", p 645; 2007.
URL (2009-04-09) (from UL e-Resources):
http://ieeexplore.ieee.org.proxy.lib.ul.ie/stamp/stamp.jsp?arnumber=04339942

[17] Kai Borre. "*The Galileo signals with emphasis on L1 OS*", *12th International Power Electronics and Motion Control Conference, ERE-PEMC 2006,* pp 2027, 2028; 30-08-2006/ 01-09-2006.

[18] Emilie Rebeyrol, Olivier Julien, Christophe Macabiau, Lionel Ries, Antoine Delatour, Laurent Lestarquit. "*Galileo civil signal modulations*", *GPS Solut (2007) 11:159-171, DOI 10.1007/s10291-006-0047-3*, pp 159, 163-165; 21 December 2006.

[19] Elena Simona Lohan, Abdelmonaem Lakhzouri, Markku Renfors. "*Binary-Offset-Carrier modulation techniques with applications in satellite navigation system*", *Wireless Communications and Mobile Computing, Wiley InterScience, DOI 10.1002/wcm.407*, pp 767-770; 7th July 2006.

[20] Günter W. Hein, Markus Irsigler, José Ángel Avila Rodríguez and Thomas Pany. "Performance of Galileo L1 Signal Candidates", pp 3-4, 6-8, 13-14.
URL (2009-04-13): http://forschung.unibw-muenchen.de/papers/ktmzvhb7tqqpis3srpl7anp3bk6izl.pdf

[21] Elena Simona Lohan, Abdelmonaem Lakhzouri, Markku Renfors. "*Complex double-binary-offset-carrier modulation for a unitary characterisation of Galileo and GPS signals*" The Institute of Engineering and Technology 2006. IEE Proc.-Radar Sonar Navig., Vol 153, No 5, doi:10.1049/ip-rsn:20060005, pp 403-404; October 2006.

Appendix B

Matlab scripts

1. Navigation Signals

```
function AllSign

[Ie5,Qe5]=sE5n;
[Ie6,Qe6,ChAe6]=sE6;
[Il1,Ql1]=sL1;

% figure(1);%OK
% plot(abs(Ie5));
% title('E5aI & E5bI');
%
% figure(2);%OK
% plot(abs(Ie6));
% title('E6B');
%
% figure(3);%OK
% plot(abs(ChAe6));
% title('E6A');
%
% figure(4);%OK
% plot(abs(Il1));
% title('L1B & L1C');
%
% figure(5);%OK
% plot(abs(Qe5),'color','red');
% title('E5aQ & E5bQ');
%
% figure(6);%OK
% plot(abs(Qe6),'color','red');
% title('E6C');
%
% figure(7);%OK
% plot(abs(Ql1),'color','red');
% title('L1A');

%
% Ie5=Ie5(0.410400e6:4.910400e6);
% Qe5=Qe5(0.410400e6:4.910400e6);
%
% ChAe6=ChAe6(100000:1805999);
% Ie6=Ie6(100000:1805999);
%
Ql1=Ql1(100000:end-100000);
% Qe6=Qe6(0.1e6:1945999);
%2045999

% z1=zeros(1,87);
% z2=zeros(1,200);

% I pad with zeros Ie6 to have same length as ChAe6
```

```
L=length(ChAe6)-length(Ie6);
L=L/2;

p=zeros(1,L);
Ie6=[p Ie6 p];

I=[abs(Ie5)   (abs(Ie6)+abs(ChAe6)) abs(Il1)];
Q=[abs(Qe5)   abs(Qe6)   abs(Ql1)];

%_____________________PLOT_____________________________________________

figure(8);
subplot(2,1,1), plot(I);
xlim([0 8.45e6]);
ylabel('In-Phase');
title('Navigation Signals');
% str1(1)={'E5aI & E5bI'};
% str1(2)={'AltBOC(15,10)'};
% text(1.7e6,1e-4,str1);
% str2(1)={'E6A'};
% str2(2)={'BOCcos(10,5)'};
% text(4.3e6,1.65e-4,str2);
% str3(1)={'E6B'};
% str3(2)={'BPSK(5)'};
% text(5e6,2.55e-4,str3);
% str4(1)={'L1B & L1C'};
% str4(2)={'CBOC(6,1,1/11)'};
% text(6e6,2.5e-4,str4);

subplot(2,1,2), plot(Q,'color','red');
xlim([0 8.273e6]);
ylabel('Quadrature');
title('Navigation Signals');
% str1(1)={'E5aQ & E5bQ'};
% str1(2)={'AltBOC(15,10)'};
% text(1.7e6,1e-4,str1);
% str2(1)={'E6C'};
% str2(2)={'BPSK(5)'};
% text(4.4e6,1.3e-4,str2);
% str3(1)={'L1A'};
% str3(2)={'BOCcos(15,2.5)'};
% text(5.85e6,1.75e-4,str3);
```

```
function [RGs,IGs]=sE5

% Generates Galileo's signal E5 with AltBOC(15,10) modulation and
% parameters with one eigth (T/8) of the subcarriers' period.
%************************************************************

fsamp=240*1.023e6;  % Sampling frequency:    245.52 Mhz
fs=15*1.023e6;      % Sub-Carrier frequency: 15.345 Mhz
fc=10*1.023e6;      % Chiprate:              10.23  Mcps
SraI=50;            % Symbol rate aI:        50     Sps
SrbI=250;           % Symbol rate bI:        250    Sps
Tsamp=1/fsamp;      % Sample time            10.861 ns
TraI=1/SraI;        % aI symbol time         20     ms
TrbI=1/SrbI;        % bI symbol time          4     ms
BaI=1;              % 1 bits ---> 40ms duration
BbI=5;              % 5 bits ---> 40ms duration
```

```
D=BaI*TraI;          % Duration of simulation
                     % It'd be the same if D=BbI*TrbI
t=0:Tsamp:D-Tsamp;

%++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++++

[CaI,CaQ,CbI,CbQ]=draftcdmaE5;

CaI=expan(CaI,fsamp,fc);
CaQ=expan(CaQ,fsamp,fc);
CbI=expan(CbI,fsamp,fc);
CbQ=expan(CbQ,fsamp,fc);

% PROBLEM: Since the duration of pilot channels is longer, when it comes to
% multiply the signals in Matlab, all signals must have the same length.
% Therefore, quadrature channels' length will be truncated to in-phase length.

CaI=fulltime(CaI',BaI);
CaQ=CaQ(1:1:length(CaI));

CbI=fulltime(CbI',BbI);
CbQ=CbQ(1:1:length(CbI));

% However, the shorter signal coud be padded with zeros so that they had
% the same length. But, doing so it would entail memory problems.

% It can also be possible to set a common length for all signals so that
% all of them are 100 ms long. Then, data channels should send enough bits
% to fill the whole duration. Therefore, doing so the timeline should be
% varied also, which is the origin of all the inconveniences.

%++++++++++++++++++++data stream++++++++++++++++++++++++++++++++

DaI=-1;                  %-sign(randn(1,BaI));
DbI=[1 -1 1 1 -1];       %-sign(randn(1,BbI));

DaI=expan(DaI,fsamp,SraI);
DbI=expan(DbI,fsamp,SrbI);

%---------------------signal components---------------------------
sdaI=DaI'.*CaI;                  % e5aI
sdaQ=CaQ';                       % e5aQ
sdbI=DbI'.*CbI;                  % e5bI
sdbQ=CbQ';                       % e5bQ
asdaI=sdaQ.*sdbI.*sdbQ;          % *e5aI = e5aQÂ·e5bIÂ·e5bQ
asdaQ=sdaI.*sdbI.*sdbQ;          % *e5aQ = e5aIÂ·E5bIÂ·e5bQ
asdbI=sdbQ.*sdaI.*sdaQ;          % *e5bI = e5bQÂ·e5aIÂ·e5aQ
asdbQ=sdbI.*sdaI.*sdaQ;          % *e5bQ = e5bIÂ·e5aIÂ·e5aQ

%_____________________signal parameters_____________________________
% The parameters scE5-S and scE5-P represent the four-valued sub-carrier
% functions for the single signals and the product signals respectively:

sc_s=parameters('s');       %sc_s
sc_p=parameters('p');       %sc_p

samples=[sc_s(4) sc_s(3) sc_s(2) sc_s(1)];
dsc_s=[samples sc_s];
dsc_s=dsc_s(1:1:length(dsc_s)-4);   %sc_s delayed Ts/4

samples=[sc_p(4) sc_p(3) sc_p(2) sc_p(1)];
```

```
dsc_p=[samples sc_p];
dsc_p=dsc_p(1:1:length(dsc_p)-4);   %sc_p delayed Ts/4


% sub-carriers PLOT:

% figure(1);
% subplot(2,1,1),plot(t,sc_s); axis([2/fs 5/fs -1.3 1.3]); title('subcarrier
AS'); xlabel('2 samples/value');
% subplot(2,1,2),plot(t,sc_p); axis([2/fs 5/fs -1.3 1.3]); title('subcarrier
AP'); xlabel('2 samples/value');

%_____________________________________________________________________

%keeping on signal components...

A=1/(2*sqrt(2));

%e5(t)
s=A*(((sdaI+(j*sdaQ)).*(sc_s-(j*dsc_s)))+((sdbI+(j*sdbQ)).*(sc_s+(j*dsc_s)))+
((asdaI+(j*asdaQ)).*(sc_p-(j*dsc_p)))+((asdbI+(j*asdbQ)).*(sc_p+(j*dsc_p))));

Re=real(s);
Im=imag(s);

[Rss,lag]=xcorr(s,'coeff');

[RGs,fre]=shcenteredFFT(Re,fsamp);
[IGs,fim]=shcenteredFFT(Im,fsamp);

% PSD PLOT
figure(2);
subplot(2,1,1), plot(fre,abs(RGs)); title('E5 PSD'); ylabel('In-Phase');
xlabel('hertzs');xlim([-125e6 125e6]);
subplot(2,1,2), plot(fim,abs(IGs)); title('E5 PSD'); ylabel('Quadrature');
xlabel('hertzs');xlim([-125e6 125e6]);

% ACF PLOT
figure(3);
plot(lag,abs(Rss)); xlim([-40 40]);
title('ACF AltBOC(15,10)');
```

```
function sE5nopa

%Generates Galileo's signal E5 with AltBOC(15,10) modulation

fsamp=120*1.023e6;  % Sampling frequency:    92.07  Mhz
fs=15*1.023e6;      % Sub-Carrier frequency: 15.345 Mhz
fc=10*1.023e6;      % Chiprate:              10.23  Mcps
SraI=50;            % Symbol rate aI:        50     Sps
SrbI=250;           % Symbol rate bI:        250    Sps
Tsamp=1/fsamp;      % Sample time            10.861 ns
TraI=1/SraI;        % aI symbol time         20     ms
TrbI=1/SrbI;        % bI symbol time          4     ms

BaI=1;              % 2 bits ---> 40ms duration
BbI=5;              % 10 bits ---> 40ms duration

D=BaI*TraI;         % Duration of simulation
```

```
                    % It'd be the same if D=BbI*TrbI
t=0:Tsamp:D-Tsamp;

%++++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++

[CaI,CaQ,CbI,CbQ]=draftcdmaE5;

CaI=expan(CaI,fsamp,fc);
CaQ=expan(CaQ,fsamp,fc);
CbI=expan(CbI,fsamp,fc);
CbQ=expan(CbQ,fsamp,fc);

CaI=fulltime(CaI',BaI);
CaQ=CaQ(1:1:length(CaI));

CbI=fulltime(CbI',BbI);
CbQ=CbQ(1:1:length(CbI));

%figure(1);
%plot(t,CaI);axis([0 (900*Tsamp) -1.15 1.15]);

%++++++++++++++++++++++data stream++++++++++++++++++++++++++++++++

DaI=-sign(randn(1,BaI));
DbI=-sign(randn(1,BbI));

DaI=expan(DaI,fsamp,SraI);
DbI=expan(DbI,fsamp,SrbI);

%figure(2);
%subplot(2,1,1), plot(t,DaI);axis([0 .2 -1.1 1.1]); xlabel('seconds');
%subplot(2,1,2), plot(t,DbI);axis([0 .2 -1.1 1.1]); xlabel('seconds');

%++++++++++++++++++++++sub-carrier sequence++++++++++++++++++++++++

sc_s=sign(sin(2*pi*fs*t));
sc_p=sign(cos(2*pi*fs*t));

%figure(3);
%hold on
%plot(t,sc); axis([0 7.5/fs -1.1 1.1]);
%plot(t,fullCaI,'color','r'); axis([0 5/fc -1.1 1.1]);
%hold off;

%---------------------signal components----------------------------
sdaI=DaI'.*CaI;                 % e5aI
sdaQ=CaQ';                      % e5aQ
sdbI=DbI'.*CbI;                 % e5bI
sdbQ=CbQ';                      % e5bQ
asdaI=sdaQ.*sdbI.*sdbQ;         % *e5aI = e5aQÂ·e5bIÂ·e5bQ
asdaQ=sdaI.*sdbI.*sdbQ;         % *e5aQ = e5aIÂ·E5bIÂ·e5bQ
asdbI=sdbQ.*sdaI.*sdaQ;         % *e5bI = e5bQÂ·e5aIÂ·e5aQ
asdbQ=sdbI.*sdaI.*sdaQ;         % *e5bQ = e5bIÂ·e5aIÂ·e5aQ

%figure(4);
%subplot(2,1,1), plot(t,asdaI); axis([l1 l2 -1.1 1.1]); title('In-phase');
%subplot(2,1,2), plot(t,asdaQ); axis([l1 l2 -1.1 1.1]); title('Quadrature');

%sc_s
%sc_p
```

```
ind=length(sc_s);
samples=[sc_s(ind-4) sc_s(ind-3) sc_s(ind-2) sc_s(ind-1)];
dsc_s=[samples sc_s];
dsc_s=dsc_s(1:1:length(dsc_s)-4);   %sc_s delayed Ts/4

ind=length(sc_p);
samples=[sc_p(ind-4) sc_p(ind-3) sc_p(ind-2) sc_p(ind-1)];
dsc_p=[samples sc_p];
dsc_p=dsc_p(1:1:length(dsc_p)-4);   %sc_p delayed Ts/4
%
% figure(5);
% subplot(2,1,1),plot(t,sc_p); axis([0 3/fs -1.3 1.3]);
% subplot(2,1,2),plot(t,dsc_p); axis([0 3/fs -1.3 1.3]);

%_____________________________________________________________________


A=1/(2*sqrt(2));

%e5(t)
sE5=A*(((sdaI+(j*sdaQ)).*(sc_s-(j*dsc_s)))+((sdbI+(j*sdbQ)).*(sc_s+(j*dsc_s)))+
((asdaI+(j*asdaQ)).*(sc_p-(j*dsc_p)))+((asdbI+(j*asdbQ)).*(sc_p+(j*dsc_p))));

Re=real(sE5);
Im=imag(sE5);

[Rss,lag1]=xcorr(sE5,'coeff');
[Ree,lag2]=xcorr(Re,'coeff');
[Imm,lag3]=xcorr(Im,'coeff');

[RGboc,fre]=shcenteredFFT(Ree,fsamp);
[IGboc,fim]=shcenteredFFT(Imm,fsamp);
[Gboc,f]=shcenteredFFT(Rss,fsamp);

% PSD PLOT
% figure(6);
% subplot(3,1,2), plot(fre,abs(RGboc)); title('Real PSD NP'); ylabel('In-
% Phase'); xlabel('hertzs');
% subplot(3,1,3), plot(fim,abs(IGboc)); title('Imaginary PSD NP');
% ylabel('Quadrature'); xlabel('hertzs');
% subplot(3,1,1), plot(f,abs(Gboc));title('E5 PSD NP');

% ACF PLOT
% figure(7);
% subplot(2,1,1),plot(lag2,abs(Ree)); xlim([-40 40]); title('E5 ACF RE
% AltBOC(15,10) NP'); xlabel('samples');
% subplot(2,1,2),plot(lag3,abs(Imm)); xlim([-40 40]); title('E5 ACF IM
% AltBOC(15,10) NP'); xlabel('samples');

figure(8);
plot(lag1,abs(Rss)), xlim([-40 40]); xlabel('samples');
title('ACF AltBOC(15,10) NP');
```

```
function [RGs,IGs]=sE5n

% Generates Galileo's signal E5 with AltBOC(15,10) modulation and
% parameters with one eigth (T/8) of subcarriers' period.
%************************************************************
```

```
fsamp=240*1.023e6;  % Sampling frequency:    245.52 Mhz
fs=15*1.023e6;      % Sub-Carrier frequency: 15.345 Mhz
fc=10*1.023e6;      % Chiprate:              10.23  Mcps
SraI=50;            % Symbol rate aI:        50     Sps
SrbI=250;           % Symbol rate bI:        250    Sps
Tsamp=1/fsamp;      % Sample time            10.861 ns
TraI=1/SraI;        % aI symbol time         20     ms
TrbI=1/SrbI;        % bI symbol time          4     ms
BaI=1;              % 1 bits ---> 40ms duration
BbI=5;              % 5 bits ---> 40ms duration
D=BaI*TraI;         % Duration of simulation
                    % It'd be the same if D=BbI*TrbI
t=0:Tsamp:D-Tsamp;

%++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++++

[CaI,CaQ,CbI,CbQ]=draftcdmaE5;


CaI=expan(CaI,fsamp,fc);
CaQ=expan(CaQ,fsamp,fc);
CbI=expan(CbI,fsamp,fc);
CbQ=expan(CbQ,fsamp,fc);

CaI=fulltime(CaI',BaI);
CaQ=CaQ(1:1:length(CaI));

CbI=fulltime(CbI',BbI);
CbQ=CbQ(1:1:length(CbI));

% [CAI,fa]=shcenteredFFT(CaI,fsamp);
% [CBI,fb]=shcenteredFFT(CbI,fsamp);
%
% display(max(abs(CAI).^2));
% display(max(abs(CBI).^2));

%++++++++++++++++++++++data stream++++++++++++++++++++++++++++++++

DaI=1;                          %-sign(randn(1,BaI));
DbI=[1 1 -1 -1 -1];             %-sign(randn(1,BbI));

DaI=expan(DaI,fsamp,SraI);
DbI=expan(DbI,fsamp,SrbI);

% [DAI,fa]=shcenteredFFT(DaI,fsamp);
% [DBI,fb]=shcenteredFFT(DbI,fsamp);
%
% display(max(abs(DAI).^2));
% display(max(abs(DBI).^2));

%----------------------signal components----------------------------

sdaI=DaI'.*CaI;                 % e5aI
sdaQ=CaQ';                      % e5aQ
sdbI=DbI'.*CbI;                 % e5bI
sdbQ=CbQ';                      % e5bQ
asdaI=sdaQ.*sdbI.*sdbQ;         % *e5aI = e5aQÂ·e5bIÂ·e5bQ
asdaQ=sdaI.*sdbI.*sdbQ;         % *e5aQ = e5aIÂ·E5bIÂ·e5bQ
asdbI=sdbQ.*sdaI.*sdaQ;         % *e5bI = e5bQÂ·e5aIÂ·e5aQ
asdbQ=sdbI.*sdaI.*sdaQ;         % *e5bQ = e5bIÂ·e5aIÂ·e5aQ
```

```
% [SDAI,fa]=shcenteredFFT(sdaI,fsamp);
% [SDBI,fb]=shcenteredFFT(sdbI,fsamp);

% figure(1)
% subplot(1,2,1), plot(fa,abs(SDAI).^2);ylim([0 max(abs(SDBI).^2)]);
% subplot(1,2,2), plot(fb,abs(SDBI).^2);ylim([0 max(abs(SDBI).^2)]);

% display(max(abs(SDAI).^2));
% display(max(abs(SDBI).^2));


%_____________________signal parameters_______________________________

sc_s=parameters('s');       %sc_s
sc_p=parameters('p');       %sc_p

samples=[sc_s(4) sc_s(3) sc_s(2) sc_s(1)];
dsc_s=[samples sc_s];
dsc_s=dsc_s(1:1:length(dsc_s)-4);   %sc_s delayed Ts/4

samples=[sc_p(4) sc_p(3) sc_p(2) sc_p(1)];
dsc_p=[samples sc_p];
dsc_p=dsc_p(1:1:length(dsc_p)-4);   %sc_p delayed Ts/4

figure(3);
subplot(2,2,1),plot(t,sc_s); axis([0 3/fs -1.3 1.3]); title('subcarrier AS');
ylabel('2 samples/value');xlabel('seconds');
subplot(2,2,2),plot(t,sc_p); axis([0 3/fs -1.3 1.3]); title('subcarrier AP');
ylabel('2 samples/value');xlabel('seconds');
subplot(2,2,3),plot(t,dsc_s); axis([0 3/fs -1.3 1.3]); title('T/4 delayed
subcarrier AS'); ylabel('2 samples/value');xlabel('seconds');
subplot(2,2,4),plot(t,dsc_p); axis([0 3/fs -1.3 1.3]); title('T/4 delayed
subcarrier AP'); ylabel('2 samples/value');xlabel('seconds');

pcss=sc_s+j*dsc_s;
pcsp=sc_p+j*dsc_p;

% [PCSS,fs]=shcenteredFFT(pcss,fsamp);
% [PCSP,fs]=shcenteredFFT(pcsp,fsamp);
%
% figure(6)
% subplot(1,2,1),plot(fs,abs(PCSS).^2,'linewidth',2);title('pcsS
spectrum');xlabel({'hertzs';'';'\bf \it {zoom @ 15.345 Mhz}'});axis([15.3448e6
15.345200e6 0 1.7]);%xlim([-130e6 130e6]);
%
subplot(1,2,2),plot(fs,abs(PCSP).^2,'color','red','linewidth',2);xlabel({'hertzs
';'';'\bf \it {zoom @ 76.725 Mhz}'});title('pcsP spectrum');axis([76.724800e6
76.725200e6 0 .11]);%xlim([-130e6 130e6]);

% figure(7)
% plot(fs,abs(PCSS).^2,'linewidth',2); xlabel('hertzs');xlim([-130e6 130e6]);
% hold on
% plot(fs,abs(PCSP).^2,'color','red','linewidth',2);xlabel('hertzs');title('pcsS
and pcsP spectrum');xlim([-130e6 130e6]);
% hold off

%_____________________________________________________________________

%keeping on signal components...

A=1/(2*sqrt(2));
```

```
%*********************Tests with E5 signal expressions*********************

% e5(t)

% Expression 1:
% s1=A*(((sdaI+(j*sdaQ)).*(sc_s-(j*dsc_s)))+((sdbI+(j*sdbQ)).*(sc_s+(j*dsc_s)))+
((asdaI+(j*asdaQ)).*(sc_p-(j*dsc_p)))+((asdbI+(j*asdbQ)).*(sc_p+(j*dsc_p))));

% Expression 2:
% s2=A*(sdaI.*conj(pcss) + sdbI.*pcss + asdaI.*conj(pcsp) + asdbI.*pcsp +
j*(sdaQ.*conj(pcss) + sdbQ.*(pcss) + asdaQ.*conj(pcsp) + asdbQ.*pcsp));


%_____________________Real and Imaginary part for expression 1:

% Re1=A*(sdaI.*sc_s + sdaQ.*dsc_s + sdbI.*sc_s - sdbQ.*dsc_s + asdaI.*sc_p +
asdaQ.*dsc_p + asdbI.*sc_p - asdbQ.*dsc_p);
% Im1=A*(-sdaI.*dsc_s + sdaQ.*sc_s + sdbI.*dsc_s + sdbQ.*sc_s - asdaI.*dsc_p +
asdaQ.*sc_p + asdbQ.*sc_p + asdbI.*dsc_p);

% [RE,ff]=shcenteredFFT(Re1,fsamp);
% [IM,ff]=shcenteredFFT(Im1,fsamp);

% figure(1);
% subplot(2,1,1),plot(ff,abs(RE).^2);xlabel('hertzs');
% subplot(2,1,2),plot(ff,abs(IM).^2);xlabel('hertzs');

%_____________________Real and Imaginary part for expression 2:

% Re2=real(s2);
% Im2=imag(s2);
%
% [S1,f1]=shcenteredFFT(Re2,fsamp);
% [S2,f1]=shcenteredFFT(Im2,fsamp);
%
% figure(2);
% subplot(2,1,1),plot(f1,abs(S1).^2);xlabel('hertzs');
% subplot(2,1,2),plot(f1,abs(S2).^2);xlabel('hertzs');

%;;;;;;;;;;;;;;;;;;;;RESULTS:
%
% Both plots are the same. That means that it is not a thing of using the
% real() and imag() commands, but separating the signal in Real and Imaginary
% parts. Thus, the spectrum is mirrored. A cause of this must be because
% the signal must be Real and even, then, its Fourier transform is real and
% even as well...

% If now we try to plot the same signal but using expression 2 and
% transforming each of the channels separately (the ones in the in-phase
% component and the other ones in the quadrature component):

% inph=A*(sdaI.*conj(pcss) + sdbI.*pcss + asdaI.*conj(pcsp) + asdbI.*pcsp);
% quad=A*(sdaQ.*conj(pcss) + sdbQ.*(pcss) + asdaQ.*conj(pcsp) + asdbQ.*pcsp);
%
% [I,f]=shcenteredFFT(inph,fsamp);
% [Q,f]=shcenteredFFT(quad,fsamp);

% figure(3);
% subplot(2,1,1),plot(f,abs(I).^2);ylabel('In-Phase');xlabel('hertzs');
%
```

```
subplot(2,1,2),plot(f,abs(Q).^2,'color','r');ylabel('Quadrature');xlabel('hertzs
');

% Doing so, we can see that the four channels are different from each
% other and that the spectrum is not mirrored. Therefore, the point is how
% to manage expression 1 so that when we plot it, it shows what we want.


% If we do the same but transforming each of the different channels (I & Q):

x1=sdaI.*conj(pcss) + sdbI.*pcss;
y1=sdaQ.*conj(pcss) + sdbQ.*pcss;

x2=asdaI.*conj(pcsp) + asdbI.*pcsp;
y2=asdaQ.*conj(pcsp) + asdbQ.*pcsp;

[X1,f]=shcenteredFFT(x1,fsamp);
[Y1,f]=shcenteredFFT(y1,fsamp);
[X2,f]=shcenteredFFT(x2,fsamp);
[Y2,f]=shcenteredFFT(y2,fsamp);

figure(4);
subplot(2,2,1),plot(f,abs(X1).^2);title('DATA spectrum:  aI + bI');ylabel('In-
Phase');xlabel('hertzs');xlim([-130e6 130e6]);
subplot(2,2,2),plot(f,abs(Y1).^2);title('PILOT spectrum: aQ +
bQ');ylabel('Quadrature');xlabel('hertzs');xlim([-130e6 130e6]);
subplot(2,2,3),plot(f,abs(X2).^2);title('Dashed signals data
spectrum');ylabel('In-Phase');xlabel('hertzs');xlim([-130e6 130e6]);
subplot(2,2,4),plot(f,abs(Y2).^2);title('Dashed signals pilot
spectrum');ylabel('Quadrature');xlabel('hertzs');xlim([-130e6 130e6]);

RGs=abs(X1+X2).^2;
IGs=abs(Y1+Y2).^2;

figure(5);
subplot(2,1,1),plot(f,RGs);ylabel('In-Phase');xlabel('hertzs');title('E5aI  and
E5bI spectrum');xlim([-130e6 130e6]);
subplot(2,1,2),plot(f,IGs);ylabel('Quadrature');xlabel('hertzs');title('E5aQ
and E5bQ spectrum');xlim([-130e6 130e6]);
```

```
function [Sbb,Scc,SchA]=sE6

%Generates Galileo's E6 signal modulated with CASM. There are 3 channels: A,B&C.
%Channel A is modulated with BOCcos(10,5). Its ranging codes and
%navigation data are encrypted and they are not provided.
%Channels B & C are modulated with a simple BPSK with chiprate = 5.115 Mcps
%and symbol rate 1000 Sps.

fo=1.023e6;
fsamp=100*fo;
fs=10*fo;
fc=5*fo;

SrB=1000;
Tsamp=1/fsamp;

BCB=10;
BCA=BCB;
```

```
D=BCB/SrB;

t=0:Tsamp:D-Tsamp;

%++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++++
%E6 ranging codes are not provided in the SIS-ICD Draft 1   :(

[CB,CC]=cdmaE6;

% channel A=-sign(B) because it is not provided

CA=-CB;

CA=expan(CA,fsamp,fc);
CB=expan(CB,fsamp,fc);
CC=expan(CC,fsamp,fc);

CA=fulltime(CA',BCA);
CB=fulltime(CB',BCB);
CC=CC(1:1:length(CB));

%figure(1);
%subplot(3,1,1), plot(t,CA); axis([0 30/fc -1.2 1.2]); title('CDMA A');
%subplot(3,1,2), plot(t,CB); axis([0 30/fc -1.2 1.2]); title('CDMA B');
%subplot(3,1,3), plot(t,CC); axis([0 30/fc -1.2 1.2]); title('CDMA C');

%++++++++++++++++++++data stream++++++++++++++++++++++++++++++++

DCB=-sign(randn(1,BCB));
DCB=expan(DCB,fsamp,SrB);

DCA=sign(randn(1,BCA));
DCA=expan(DCA,fsamp,SrB);


%figure(2);
%plot(t,DCB); axis([0 5/SrB -1.2 1.2]); title('bits channel B');
%plot(t,DCA); axis([0 5/SrB -1.2 1.2]); title('bits channel A');

%++++++++++++++++++++signal components++++++++++++++++++++++++++

A=1/sqrt(2);

%E6 signal channel B&C BPSK:
%carrier=sin(2*pi*30*fo*t);          % to see the spectrum at 30 MHz

e6B=A*DCB'.*CB;                      % .*carrier; %.*sc;
e6C=A*CC';                           % .*carrier;     %.*sc;

sE6=(e6B-e6C);

[RE6,lag1]=xcorr(sE6,'coeff');
[SE6,fE6]=shcenteredFFT(RE6,fsamp);

[Rbb]=xcorr(e6B,'coeff');
[Rcc]=xcorr(e6C,'coeff');

[Sbb,fb]=shcenteredFFT(Rbb,fsamp);
[Scc,fc]=shcenteredFFT(Rcc,fsamp);

%E6_channelA  BOCcos(10,5);
```

```
sc=sign(cos(2*pi*fs*t));

chA=DCA'.*CA.*sc;

[RchA,lag2]=xcorr(chA,'coeff');
[SchA,fchA]=shcenteredFFT(RchA,fsamp);

T=SE6+SchA;

% PSD PLOT
figure(3);
subplot(3,1,1), plot(fE6,abs(T)); title('E6 PSD'); xlabel('hertzs');
subplot(3,1,2), plot(fE6,abs(SE6),'color','g'); title('Ch B&C PSD');
xlabel('hertzs');
subplot(3,1,3), plot(fchA,abs(SchA),'color','r'); title('Ch A PSD');
xlabel('hertzs');

% ACF PLOT
figure(4);
subplot(2,1,1), plot(lag1,RE6); xlim([-100 100]); title('Ch B&C ACF
BPSK');xlabel('samples');
subplot(2,1,2), plot(lag2,abs(RchA)); xlim([-100 100]); title('Ch A ACF
BOCcos(10,5)');xlabel('samples');
```

```
function[Sreal,Simag]=sL1

%Generates Galileo's L1 signal modulated with CASM. There are 3 channels
%A,B and C. B & C are modulated with CBOC(6,1,1/11) (composite BOC) and
%channel A ranging codes and navigation data are encrypted and they are
%not provided. It is modulated with a BOCcos(15,2.5).

fo=1.023e6;
fsamp=90*fo;
fs_a=fo;                          %sub-carrier parameter a
fs_b=6*fo;                        %sub-carrier parameter b
fs_A=15*fo;                       %sub-carrier channel A
fc_A=2.5*fo;                      %chip rate channel A
fc=fo;                            %chip rate channels B&C

SrA=625;
SrB=250;
Tsamp=1/fsamp;

BCA=5;
BCB=2;
D=BCB/SrB;

t=0:Tsamp:D-Tsamp;

%++++++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++

[CA,CB,CC]=cdmaL1;

CA=expan(CA,fsamp,fc_A);
CB=expan(CB,fsamp,fc);
CC=expan(CC,fsamp,fc);

CA=fulltime(CA',BCA);
```

```
CB=fulltime(CB',BCB);
CC=CC(1:1:length(CB));

%figure(1);
%subplot(3,1,1), plot(t,CA); axis([0 30/fc_A -1.2 1.2]); title('CDMA A');
%subplot(3,1,2), plot(t,CB); axis([0 60/fc_A -1.2 1.2]); title('CDMA B');
%subplot(3,1,3), plot(t,CC); axis([0 15/fc_A -1.2 1.2]); title('CDMA C');

%++++++++++++++++++++++++++data stream++++++++++++++++++++++++++++++++

DCA=-sign(randn(1,BCA));
DCA=expan(DCA,fsamp,SrA);

DCB=-sign(randn(1,BCB));
DCB=expan(DCB,fsamp,SrB);

%figure(2);
%subplot(2,1,1), plot(t,DCA); axis([0 5/SrA -1.2 1.2]); title('bits channel A');
%subplot(2,1,2), plot(t,DCB); axis([0 2/SrB -1.2 1.2]); title('bits channel B');

%++++++++++++++++++++++++signal components++++++++++++++++++++++++++

scA=sign(cos(2*pi*fs_A*t));             %sucbarrier channel A: 15fo

sca=sign(sin(2*pi*fs_a*t));             %subcarrier parameter a: fo
scb=sign(sin(2*pi*fs_b*t));             %subcarrier parameter b: 6fo

alfa=sqrt(10/11);                       %parameters for E1 signal
beta=sqrt(1/11);

scB=(alfa*sca)+(beta*scb);              %Subcarriers ch B&C for E1 signal
scC=(alfa*sca)-(beta*scb);
A=1/sqrt(2);

e1A=DCA'.*CA;
e1B=DCB'.*CB;
e1C=CC';

%----------------------E1 SIGNAL-----------------------------------
sE1=A*((e1B.*scB)-(e1C.*scC));

RE1=xcorr(sE1,'coeff');
[SE1,fE1]=shcenteredFFT(RE1,fsamp);

% figure(3);
% plot(fE1,abs(SE1)); title('E1 PSD');

%------------------------------------------------------------------

%----------------------L1 CASM SIGNAL------------------------------
% By Galileo SIS-ICD ---> fs is fo for both channels

eL1B=e1B.*sign(sin(2*pi*fo*t));
eL1C=e1C.*sign(sin(2*pi*fo*t));

%It is only provided that it's a BOCcos(15,2.5)

eL1A=e1A.*scA;

sCASM=(1/3)*((sqrt(2)*(eL1B-eL1C))+(j*((2*eL1A)+(eL1A.*eL1B.*eL1C))));
```

```
Re=(1/3)*(sqrt(2)*(eL1B-eL1C));%real(sCASM);
Im=(1/3)*((2*eL1A)+(eL1A.*eL1B.*eL1C));%imag(sCASM);

Rre=xcorr(Re,'coeff');
Rim=xcorr(Im,'coeff');
Rss=xcorr(sCASM,'coeff');

[Sreal,fre]=shcenteredFFT(Rre,fsamp);
[Simag,fim]=shcenteredFFT(Rim,fsamp);
[Ss,fs]=shcenteredFFT(Rss,fsamp);

[ReA,lag1]=xcorr(eL1A,'coeff');
[ReC,lag2]=xcorr(eL1C,'coeff');

% ACF PLOT
figure(4);
subplot(2,1,1), plot(lag1,ReA); xlim([-100 100]); title('ACF L1A
BOCcos(15,2.5)');xlabel('samples');
subplot(2,1,2), plot(lag2,ReC); xlim([-100 100]); title('ACF L1C
BOC(1,1)');xlabel('samples');

% PSD PLOT
figure(5);
subplot(3,1,1), plot(fs,abs(Ss)); title('L1 CASM PSD');xlabel('hertzs');
subplot(3,1,2), plot(fre,abs(Sreal),'color','g');xlabel('hertzs'); ylabel('In-
phase'); title('L1 CASM Real PSD');
subplot(3,1,3), plot(fim,abs(Simag),'color','r');xlabel('hertzs'); ylabel
('Quadrature'); title('L1 CASM Imaginary PSD');
```

```
function sE5ce

% checks if signal e5 is a complex envelope signal or not.

fsamp=240*1.023e6;  % Sampling frequency:    245.52 Mhz
fs=15*1.023e6;      % Sub-Carrier frequency: 15.345 Mhz
fc=10*1.023e6;      % Chiprate:              10.23  Mcps
SraI=50;            % Symbol rate aI:        50     Sps
SrbI=250;           % Symbol rate bI:        250    Sps
Tsamp=1/fsamp;      % Sample time            10.861 ns
TraI=1/SraI;        % aI symbol time         20     ms
%TrbI=1/SrbI;       % bI symbol time          4     ms

BaI=1;              %  1 bits ---> 40ms duration
BbI=5;              %  5 bits ---> 40ms duration

D=BaI*TraI;         % Duration of simulation
                    % It'd be the same if D=BbI*TrbI
t=0:Tsamp:D-Tsamp;

theta = linspace(0,2*pi,100);     % create vector theta
x = cos(theta);                   % generate x-coordinate
y = sin(theta);                   % generate y-coordinate

%++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++++

[CaI,CaQ,CbI,CbQ]=draftcdmaE5;

CaI=expan(CaI,fsamp,fc);
```

```
CaQ=expan(CaQ,fsamp,fc);
CbI=expan(CbI,fsamp,fc);
CbQ=expan(CbQ,fsamp,fc);

CaI=fulltime(CaI',BaI);
CaQ=CaQ(1:1:length(CaI));

CbI=fulltime(CbI',BbI);
CbQ=CbQ(1:1:length(CbI));

%++++++++++++++++++++data stream++++++++++++++++++++++++++++++++

DaI=-1;%-sign(randn(1,BaI));
DbI=[1 -1 1 1 -1];%-sign(randn(1,BbI));

DaI=expan(DaI,fsamp,SraI);
DbI=expan(DbI,fsamp,SrbI);

%---------------------signal components---------------------------
sdaI=DaI'.*CaI;                  % e5aI
sdaQ=CaQ';                       % e5aQ
sdbI=DbI'.*CbI;                  % e5bI
sdbQ=CbQ';                       % e5bQ
asdaI=sdaQ.*sdbI.*sdbQ;          % *e5aI = e5aQÂ·e5bIÂ·e5bQ
asdaQ=sdaI.*sdbI.*sdbQ;          % *e5aQ = e5aIÂ·E5bIÂ·e5bQ
asdbI=sdbQ.*sdaI.*sdaQ;          % *e5bI = e5bQÂ·e5aIÂ·e5aQ
asdbQ=sdbI.*sdaI.*sdaQ;          % *e5bQ = e5bIÂ·e5aIÂ·e5aQ

% figure(1)
% subplot(4,2,1), plot(t,sdaI); axis([0 10/fs -1.3 1.3]);
% subplot(4,2,2), plot(t,sdaQ); axis([0 10/fs -1.3 1.3]);
% subplot(4,2,3), plot(t,sdbI); axis([0 10/fs -1.3 1.3]);
% subplot(4,2,4), plot(t,sdbQ); axis([0 10/fs -1.3 1.3]);
% subplot(4,2,5), plot(t,asdaI); axis([0 10/fs -1.3 1.3]);
% subplot(4,2,6), plot(t,asdbI); axis([0 10/fs -1.3 1.3]);
% subplot(4,2,7), plot(t,asdaQ); axis([0 10/fs -1.3 1.3]);
% subplot(4,2,8), plot(t,asdbQ); axis([0 10/fs -1.3 1.3]);

%_____________________signal parameters_____________________________
%The parameters scE5-S and scE5-P represent the four-valued sub-carrier
%functions for the single signals and the product signals respectively:

sc_s=parameters('s');       %sc_s
sc_p=parameters('p');       %sc_p

%ind=length(sc_s);
samples=[sc_s(4) sc_s(3) sc_s(2) sc_s(1)];
dsc_s=[samples sc_s];
dsc_s=dsc_s(1:1:length(dsc_s)-4);   %sc_s delayed Ts/4

%ind=length(sc_p);
samples=[sc_p(4) sc_p(3) sc_p(2) sc_p(1)];
dsc_p=[samples sc_p];
dsc_p=dsc_p(1:1:length(dsc_p)-4);   %sc_p delayed Ts/4

pcss=sc_s+j*dsc_s;
pcsp=sc_p+j*dsc_p;

%_____________________________________________________________________

% keeping on signal components...
```

```
A=1/(2*sqrt(2));

% e5(t)

s=A*(sdaI.*conj(pcss) + sdbI.*pcss + asdaI.*conj(pcsp) + asdbI.*pcsp +
j*(sdaQ.*conj(pcss) + sdbQ.*(pcss) + asdaQ.*conj(pcsp) + asdbQ.*pcsp));
%s=A*(((sdaI+(j*sdaQ)).*(sc_s-(j*dsc_s)))+((sdbI+(j*sdbQ)).*(sc_s+(j*dsc_s)))+
((asdaI+(j*asdaQ)).*(sc_p-(j*dsc_p)))+((asdbI+(j*asdbQ)).*(sc_p+(j*dsc_p))));


figure(1)
plot(s);   axis([-1.1 1.1 -1.1 1.1]); xlabel('In-Phase'); ylabel('Quadrature');
title('\bf {Equivalent 8-PSK constellation for AltBOC(15,10)}');
hold on
plot(x,y,'color','m','linewidth',2);
axis('equal');
hold off

R=real(s);
I=imag(s);

% Peak-to-Peak Average Power Ratio -.-.-.-.- Defined as: 10 log (Ppeak/Paverage)

avrg=(s*s')/length(s);              % Mean Square Value: summation of all
samples divided by number of samples
peak=max(s.*conj(s));               % Peak Value: all samples to the square and
then we pick the highest one
papr=peak./avrg;                    % PAPR lineal
dBpapr=10*log10(papr);              % PAPR dB


figure(2)
subplot(1,2,1),regorcdfplot(papr,'PAPR values E5','CDF');
subplot(1,2,2),ccdfplot(papr,'PAPR values E5','CCDF');

figure(3)
subplot(2,2,1),regorcdfplot(R,'real values E5','CDF');
subplot(2,2,2),ccdfplot(R,'real values E5','CCDF');
subplot(2,2,3),regorcdfplot(I,'imaginary values E5','CDF');
subplot(2,2,4),ccdfplot(I,'imaginary values E5','CDF');
```

```
function sE6ce

% Checks the constant modulus envelope for signal E6

fo=1.023e6;
fsamp=100*fo;
fs=10*fo;
fc=5*fo;

SrB=1000;
Tsamp=1/fsamp;

BCB=10;
BCA=BCB;
D=BCB/SrB;
```

```
t=0:Tsamp:D-Tsamp;

theta = linspace(0,2*pi,100);     % create vector theta
x = sqrt(2)*cos(theta);                   % generate x-coordinate
y = sqrt(2)*sin(theta);                   % generate y-coordinate

%++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++
%E6 ranging codes are not provided in the SIS-ICD Draft 1   :(

[CB,CC]=cdmaE6;

CA=-CB;

CA=expan(CA,fsamp,fc);
CB=expan(CB,fsamp,fc);
CC=expan(CC,fsamp,fc);

CA=fulltime(CA',BCA);
CB=fulltime(CB',BCB);
CC=CC(1:1:length(CB));

%++++++++++++++++++++++data stream++++++++++++++++++++++++++++++++

DCB=[1 -1 1 -1 1 1 -1 1 -1 -1];%-sign(randn(1,BCB));
DCB=expan(DCB,fsamp,SrB);

DCA=[-1 1 -1 -1 -1 1 1 1 1 -1];%sign(randn(1,BCA));
DCA=expan(DCA,fsamp,SrB);

%++++++++++++++++++++++signal components++++++++++++++++++++++++++

A=1/sqrt(2);

% E6 signal channel B&C BPSK:

e6B=DCB'.*CB;
e6C=CC';

sBC=A*(e6B-e6C);                    % channels B&C in time as stated in OS SIS
ICD

% E6_channelA  BOCcos(10,5);

sc=sign(cos(2*pi*fs*t));

chA=DCA'.*CA.*sc;                   % channel A in time

% E6 Multiplexing technique as stated in Reference [11]

m=0.6155;                           % modulation index

sE6=(chA*cos(m)-e6C*sin(m))+j*(e6B*cos(m)+chA.*e6B.*e6C*sin(m));

In=real(sE6);
Q=imag(sE6);

figure(1)
plot(sE6);axis([-1.5 1.5 -1.6 1.6]); xlabel('In-
phase');ylabel('Quadrature');title('\bf \it {Constellation for E6 signal}');
hold on
plot(x,y,'color','m','linewidth',2);
```

```
axis('equal');
hold off

Rss=xcorr(sE6,'coeff');
Rii=xcorr(In,'coeff');
Rqq=xcorr(Q,'coeff');

[SE6,f]=shcenteredFFT(Rss,fsamp);
[IN,f]=shcenteredFFT(Rii,fsamp);
[QUA,f]=shcenteredFFT(Rqq,fsamp);

figure(2);
subplot(3,1,1), plot(f,abs(SE6));title('E6 PSD');
subplot(3,1,2), plot(f,abs(IN),'color','g');ylabel('In-
Phase');xlabel('hertzs');title('E6 Real PSD');
subplot(3,1,3),
plot(f,abs(QUA),'color','r');ylabel('Quadrature');xlabel('hertzs');title('E6
Imaginary PSD');

% Peak-to-Peak Average Power Ratio -.-.-.-.- Defined as: 10 log (Ppeak/Paverage)

avrg=(sE6*sE6')/length(sE6);        % Mean Square Value: summation of all
samples divided by number of samples
peak=max(sE6.*conj(sE6));           % Peak Value: all samples to the square and
then we pick the highest one
papr=peak./avrg;                    % PAPR lineal
dBpapr=10*log10(papr);              % PAPR dB

display(avrg);
display(peak);
display(papr);

figure(3)
subplot(2,2,1),regorcdfplot(In,'real values E6','CDF');
subplot(2,2,2),ccdfplot(In,'real values E6','CCDF');
subplot(2,2,3),regorcdfplot(Q,'imaginary values E6','CDF');
subplot(2,2,4),ccdfplot(Q,'imaginary values E6','CCDF');

figure(4)
subplot(1,2,1),regorcdfplot(papr,'PAPR values E6','CDF');
subplot(1,2,2),ccdfplot(papr,'PAPR values E6','CCDF');
```

```
function sL1ce

%checks the constant envelope of L1 signal

fo=1.023e6;
fsamp=90*fo;
fs_a=fo;                        %sub-carrier parameter a
fs_b=6*fo;                      %sub-carrier parameter b
fs_A=15*fo;                     %sub-carrier channel A
fc_A=2.5*fo;                    %chip rate channel A
fc=fo;                          %chip rate channels B&C

SrA=625;
SrB=250;
Tsamp=1/fsamp;
```

```
BCA=5;
BCB=2;
D=BCB/SrB;

t=0:Tsamp:D-Tsamp;

theta = linspace(0,2*pi,100);     % create vector theta
x = cos(theta);                   % generate x-coordinate
y = sin(theta);                   % generate y-coordinate

%++++++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++

[CA,CB,CC]=cdmaL1;

CA=expan(CA,fsamp,fc_A);
CB=expan(CB,fsamp,fc);
CC=expan(CC,fsamp,fc);

CA=fulltime(CA',BCA);
CB=fulltime(CB',BCB);
CC=CC(1:1:length(CB));

%++++++++++++++++++++++++++data stream++++++++++++++++++++++++++++++

DCA=-sign(randn(1,BCA));
DCA=expan(DCA,fsamp,SrA);

DCB=-sign(randn(1,BCB));
DCB=expan(DCB,fsamp,SrB);

%++++++++++++++++++++++++++signal components++++++++++++++++++++++++++

scA=sign(cos(2*pi*fs_A*t));             %sucbarrier channel A: 15fo

sca=sign(sin(2*pi*fs_a*t));             %subcarrier parameter a: fo
scb=sign(sin(2*pi*fs_b*t));             %subcarrier parameter b: 6fo

alfa=sqrt(10/11);                       %parameters for E1 signal
beta=sqrt(1/11);

scB=(alfa*sca)+(beta*scb);              %Subcarriers ch B&C for E1 signal
scC=(alfa*sca)-(beta*scb);
A=1/sqrt(2);

e1A=DCA'.*CA;
e1B=DCB'.*CB;
e1C=CC';

%----------------------E1 SIGNAL-----------------------------------

sE1=A*((e1B.*scB)-(e1C.*scC));

%---------------------L1 CASM SIGNAL------------------------------
% By Galileo SIS-ICD ---> fs is fo for both channels

eL1B=e1B.*sign(sin(2*pi*fo*t));
eL1C=e1C.*sign(sin(2*pi*fo*t));

%It is only provided that it's a BOCcos(15,2.5)

eL1A=e1A.*scA;
```

```
sCASM=(1/3)*((sqrt(2)*(eL1B-eL1C))+(j*((2*eL1A)+(eL1A.*eL1B.*eL1C))));

R=real(sCASM);
I=imag(sCASM);
pow=sCASM.*conj(sCASM);
display(min(pow));
display(max(pow));

figure(1);
plot(sCASM);axis([-1.1 1.1 -1.1 1.1]);xlabel('In-
Phase');ylabel('Quadrature');title('\bf \it{Constellation for L1 signal}');
hold on
plot(x,y,'color','m','linewidth',2);
axis('equal');
hold off

% Peak-to-Peak Average Power Ratio -.-.-.-.- Defined as: 10 log (Ppeak/Paverage)

avrg=(sCASM*sCASM')/length(sCASM);  % Mean Square Value: summation of all
samples divided by number of samples
peak=max(sCASM.*conj(sCASM));       % Peak Value: all samples to the square and
then we pick the highest one
papr=peak./avrg;                    % PAPR lineal
dBpapr=10*log10(papr);              % PAPR dB

% CDF and CCDF

figure(2)
subplot(1,2,1),regorcdfplot(papr,'PAPR values L1','CDF');
subplot(1,2,2),ccdfplot(papr,'PAPR values L1','CCDF');

figure(3)
subplot(2,2,1),regorcdfplot(R,'real values L1','CDF');
subplot(2,2,2),ccdfplot(R,'real values L1','CCDF');
subplot(2,2,3),regorcdfplot(I,'imaginary values L1','CDF');
subplot(2,2,4),ccdfplot(I,'imaginary values L1','CCDF');
```

```
function tE5

% this script implements the PSD but without doing the ACF, just
% transforming the signal in time and applying the square.

fsamp=240*1.023e6;    % Sampling frequency:    245.52 Mhz
%fs=15*1.023e6;       % Sub-Carrier frequency: 15.345 Mhz
fc=10*1.023e6;        % Chiprate:              10.23  Mcps
SraI=50;              % Symbol rate aI:        50     Sps
SrbI=250;             % Symbol rate bI:        250    Sps
%Tsamp=1/fsamp;       % Sample time            10.861 ns
%TraI=1/SraI;         % aI symbol time         20     ms
%TrbI=1/SrbI;         % bI symbol time          4     ms

BaI=1;                %  1 bits ---> 40ms duration
BbI=5;                %  5 bits ---> 40ms duration

%D=BaI*TraI;          % Duration of simulation
                      % It'd be the same if D=BbI*TrbI
%t=0:Tsamp:D-Tsamp;

%++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++++
```

```
[CaI,CaQ,CbI,CbQ]=draftcdmaE5;

CaI=expan(CaI,fsamp,fc);
CaQ=expan(CaQ,fsamp,fc);
CbI=expan(CbI,fsamp,fc);
CbQ=expan(CbQ,fsamp,fc);

CaI=fulltime(CaI',BaI);
CaQ=CaQ(1:1:length(CaI));

CbI=fulltime(CbI',BbI);
CbQ=CbQ(1:1:length(CbI));

%++++++++++++++++++++++data stream++++++++++++++++++++++++++++++++++

DaI=-1;%-sign(randn(1,BaI));
DbI=[1 -1 1 1 -1];%-sign(randn(1,BbI));

DaI=expan(DaI,fsamp,SraI);
DbI=expan(DbI,fsamp,SrbI);

%----------------------signal components----------------------------
sdaI=DaI'.*CaI;                  % e5aI
sdaQ=CaQ';                       % e5aQ
sdbI=DbI'.*CbI;                  % e5bI
sdbQ=CbQ';                       % e5bQ
asdaI=sdaQ.*sdbI.*sdbQ;          % *e5aI = e5aQÂ·e5bIÂ·e5bQ
asdaQ=sdaI.*sdbI.*sdbQ;          % *e5aQ = e5aIÂ·E5bIÂ·e5bQ
asdbI=sdbQ.*sdaI.*sdaQ;          % *e5bI = e5bQÂ·e5aIÂ·e5aQ
asdbQ=sdbI.*sdaI.*sdaQ;          % *e5bQ = e5bIÂ·e5aIÂ·e5aQ

%_____________________signal parameters_________________________
%The parameters scE5-S and scE5-P represent the four-valued sub-carrier
%functions for the single signals and the product signals respectively:

sc_s=parameters('s');       %sc_s
sc_p=parameters('p');       %sc_p

samples=[sc_s(4) sc_s(3) sc_s(2) sc_s(1)];
dsc_s=[samples sc_s];
dsc_s=dsc_s(1:1:length(dsc_s)-4);   %sc_s delayed Ts/4

samples=[sc_p(4) sc_p(3) sc_p(2) sc_p(1)];
dsc_p=[samples sc_p];
dsc_p=dsc_p(1:1:length(dsc_p)-4);   %sc_p delayed Ts/4

pcss=sc_s+j*dsc_s;
pcsp=sc_p+j*dsc_p;

% [PCSS,fs]=shcenteredFFT(pcss,fsamp);
% [PCSP,fs]=shcenteredFFT(pcsp,fsamp);

% figure(1)
% subplot(1,2,1),plot(fs,abs(PCSS).^2);title('subcarrier S');
% subplot(1,2,2),plot(fs,abs(PCSP).^2,'color','red');title('subcarrier P');
%
% figure(2)
% plot(fs,abs(PCSS).^2); %xlim([17.536800e6 17.537600e6]);
% hold on
% plot(fs,abs(PCSP).^2,'color','red')
```

```
% hold off

%_____________________________________________________________________

% keeping on signal components...

% A=1/(2*sqrt(2));

% e5(t)

% s=(((sdaI+(j*sdaQ)).*(sc_s-(j*dsc_s)))+((sdbI+(j*sdbQ)).*(sc_s+(j*dsc_s)))+
((asdaI+(j*asdaQ)).*(sc_p-(j*dsc_p)))+((asdbI+(j*asdbQ)).*(sc_p+(j*dsc_p))));
%
% sd=((sdaI+(j*sdaQ)).*(sc_s-(j*dsc_s)))+((sdbI+(j*sdbQ)).*(sc_s+(j*dsc_s)));
% data
%
% sp=((asdaI+(j*asdaQ)).*(sc_p-(j*dsc_p)))+((asdbI+(j*asdbQ)).*(sc_p+
(j*dsc_p)));  % pilot

% [S,fs]=shcenteredFFT(s,fsamp);
% [SD,f]=shcenteredFFT(sd,fsamp);
% [SP,f]=shcenteredFFT(sp,fsamp);


% sd can be also expressed as:
% sd=sdaI.*conj(pcss) + sdbI.*pcss + j*(sdaQ.*conj(pcss)+sdbQ.*pcss);

% s1=sdaI.*conj(pcss);
% s2=sdbI.*pcss;
% s3=sdaQ.*conj(pcss);
% s4=sdbQ.*pcss;

% [S1,f]=shcenteredFFT(s1,fsamp);
% [S2,f]=shcenteredFFT(s2,fsamp);
% [S3,f]=shcenteredFFT(s3,fsamp);
% [S4,f]=shcenteredFFT(s4,fsamp);

% figure(4)
% subplot(2,2,1), plot(f,abs(S1).^2);xlabel('hertzs');
% subplot(2,2,2), plot(f,abs(S2).^2);xlabel('hertzs');
% subplot(2,2,3), plot(f,abs(S3).^2,'color','red');xlabel('hertzs');
% subplot(2,2,4), plot(f,abs(S4).^2,'color','red');xlabel('hertzs');

% sp can be also expressed as:
% sp=asdaI.*conj(pcsp) + asdbI.*pcsp + j*(asdaQ.*conj(pcsp) + asdbQ.*(pcsp))

as1=asdaI.*conj(pcsp);
as2=asdbI.*pcsp;
as3=asdaQ.*conj(pcsp);
as4=asdbQ.*pcsp;

[AS1,f]=shcenteredFFT(as1,fsamp);
[AS2,f]=shcenteredFFT(as2,fsamp);
[AS3,f]=shcenteredFFT(as3,fsamp);
[AS4,f]=shcenteredFFT(as4,fsamp);

figure(5)
subplot(2,2,1), plot(f,abs(AS1).^2);xlabel('hertzs');
subplot(2,2,2), plot(f,abs(AS2).^2);xlabel('hertzs');
subplot(2,2,3), plot(f,abs(AS3).^2,'color','red');xlabel('hertzs');
subplot(2,2,4), plot(f,abs(AS4).^2,'color','red');xlabel('hertzs');
```

```
% figure(6)
% subplot(4,1,1),
plot(fs,abs(imag(S)).^2,'color','r');xlabel('hertzs');ylabel('quadrature');
% subplot(4,1,2), plot(fs,abs(real(S)).^2);xlabel('hertzs');ylabel('in-phase');
% subplot(4,1,3), plot(f,abs(SD).^2,'color','c');xlabel('hertzs');title('Data');
% subplot(4,1,4),
plot(f,abs(SP).^2,'color','m');xlabel('hertzs');title('Pilot');
```

```
function tL1

%Generates Galileo's L1 signal modulated with CASM. There are 3 channels
%A,B and C. B & C are modulated with CBOC(6,1,1/11) (composite BOC) and
%channel A ranging codes and navigation data are encrypted and they are
%not provided. It is modulated with a BOCcos(15,2.5).

fo=1.023e6;
fsamp=90*fo;
fs_a=fo;                          %sub-carrier parameter a
fs_b=6*fo;                        %sub-carrier parameter b
fs_A=15*fo;                       %sub-carrier channel A
fc_A=2.5*fo;                      %chip rate channel A
fc=fo;                            %chip rate channels B&C

SrA=625;
SrB=250;
Tsamp=1/fsamp;

BCA=5;
BCB=2;
D=BCB/SrB;

t=0:Tsamp:D-Tsamp;

%++++++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++

[CA,CB,CC]=cdmaL1;

CA=expan(CA,fsamp,fc_A);
CB=expan(CB,fsamp,fc);
CC=expan(CC,fsamp,fc);

CA=fulltime(CA',BCA);
CB=fulltime(CB',BCB);
CC=CC(1:1:length(CB));

%figure(1);
%subplot(3,1,1), plot(t,CA); axis([0 30/fc_A -1.2 1.2]); title('CDMA A');
%subplot(3,1,2), plot(t,CB); axis([0 60/fc_A -1.2 1.2]); title('CDMA B');
%subplot(3,1,3), plot(t,CC); axis([0 15/fc_A -1.2 1.2]); title('CDMA C');

[CDMAA,f1]=shcenteredFFT(CA,fsamp);
[CDMAB,f1]=shcenteredFFT(CB,fsamp);
[CDMAC,f1]=shcenteredFFT(CC,fsamp);

figure(1)
subplot(1,3,1), plot(f1,abs(CDMAA).^2);xlabel('hertzs');
xlim([-10e6 10e6]);
title('CDMA spectrum ChA     2.5 fo Mcps');
```

```
subplot(1,3,2), plot(f1,abs(CDMAB).^2);xlabel('hertzs');
xlim([-5e6 5e6]);
title('CDMA spectrum ChB     fo Mcps');
subplot(1,3,3), plot(f1,abs(CDMAC).^2);xlabel('hertzs');
xlim([-5e6 5e6]);
title('CDMA spectrum ChC     fo Mcps');

%++++++++++++++++++++++++++data stream++++++++++++++++++++++++++++++

DCA=-sign(randn(1,BCA));
DCA=expan(DCA,fsamp,SrA);

DCB=[1 -1];%-sign(randn(1,BCB));
DCB=expan(DCB,fsamp,SrB);


[SDCA,f2]=shcenteredFFT(DCA,fsamp);
[SDCB,f2]=shcenteredFFT(DCB,fsamp);

figure(2)
subplot(1,2,1), plot(f2,abs(SDCA).^2);xlabel('hertzs');
xlim([-625 625]);
title('Data spectra aI');
subplot(1,2,2), plot(f2,abs(SDCB).^2);xlabel('hertzs');
xlim([-250 250]);
title('Data spectra bI');

%figure(2);
%subplot(2,1,1), plot(t,DCA); axis([0 5/SrA -1.2 1.2]); title('bits channel A');
%subplot(2,1,2), plot(t,DCB); axis([0 2/SrB -1.2 1.2]); title('bits channel B');

%++++++++++++++++++++++++++signal components++++++++++++++++++++++++++

scA=sign(cos(2*pi*fs_A*t));             %sucbarrier channel A: 15fo

sca=sign(sin(2*pi*fs_a*t));             %subcarrier parameter a: fo
scb=sign(sin(2*pi*fs_b*t));             %subcarrier parameter b: 6fo

alfa=sqrt(10/11);                       %parameters for E1 signal
beta=sqrt(1/11);

scB=(alfa*sca)+(beta*scb);              %Subcarriers ch B&C for E1 signal
scC=(alfa*sca)-(beta*scb);

figure(3);
subplot(2,1,1), plot(t,scB); xlim([5/fs_b 100/fs_b]); title('sc
B');xlabel('seconds');
subplot(2,1,2), plot(t,scC); xlim([5/fs_b 100/fs_b]); title('sc
C');xlabel('seconds');

A=1/sqrt(2);

e1A=DCA'.*CA;
e1B=DCB'.*CB;
e1C=CC';

[E1A,f3]=shcenteredFFT(e1A,fsamp);
[E1B,f3]=shcenteredFFT(e1B,fsamp);
[E1C,f3]=shcenteredFFT(e1C,fsamp);

figure(4)
```

```
subplot(1,3,1), plot(f3,abs(E1A).^2);xlabel('hertzs');
xlim([-10e6 10e6]);
title('Data*CDMA ChA');
subplot(1,3,2), plot(f3,abs(E1B).^2);xlabel('hertzs');
xlim([-5e6 5e6]);
title('Data*CDMA ChB');
subplot(1,3,3), plot(f3,abs(E1C).^2);xlabel('hertzs');
xlim([-5e6 5e6]);
title('Data*CDMA ChC');

%----------------------E1 SIGNAL-------------------------------------
sE1=A*((e1B.*scB)-(e1C.*scC));

RE1=xcorr(sE1,'coeff');
[SE1,fE1]=shcenteredFFT(RE1,fsamp);

figure(5);
plot(fE1,abs(SE1));xlabel('hertzs');
ylabel('PSD A*((e1BÂ·scB)-(e1CÂ·scC))');
title('E1 PSD');


%---------------------------------------------------------------------

%----------------------L1 CASM SIGNAL-------------------------------
% By Galileo SIS-ICD ---> fs is fo for both channels

eL1B=e1B.*sign(sin(2*pi*fo*t));
eL1C=e1C.*sign(sin(2*pi*fo*t));

%It is only provided that it's a BOCcos(15,2.5)

eL1A=e1A.*scA;

sCASM=(1/3)*((sqrt(2)*(eL1B-eL1C))+(j*((2*eL1A)+(eL1A.*eL1B.*eL1C))));

Re=real(sCASM);
Im=imag(sCASM);

Rre=xcorr(Re,'coeff');
Rim=xcorr(Im,'coeff');
Rss=xcorr(sCASM,'coeff');

[Sreal,fre]=shcenteredFFT(Rre,fsamp);
[Simag,fim]=shcenteredFFT(Rim,fsamp);
[Ss,fs]=shcenteredFFT(Rss,fsamp);

[ReA,lag1]=xcorr(eL1A,'coeff');
[ReC,lag2]=xcorr(eL1C,'coeff');

figure(6);
subplot(2,1,1), plot(lag1,ReA); xlim([-100 100]); title('ACF L1A
BOCcos(15,2.5)');xlabel('samples');
subplot(2,1,2), plot(lag2,ReC); xlim([-100 100]); title('ACF L1C
BOC(1,1)');xlabel('samples');

figure(7);
subplot(3,1,1), plot(fs,abs(Ss)); title('L1 CASM PSD');xlabel('hertzs');
subplot(3,1,2), plot(fre,abs(Sreal),'color','g'); ylabel('In-phase'); title('L1
CASM Real PSD');xlabel('hertzs');
subplot(3,1,3), plot(fim,abs(Simag),'color','r'); ylabel ('Quadrature');
```

```
title('L1 CASM Imaginary PSD');xlabel('hertzs');
```

```
function sE5ab

% Shows how an AltBOC modulation is like, taking the data from the E5
% signal. This plot is without parametric subcarriers, but with complex
% rectangular subcarriers (as it is an AltBOC theoretically).

%************************************************************

fsamp=240*1.023e6;  % Sampling frequency:    245.52 Mhz
fs=15*1.023e6;      % Sub-Carrier frequency: 15.345 Mhz
fc=10*1.023e6;      % Chiprate:              10.23  Mcps
SraI=50;            % Symbol rate aI:        50     Sps
SrbI=250;           % Symbol rate bI:        250    Sps
Tsamp=1/fsamp;      % Sample time            10.861 ns
TraI=1/SraI;        % aI symbol time         20     ms
TrbI=1/SrbI;        % bI symbol time          4     ms

BaI=1;              %  1 bits ---> 40ms duration
BbI=5;              %  5 bits ---> 40ms duration

D=BaI*TraI;         % Duration of simulation
                    % It'd be the same if D=BbI*TrbI
t=0:Tsamp:D-Tsamp;

%++++++++++++++++++++++cdma sequences++++++++++++++++++++++++++++++

[CaI,CaQ,CbI,CbQ]=draftcdmaE5;

CaI=expan(CaI,fsamp,fc);
CaQ=expan(CaQ,fsamp,fc);
CbI=expan(CbI,fsamp,fc);
CbQ=expan(CbQ,fsamp,fc);


CaI=fulltime(CaI',BaI);
CaQ=CaQ(1:1:length(CaI));

CbI=fulltime(CbI',BbI);
CbQ=CbQ(1:1:length(CbI));

%++++++++++++++++++++++data stream++++++++++++++++++++++++++++++++++

DaI=-1;
DbI=[1 -1 1 1 -1];

DaI=expan(DaI,fsamp,SraI);
DbI=expan(DbI,fsamp,SrbI);

%-------------------------signal components----------------------------

sdaI=DaI'.*CaI;                 % e5aI
sdaQ=CaQ';                      % e5aQ
sdbI=DbI'.*CbI;                 % e5bI
sdbQ=CbQ';                      % e5bQ
asdaI=sdaQ.*sdbI.*sdbQ;         % *e5aI = e5aQÂ·e5bIÂ·e5bQ
asdaQ=sdaI.*sdbI.*sdbQ;         % *e5aQ = e5aIÂ·E5bIÂ·e5bQ
```

```
asdbI=sdbQ.*sdaI.*sdaQ;          % *e5bI = e5bQÂ·e5aIÂ·e5aQ
asdbQ=sdbI.*sdaI.*sdaQ;          % *e5bQ = e5bIÂ·e5aIÂ·e5aQ

%------------------------subcarriers-----------------------------------

c=cos(2*pi*fs*t);
s=sin(2*pi*fs*t);
cr=sign(c);
sr=sign(s);

crs=cr+j*sr;                     % complex rectangular subcarrier

% [CRS,fs]=shcenteredFFT(crs,fsamp);
% [CR,fs]=shcenteredFFT(cr,fsamp);
% [SR,fs]=shcenteredFFT(sr,fsamp);

% figure(1);
% subplot(3,1,1),plot(fs,abs(CRS),'linewidth',2, 'color','red');title('|
CRS(f)|');xlabel('hertzs');xlim([-80e6 80e6]);
% subplot(3,1,2),plot(fs,abs(CR),'linewidth',2); title('|
CR(f)|');xlabel('hertzs');xlabel('hertzs');xlim([-80e6 80e6]);
% subplot(3,1,3),plot(fs,abs(SR),'linewidth',2); title('|
SR(f)|');xlabel('hertzs');xlabel('hertzs');xlim([-80e6 80e6]);


%_______________________________Signal Generation_________________________

% e5(t)
% s=A*(((sdaI+(j*sdaQ)).*(sc_s-(j*dsc_s)))+((sdbI+(j*sdbQ)).*(sc_s+(j*dsc_s)))+
((asdaI+(j*asdaQ)).*(sc_p-(j*dsc_p)))+((asdbI+(j*asdbQ)).*(sc_p+(j*dsc_p))));

% This is another way of representing E5 signal:
% two first addends (1)
% two second addends (2).

% Radd1=((sdaI+sdbI).*sc_s)+((sdaQ-sdbQ).*dsc_s);
% Iadd1=((sdaQ+sdbQ).*sc_s)+((-sdaI+sdbI).*dsc_s);
%
% Radd2=((asdaI+asdbI).*sc_p)+((asdaQ-asdbQ).*dsc_p);
% Iadd2=(((asdaQ+asdbQ).*sc_p)+(-asdaI+asdbI).*dsc_p);

% An AltBOC modulation can be written like this without pilot channels:

% x=sdaI.*crs + sdbI.*conj(crs);
% or like:
% y=(sdaI+sdbI).*cr + j*(sdaI-sdbI).*sr;

% x1=sdaI.*crs;
% x2=sdbI.*conj(crs);

% [X,f3]=shcenteredFFT(x,fsamp);
% [Y,f3]=shcenteredFFT(y,fsamp);

% [X1,f3]=shcenteredFFT(x1,fsamp);
% [X2,f3]=shcenteredFFT(x2,fsamp);

% figure(2)
% subplot(1,2,1),
% plot(f3,abs(X).^2);title('|S(f)|Â²');xlabel('hertzs');
% subplot(1,2,2),plot(f3,abs(X1+X2).^2);title('|Ca+Cb|Â²');xlabel('hertzs');
%
```

```
% figure(3)
% subplot(1,2,1),plot(f3,abs(X1).^2);title('channel Ca');xlabel('hertzs');
% subplot(1,2,2),plot(f3,abs(X2).^2,'color','red');title('Channel
Cb');xlabel('hertzs');

% If pilot channels have to be added, the expression is longer...

% yy=((sdaI+sdbI).*cr - (sdaQ-sdbQ).*sr) + j*((sdaQ+sdbQ).*cr +(sdaI-sdbI).*sr);

% but can be also expressed like (shorter):

xx=sdaI.*crs + sdbI.*conj(crs) + j*(sdaQ.*crs + sdbQ.*conj(crs));

[XX,ff]=shcenteredFFT(xx,fsamp);
% [YY,ff]=shcenteredFFT(yy,fsamp);

xx1=sdaI.*crs;
xx2=sdbI.*conj(crs);
xx3=sdaQ.*crs;
xx4=sdbQ.*conj(crs);

[XX1,f1]=shcenteredFFT(xx1,fsamp);
[XX2,f1]=shcenteredFFT(xx2,fsamp);
[XX3,f1]=shcenteredFFT(xx3,fsamp);
[XX4,f1]=shcenteredFFT(xx4,fsamp);

figure(4)
subplot(2,2,1),plot(f1,abs(XX1).^2);title('channel Ca');xlabel('hertzs');xlim([-
80e6 80e6]);
subplot(2,2,2),plot(f1,abs(XX2).^2);title('channel Cb');xlabel('hertzs');xlim([-
80e6 80e6]);
subplot(2,2,3),plot(f1,abs(XX3).^2,'color','red');title('channel Ca`
');xlabel('hertzs');xlim([-80e6 80e6]);
subplot(2,2,4),plot(f1,abs(XX4).^2,'color','red');title('channel Cb`
');xlabel('hertzs');xlim([-80e6 80e6]);

figure(5)
subplot(1,2,1),plot(ff,abs(XX1+XX2).^2);title('E5aI +
E5bI');xlabel('hertzs');xlim([-90e6 90e6]);
subplot(1,2,2),plot(ff,abs(XX3+XX4).^2,'color','red');title('E5aQ +
E5bQ');xlabel('hertzs');xlim([-90e6 90e6]);
```

1.1 Secondary functions for navigation signals

```
function [p]=parameters(c)

% implements the parametric sub-carriers for signal E5

fsamp=240*1.023e6;
fs=15*1.023e6;
Ts=1/fs;
Tsamp=1/fsamp;
SraI=50;
TraI=1/SraI;
BaI=1;

D=BaI*TraI;
t=0:Tsamp:D-Tsamp;
```

```
sc=sign(sin(2*pi*fs*t));
for i=1:length(sc)
    if sc(i)==-1
        sc(i)=1;
    end
end

sq=sqrt(2);
AS=[((sq+1)*0.5) (0.5) (-0.5) ((-sq-1)*0.5) ((-sq-1)*0.5)  (-0.5) (0.5)
((sq+1)*.5)];
AP=[((-sq+1)*0.5) (0.5) (-0.5) ((sq-1)*0.5) ((sq-1)*0.5)  (-0.5) (0.5) ((-
sq+1)*0.5)];

if c=='s'
    A=AS;
else
    A=AP;
end

pasc=zeros(1,length(sc));
j=1;
i=1;
while i<=length(sc)
    pasc(i)=sc(i)*A(j);
    pasc(i+1)=sc(i+1)*A(j);
    %pasc(i+2)=sc(i+2)*AS(j); ----> in case fsamp=360 MHz
    j=j+1;
    i=i+2;
    if j==8
        j=1;
    end
end

p=pasc;
```

```
function [d]=expan(info,fs,fb)

% makes an oversampling of the symbols using the kron function.
% if we have an m-by-n object (X) and another p-by-q (Y) object
% the result will be an object Z of m*p-by-n*q
% so, if we want a matrix of 110-by-10 elements, we have to introduce
% a 1-by-10 object and a 110-by-1 object.

vinfo=ones(fs/fb,1);
dmatrix=kron(info,vinfo);
d=dmatrix(:);       % puts all the columns back to back in a vector

% This is the same process but manually:

%dmat=ones(fs/fb,1)*info;
%display (dmat);
%d=dmat(:);

%in case fs/fb is not a integer, vinfo will be a
%truncated vector and the result of the kron function
%is not the expected
```

```
longd=length(d);
longi=length(info);
a=fs/fb;
b=(a*longi)-longd;

if (longd~=(a*longi))
    last=d(end);
    rest=last*ones(b,1);
    d=[d;rest];
end
```

```
function [d]=fulltime(x,b)

% Repeats the signal until desired length

aux=[];
for i=1:b
    aux=[aux x];
end
d=aux;
```

```
function [X,freq]=shcenteredFFT(x,Fs)

% this is a custom function that helps in plotting the autocorrelation two-sided
% spectrum
% x is the signal that is to be transformed
% Fs is the sampling rate

N=length(x);

% frequency axis

if mod(N,2)==0
    k=-N/2:N/2-1; % N even
else
    k=-(N-1)/2:(N-1)/2; % N odd
end

T=N/Fs;
freq=k/T;  %the frequency axis

xs=fftshift(x); %to avoid error in the phase

% takes the fft of the signal, and adjusts the amplitude accordingly

X=fft(xs)/N; % normalize the data

X=fftshift(X); %shifts the fft data so that it is centered
```

```
function [handleCDF,stats] = ccdfplot(x,xl,tl)

%   CDFPLOT Display an empirical cumulative distribution function.
```

```
%   CDFPLOT(X) plots an empirical cumulative distribution function (CDF)
%   of the observations in the data sample vector X. X may be a row or
%   column vector, and represents a random sample of observations from
%   some underlying distribution.
%
%   H = CDFPLOT(X) plots F(x), the empirical (or sample) CDF versus the
%   observations in X. The empirical CDF, F(x), is defined as follows:
%
%   F(x) = (Number of observations <= x)/(Total number of observations)
%
%   for all values in the sample vector X. If X contains missing data
%   indicated by NaN's (IEEE arithmetic representation for
%   Not-a-Number), the missing observations will be ignored.
%
%   H is the handle of the empirical CDF curve (a Handle Graphics 'line'
%   object).
%
%   [H,STATS] = CDFPLOT(X) also returns a statistical summary structure
%   with the following fields:
%
%      STATS.min    = minimum value of the vector X.
%      STATS.max    = maximum value of the vector X.
%      STATS.mean   = sample mean of the vector X.
%      STATS.median = sample median (50th percentile) of the vector X.
%      STATS.std    = sample standard deviation of the vector X.
%
%   In addition to qualitative visual benefits, the empirical CDF is
%   useful for general-purpose goodness-of-fit hypothesis testing, such
%   as the Kolmogorov-Smirnov tests in which the test statistic is the
%   largest deviation of the empirical CDF from a hypothesized theoretical
%   CDF.
%
%   See also QQPLOT, KSTEST, KSTEST2, LILLIETEST.

% Copyright 1993-2004 The MathWorks, Inc.
% $Revision: 1.5.2.1 $   $ Date: 1998/01/30 13:45:34 $

% Get sample cdf, display error message if any

[yy,xx,n,emsg] = ccdfcalc(x);

% if ~isempty(eid)
%    error(sprintf('stats:cdfplot:%s',eid),emsg);
% end

% Create vectors for plotting
k = length(xx);
n = reshape(repmat(1:k, 2, 1), 2*k, 1);
xCDF    = [-Inf; xx(n); Inf];
yCDF    = [0; 0; yy(1+n)];

%
% Now plot the sample (empirical) CDF staircase.
%

hCDF = plot(xCDF , yCDF,'linewidth',2);
if (nargout>0), handleCDF=hCDF; end
grid  ('on')
xlabel(xl)
ylabel('1-Probability')
title (tl)
```

```
axis([-2 2 -0.05 1.05])

%
% Compute summary statistics if requested.
%

if nargout > 1
   stats.min    =  min(x);
   stats.max    =  max(x);
   stats.mean   =  mean(x);
   stats.median =  median(x);
   stats.std    =  std(x);
end
```

```
function [yCDF,xCDF,n,emsg] = ccdfcalc(x,xname)

%CDFCALC Calculate an empirical cumulative distribution function.
%   [YCDF,XCDF] = CDFCALC(X) calculates an empirical cumulative
%   distribution function (CDF) of the observations in the data sample
%   vector X. X may be a row or column vector, and represents a random
%   sample of observations from some underlying distribution.  On
%   return XCDF is the set of X values at which the CDF increases.
%   At XCDF(i), the function increases from YCDF(i) to YCDF(i+1).
%
%   [YCDF,XCDF,N] = CDFCALC(X) also returns N, the sample size.
%
%   [YCDF,XCDF,N,EMSG] = CDFCALC(X) also returns an error message if X
%   is not a vector or if it contains no values other than NaN.
%
%   See also CDFPLOT.

%   Copyright 1993-2002 The MathWorks, Inc.
%   $Revision: 1.5 $  $Date: 2002/01/17 21:30:11 $

% Ensure the data is a VECTOR.
yCDF = [];
xCDF = [];
if (nargin < 2)
   if isempty(inputname(1))
      xname = 'X';
   else
      xname = inputname(1);
   end
end
n = 0;
if (min(size(x)) ~= 1)
    emsg = sprintf('Input sample %s must be a vector.', xname);
    return;
end

% Remove missing observations indicated by NaN's.
x = x(~isnan(x));
n = length(x);
if n == 0
   emsg = sprintf('Input sample %s has no valid data (all NaN''s).', ...
                  xname);
   return;
end
```

```
% Sort observation data in ascending order.
x = sort(x(:));

%
% Compute cumulative sum such that the sample CDF is
% F(x) = (number of observations <= x) / (total number of observations).
% Note that the bin edges are padded with +/- infinity for auto-scaling of
% the x-axis.
%

% Get cumulative sums
yCDF = (1:n)' / n;

% Remove duplicates; only need final one with total count
notdup = ([diff(x(:)); 1] > 0);
xCDF = x(notdup);
yCDF = [0; 1-yCDF(notdup)];
emsg = '';
```

```
function [handleCDF,stats] = regorcdfplot(x,xl,tl)

%   CDFPLOT Display an empirical cumulative distribution function.
%   CDFPLOT(X) plots an empirical cumulative distribution function (CDF)
%   of the observations in the data sample vector X. X may be a row or
%   column vector, and represents a random sample of observations from
%   some underlying distribution.
%
%   H = CDFPLOT(X) plots F(x), the empirical (or sample) CDF versus the
%   observations in X. The empirical CDF, F(x), is defined as follows:
%
%   F(x) = (Number of observations <= x)/(Total number of observations)
%
%   for all values in the sample vector X. If X contains missing data
%   indicated by NaN's (IEEE arithmetic representation for
%   Not-a-Number), the missing observations will be ignored.
%
%   H is the handle of the empirical CDF curve (a Handle Graphics 'line'
%   object).
%
%   [H,STATS] = CDFPLOT(X) also returns a statistical summary structure
%   with the following fields:
%
%      STATS.min    = minimum value of the vector X.
%      STATS.max    = maximum value of the vector X.
%      STATS.mean   = sample mean of the vector X.
%      STATS.median = sample median (50th percentile) of the vector X.
%      STATS.std    = sample standard deviation of the vector X.
%
%   In addition to qualitative visual benefits, the empirical CDF is
%   useful for general-purpose goodness-of-fit hypothesis testing, such
%   as the Kolmogorov-Smirnov tests in which the test statistic is the
%   largest deviation of the empirical CDF from a hypothesized theoretical
%   CDF.
%
%   See also QQPLOT, KSTEST, KSTEST2, LILLIETEST.
```

```
% Copyright 1993-2004 The MathWorks, Inc.
% $Revision: 1.5.2.1 $   $ Date: 1998/01/30 13:45:34 $

% Get sample cdf, display error message if any

[yy,xx,n,emsg] = regorcdfcalc(x);

% if ~isempty(eid)
%    error(sprintf('stats:cdfplot:%s',eid),emsg);
% end

% Create vectors for plotting
k = length(xx);
n = reshape(repmat(1:k, 2, 1), 2*k, 1);
xCDF    = [-Inf; xx(n); Inf];
yCDF    = [0; 0; yy(1+n)];

%
% Now plot the sample (empirical) CDF staircase.
%

hCDF = plot(xCDF , yCDF,'linewidth',2);
if (nargout>0), handleCDF=hCDF; end
grid  ('on')
xlabel(xl)
ylabel('Probability')
title (tl)
axis([-2 2 -0.05 1.05])


% Compute summary statistics if requested.


if nargout > 1
   stats.min    =  min(x);
   stats.max    =  max(x);
   stats.mean   =  mean(x);
   stats.median =  median(x);
   stats.std    =  std(x);
end
```

```
function [yCDF,xCDF,n,emsg] = regorcdfcalc(x,xname)

%CDFCALC Calculate an empirical cumulative distribution function.
%   [YCDF,XCDF] = CDFCALC(X) calculates an empirical cumulative
%   distribution function (CDF) of the observations in the data sample
%   vector X. X may be a row or column vector, and represents a random
%   sample of observations from some underlying distribution.  On
%   return XCDF is the set of X values at which the CDF increases.
%   At XCDF(i), the function increases from YCDF(i) to YCDF(i+1).
%
%   [YCDF,XCDF,N] = CDFCALC(X) also returns N, the sample size.
%
%   [YCDF,XCDF,N,EMSG] = CDFCALC(X) also returns an error message if X
%   is not a vector or if it contains no values other than NaN.
%
%   See also CDFPLOT.
```

```
%   Copyright 1993-2002 The MathWorks, Inc.
%   $Revision: 1.5 $  $Date: 2002/01/17 21:30:11 $

% Ensure the data is a VECTOR.
yCDF = [];
xCDF = [];
if (nargin < 2)
   if isempty(inputname(1))
      xname = 'X';
   else
      xname = inputname(1);
   end
end
n = 0;
if (min(size(x)) ~= 1)
    emsg = sprintf('Input sample %s must be a vector.', xname);
    return;
end

% Remove missing observations indicated by NaN's.
x = x(~isnan(x));
n = length(x);
if n == 0
   emsg = sprintf('Input sample %s has no valid data (all NaN''s).', ...
                  xname);
   return;
end

% Sort observation data in ascending order.
x = sort(x(:));

%
% Compute cumulative sum such that the sample CDF is
% F(x) = (number of observations <= x) / (total number of observations).
% Note that the bin edges are padded with +/- infinity for auto-scaling of
% the x-axis.
%

% Get cumulative sums
yCDF = (1:n)' / n;

% Remove duplicates; only need final one with total count
notdup = ([diff(x(:)); 1] > 0);
xCDF = x(notdup);
yCDF = [0;yCDF(notdup)];
emsg = '';
```

2. Ranging Codes

```
function [seqE5aI,seqE5aQ,seqE5bI,seqE5bQ]=draftcdmaE5

% Implements CDMA codification for E5 signal using the scheme provided by
% Galileo OS SIS ICD.


Np=10230;
```

```
[tpaI,tpaQ,tpbI,tpbQ]=tappol;           % Tap polynomials
[svaI,svaQ,svbI,svbQ]=starval;          % Start values
[isaI,isaQ,isbI,isbQ]=initseq;          % Init Sequences for base register 2

NsaI=20; NsaQ=100; NsbI=4; NsbQ=NsaQ;   % Secondary code length
[scaI,scaQ,scbI,scbQ]=seccodeE5;        % Secondary codes


%----------------E5aI
a=1;
seqE5aI=[];
for i=1:NsaI
    pr=prilfsr(tpaI(1,:),Np);
    se=seclfsr(tpaI(2,:),Np,svaI(i,:),isaI(i,:));

    primary=xor(pr,se);
    secondary=scaI(a:1:(i*Np));
    a=a+Np;
    s=xor(primary,secondary');
    seqE5aI=[seqE5aI s];
end
seqE5aI=bipolar(seqE5aI);

%----------------E5bI
b=1;
seqE5bI=[];
for i=1:NsbI
    pr=prilfsr(tpbI(1,:),Np);
    se=seclfsr(tpbI(2,:),Np,svbI(i,:),isbI(i,:));

    primary=xor(pr,se);
    secondary=scbI(b:1:(i*Np));
    b=b+Np;
    s=xor(primary,secondary');
    seqE5bI=[seqE5bI s];
end
seqE5bI=bipolar(seqE5bI);

%----------------E5aQ
c=1;
seqE5aQ=[];
for i=1:NsaQ
    if i>50
        pr=prilfsr(tpaQ(1,:),Np);
        se=seclfsr(tpaQ(2,:),Np,svaQ(101-i,:),isaQ(101-i,:));
    else
        pr=prilfsr(tpaQ(1,:),Np);
        se=seclfsr(tpaQ(2,:),Np,svaQ(i,:),isaQ(i,:));
    end

    primary=xor(pr,se);
    secondary=scaQ(c:1:(i*Np));
    c=c+Np;
    s=xor(primary,secondary');
    seqE5aQ=[seqE5aQ s];
end
seqE5aQ=bipolar(seqE5aQ);

%----------------E5bQ
d=1;
seqE5bQ=[];
```

```
for i=1:NsbQ
    if i>50
        pr=prilfsr(tpbQ(1,:),Np);
        se=seclfsr(tpbQ(2,:),Np,svbQ(101-i,:),isbQ(101-i,:));
    else
        pr=prilfsr(tpbQ(1,:),Np);
        se=seclfsr(tpbQ(2,:),Np,svbQ(i,:),isbQ(i,:));
    end

    primary=xor(pr,se);
    secondary=scbQ(d:1:(i*Np));
    d=d+Np;
    s=xor(primary,secondary');
    seqE5bQ=[seqE5bQ s];
end
seqE5bQ=bipolar(seqE5bQ);
```

```
function [seqL1A,seqL1B,seqL1C]=cdmaL1

% CDMA ranging codes for L1 signal. Both codes are memory codes.

Np=4092;
Ns=25;
[mcB,mcC]=memcodL1;
[cB,cC]=memseqL1(mcB,mcC);      %sequences already in binary
                                %Both, cB and cC are two matrices
                                %of 50x4092. Each row represents
                                %a primary code.
seqL1B=cB(1:4092);
seqL1A=cB(4093:8184);           %Since the data is restricted, it is
                                %used one of the memory codes for
                                %channel B to complete channel A.

seqL1B=bipolar(seqL1B);
seqL1A=bipolar(seqL1A);


sec=htob('380AD90');
sec=sec(1:25);                  %last three are padded with 0
o=ones(4092,1);
xsec=kron(sec,o);
sec=xsec(:);

a=1;
seqL1C=[];
for i=1:Ns
    primary=cC(i,:);
    secondary=sec(a:1:(i*Np));
    a=a+Np;
    s=xor(primary,secondary');
    seqL1C=[seqL1C s];
end
seqL1C=bipolar(seqL1C);
```

```
function [seqE6B,seqE6C]=cdmaE6

Np=5115;
Ns=50;

Btappol1=[1 0 0 1 0 1 0 0 1 0 0 0 0 1];
Btappol2=[1 1 1 0 0 0 0 0 0 0 0 0 1 1];
Bstarval2=[0 1 0 1 0 1 1 1 0 0 0 0 0];

Ctappol1=[1 0 0 1 0 0 0 0 1 0 0 0 0 1 1];
Ctappol2=[1 0 0 0 0 0 1 1 0 0 1 1 1 0 1];
Cstarval2=[0 1 1 0 1 0 0 0 0 1 1 1 0 1];
Cseccode=[0 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0
1 1 1 0 1 1 0 0 1 1 0 0 0 1 0];

o=ones(5115,1);
seccode=kron(Cseccode,o);
seccode=seccode(:);

%--------------------------E6 cdma ranging code--------------------------------

pr=prilfsr(Btappol1,Np);
se=E6seclfsr(Btappol2,Np,Bstarval2);

primary=xor(pr,se);
seqE6B=[];
seqE6B=[seqE6B primary];                % primary is a logical array.
seqE6B=bipolar(seqE6B);                 % To convert it into double it has
                                        % to be done like that.
a=1;
seqE6C=[];
for i=1:Ns
    pr=prilfsr(Ctappol1,Np);
    se=E6seclfsr(Ctappol2,Np,Cstarval2);

    primary=xor(pr,se);
    secondary=seccode(a:1:(i*Np));
    a=a+Np;
    s=xor(primary,secondary');
    seqE6C=[seqE6C s];
end
seqE6C=bipolar(seqE6C);
```

2.1 Secondary functions for ranging codes

```
function [caI,caQ,cbI,cbQ]=tappol

coaI=['40503' '50661'];

cobI=['64021' '51445'];


caI=transf(coaI,10,15,2,'o');
caQ=caI;
cbI=transf(cobI,10,15,2,'o');
cbQ=cbI;
```

```
function [d1mat,d2mat,d3mat,d4mat]=starval

% It contains the start values for the second base register to create
% the primary code. These values are loaded into the LFSR every NP chips
% epoch


mataI=['30305', '14234', '27213', '20577', '23312', '33463', '15614', '12537',
'01527', '30236', '27344', '07272', '36377', '17046', '06434', '15405', '24252',
'11631', '24776', '00630', '11560', '17272', '27445', '31702', '13012', '14401',
'34727', '22627', '30623', '27256', '01520', '14211', '31465', '22164', '33516',
'02737', '21316', '35425', '35633', '24655', '14054', '27027', '06604', '31455',
'34465', '25273', '20763', '31721', '17312', '13277'];

mataQ=['25652', '05142', '24723', '31751', '27366', '24660', '33655', '27450',
'07626', '01705', '12717', '32122', '16075', '16644', '37556', '02477', '02265',
'06430', '25046', '12735', '04262', '11230', '00037', '06137', '04312', '20606',
'11162', '22252', '30533', '24614', '07767', '32705', '05052', '27553', '03711',
'02041', '34775', '05274', '37356', '16205', '36270', '06600', '26773', '17375',
'35267', '36255', '12044', '26442', '21621', '25411'];

matbI=['07220', '26047', '00252', '17166', '14161', '02540', '01537', '26023',
'01725', '20637', '02364', '27731', '30640', '34174', '06464', '07676', '32231',
'10353', '00755', '26077', '11644', '11537', '35115', '20452', '34645', '25664',
'21403', '32253', '02337', '30777', '27122', '22377', '36175', '33075', '33151',
'13134', '07433', '10216', '35466', '02533', '05351', '30121', '14010', '32576',
'30326', '37433', '26022', '35770', '06670', '12017'];

matbQ=['03331', '06143', '25322', '23371', '00413', '36235', '17750', '04745',
'13005', '37140', '30155', '20237', '03461', '31662', '27146', '05547', '02456',
'30013', '00322', '10761', '26767', '36004', '30713', '07662', '21610', '20134',
'11262', '10706', '34143', '11051', '25460', '17665', '32354', '21230', '20146',
'11362', '37246', '16344', '15034', '25471', '25646', '22157', '04336', '16356',
'04075', '02626', '11706', '37011', '27041', '31024'];

d1mat=transf(mataI,250,15,50,'o');
d2mat=transf(mataQ,250,15,50,'o');
d3mat=transf(matbI,250,15,50,'o');
d4mat=transf(matbQ,250,15,50,'o');
```

```
function [is1mat,is2mat,is3mat,is4mat]=initseq

% It contains the initial values that will be loaded in each LFSR for the
% second base register. For every epoch, the sequence will start with this
% bits

mataI=['3CEA9D' '9D8CF1' '45D1C8' '7A0133' '64D423' '23300D' '91CEF2' 'AA82DC'
'F2A17D' '3D84AE' '446D38' 'C514F2' '0C0184' '8767E0' 'CB8EFF' '93EBCD' '5D55CE'
'B19B7C' '5805FC' 'F99EA1' 'B23CE5' '8515E8' '436822' '30F77B' 'A7D629' '9BFAC7'
'18A25B' '69A39F' '39B27D' '454598' 'F2BC62' '9DDBC6' '332827' '6E2FCA' '22C6D5'
'E881D9' '74C4DB' '13AB03' '119323' '594886' '9F4D89' '47A3C0' 'C9ED53' '334994'
'1B2A30' '5513F3' '7831C1' '30B93A' '84D5B4' 'A5029C'];

mataQ=['515537' 'D67539' '58B2E5' '305914' '442710' '593CF8' '214AD7' '435EA6'
'C1A7D5' 'F0E94A' 'A8C239' '2EB63B' '8F0A46' '896DD4' '0245F1' 'EB0160' 'ED28B3'
'CB9F5B' '576592' 'A88811' 'DD3649' 'B59F42' 'FF81F6' 'CE8128' 'DCD55C' '79E450'
'B63460' '6D562B' '3A9010' '59CD72' 'C0211A' '28EB96' 'D7554B' '425126' 'E0DAFB'
```

```
'EF79F2' '18085D' 'D50CD8' '0447B9' '8DE877' '0D1FA0' 'C9FCF7' '48116D' '840BCC'
'152004' '0D4897' 'AF6D25' '4B7593' '71BB1B' '53DA0E'];

matbI=['C5BEA1' '4F6248' 'FD5488' '86277B' '9E39D5' 'EA7EDE' 'F28321' '4FB0C9'
'F0AB64' '79833B' 'EC2D91' '409B11' '397E16' '1E0FCD' 'CB2F5A' 'C1079A' '2D9BC6'
'BC5146' 'F848B0' '4F01E8' 'B16C9B' 'B2827D' '16C809' '7B570F' '1969C0' '512FA9'
'73F36B' '2D5317' 'EC8390' '380374' '46B4DE' '6C01D9' '0E0BB6' '2708C7' '265B55'
'A68E1C' 'C3916E' 'BDC595' '1327D0' 'EA921F' 'D45869' '3EB98A' '9FDE16' '2A04CA'
'3CA56F' '03928A' '4FB5B9' '101EC7' 'C91D4F' 'AFC22B'];

matbQ=['E49AF0' 'CE701F' '54B709' '641AB1' 'FBD0AE' '0D8BC9' '805FA5' 'D86BA0'
'A7E921' '067E55' '3E4B58' '7D82FB' 'E33BC2' '31372C' '46676F' 'D2613E' 'EB443C'
'3FD0B1' 'FCB7CF' 'B83815' '48224A' '0FEE25' '38D33B' 'C135B9' '71DE13' '7E8CFB'
'B536C3' 'B8E68C' '1E7272' 'B75B69' '533F65' '812B41' '2C4DE1' '759E2C' '7E6434'
'B43640' '05671B' '8C6FE0' '978D4E' '5319BF' '516499' '6E4292' 'DC86A3' '8C46BE'
'DF0B03' 'E9A5B2' 'B0E553' '07DBAC' '4778E4' '37AF4F'];

is1mat=transf(mataI,300,24,50,'h');
is2mat=transf(mataQ,300,24,50,'h');
is3mat=transf(matbI,300,24,50,'h');
is4mat=transf(matbQ,300,24,50,'h');
```

```
function [scaI,scaQ,scbI,scbQ]=seccodeE5

aI='842E9';
aQ=['83F6F69D8F6E15411FB8C9B1C'];
bI='E';
bQ=['CFF914EE3C6126A49FD5E5C94'];


scaI=htob(aI);
scaQ=transf(aQ,25,100,1,'h');
scbI=htob(bI);
scbQ=transf(aQ,25,100,1,'h');

o=ones(10230,1);

scaI=kron(scaI,o);
scaI=scaI(:);

scbI=kron(scbI,o);
scbI=scbI(:);

scaQ=kron(scaQ,o);
scaQ=scaQ(:);

scbQ=kron(scbQ,o);
scbQ=scbQ(:);
```

```
function [s]=prilfsr(con,epoch)

% Implements a LFSR behaviour just giving the tap polynomial (in binary).
% It is truncated with a period of "epoch" chips.
% Start values are always 1 when is reset

L=length(con);
```

```
R=L-1;              % number of registers
r=ones(1,R);        % initial state: all 1's
tap=fliplr(con);    % tap polynomial:1 x (R+1)

nt=length(tap);
nr=R;
seq=zeros(1,epoch);

for i=1:epoch
    aux=r(nr);
    for j=1:(R-1)
        if (tap(nt-j)==1)
           r(nr-j+1)=xor(r(nr-j),aux);
        else
            r(nr-j+1)=r(nr-j);
        end
    end
    r(1)=aux;
    seq(i)=aux;
end
s=(seq);
```

```
function [s]=seclfsr(con,epoch,starval,iniseq)

% Implements a LFSR behaviour ginving the tap polynomial (in binary).
% It is truncated with a period of "epoch" chips.
% Start values are given as an argument of the function
% Every "epoch" chips, the LFSR is reset witn "iniseq".

L=length(con);
R=L-1;                      % number of registers

bin=starval;
r=bin;                      % vector long 15.
r=r(2:1:length(r));         % get rid of MSB (14 registers)
tap=fliplr(con);            % tap polynomial: 1 x (R+1)

r=fliplr(r);

nt=length(tap);
nr=R;
seq=zeros(1,epoch);

for i=1:epoch
    aux=r(nr);
    for j=1:(R-1)
        if (tap(nt-j)==1)
           r(nr-j+1)=xor(r(nr-j),aux);
        else
            r(nr-j+1)=r(nr-j);
        end
    end
    r(1)=aux;
    seq(i)=aux;
end

bin2=iniseq;
seq=[bin2, seq];
seq=seq(1:1:epoch);
```

```
s=(seq);
```

```
function [s]=E6seclfsr(con,epoch,starval)

% Implements a LFSR behaviour ginving the tap polynomial (in binary).
% It is truncated with a period of "epoch" chips.
% Start values are given as an argument of the function
% Every "epoch" chips, the LFSR is reset witn "iniseq".

L=length(con);
R=L-1;                      % number of registers

bin=starval;
r=bin;                      % vector long 14.
tap=fliplr(con);            % tap polynomial: 1 x (R+1)
r=fliplr(r);                % The order of the registers is flipped because
                            % otob returns the most significant register as
                            % a first element.

nt=length(tap);
nr=R;
seq=zeros(1,epoch);

for i=1:epoch
    aux=r(nr);
    for j=1:(R-1)
        if (tap(nt-j)==1)
           r(nr-j+1)=xor(r(nr-j),aux);
        else
            r(nr-j+1)=r(nr-j);
        end
    end
    r(1)=aux;
    seq(i)=aux;
end
s=seq;
```

```
function [p]=bipolar(b)
% transforms the sequence into +1 -1 values

for i=1:length(b)
    if b(i)==0
        b(i)=-1;
    end
end
p=b;
```

```
function [matB,matC]=memcodL1
% Two first memory codes for signals E1B (L1B) and E1C (L1C)

matB=['F5D710130573541B9DBD4FD9E9B20A0D59D144C54BC7935539D2E75810FB51E494093A0A1
9DD79C70C5A98E5657AA578097777E86BCC4651CC72F2F974DC766E07AEA3D0B557EF42FF57E6A58
E805358CE9257669133B18F80FDBDFB38C5524C7FB1DE079842482990DF58F72321D9201F8979EAB
159B2679C9E95AA6D53456C0DF75C2B4316D1E2309216882854253A1FA60CA2C94ECE013E2A8C943
```

```
341E7D9E5A8464B3AD407E0AE465C3E3DD1BE60A8C3D50F831536401E776BE02A6042FC4A27AF653
F0CFC4D4D013F115310788D68CAEAD3ECCCC5330587EB3C22A1459FC8E6FCCE9CDE849A5205E70C6
D66D125814D698DD0EEBFEAE52CC65C5C84EEDF207379000E169D318426516AC5D1C31F2E18A65E0
7AE6E33FDD724B13098B3A444688389EFBBB5EEAB588742BB083B679D42FB26FF77919EAB21DE038
9D9997498F967AE05AF0F4C7E177416E18C4D5E6987ED3590690AD127D872F14A8F4903A12329732
A9768F82F295BEE391879293E3A97D51435A7F03ED7FBE275F102A83202DC3DE94AF4C712E9D006D
182693E9632933E6EB773880CF147B922E74539E4582F79E39723B4C80E42EDCE4C08A8D02221BAE
6D17734817D5B531C0D3C1AE723911F3FFF6AAC02E97FEA69E376AF4761E6451CA61FDB2F9187642
EFCD63A09AAB680770C1593EEDD4FF4293BFFD6DD2C3367E85B14A654C834B6699421A'

'96B856A629F581D1344FEF597835FE60434625D077ECF0D95FBE1155EA0431979E5AFF544AF591A
332FDAEF98AB1EDD847A73F3AF15AAEE7E9A05C9D82C59EC325EF4CF264B8ADF2A8E8BA459354CB4
B415CC50BF239ADBC31B3A9C87B0843CF3B9E6D646BA43F866276B053826F3A2334CC5E2EFB9F8F1
95B382E75EEA63F58A06B3F82A3B5C77C1800FD9498F803E524435B321210BB84690BED0BBBE16D3
63B3A90656A73720E27008852FB7DACC8284411B177728D9527C560859084A395A6F11A96AD9DB6B
43E00642B000ED12BFD967868EAB1108552CD4FC89FBC408ACE7678C381EC91DD000319124EB5D5E
F52C4CAC9AADEE2FA045C16CE492D7F43743CA77924C78696FCBF2F9F7F36D8E623752200C6FCBBD
71ABBB6877F3C5D6E6740AB0389458A6B66440858B2D383244E853646FE2714211DEA9E619625281
5BB704A20BFE556AC474F8998944E0CABBBE21A6400B87BFDCF937D12B2821D59298AF4AD378F0F4
2BD8C41693B8D993CF37C8B478F3BB5D33AD2A9FA24AD7B8FA895FDBC04964192F7BA3FF74E0E3A4
35B5DFE042E3115CACF29624C0645E9C917534A2EBC1F5665E4E1B1BC56208DBCD8A27CCB6474D5D
0E20CA4072C960E5ACE41BDA3770DF3B681F2B318F6F8E1CB17C2857350FB6009AED665E13B2780D
79217F73FAC7A8A48048DB0FB8A8A5007CDDC9A7B2DA8257C99F1CB605A18204'];

matC=['B39340CA1C817D81EF4FAE4E95BF3504A7709089FB48560E9E3EF802180E85EB2194E0590
2C6C4C52021FEB7EC64FD416BCEBC8E39D64A4B5EE345291911AB8204A888C25B1CD3D9342A56C53
8636D3EAB957037D09E879AE5F3A39834FBB84A3D8D5090D7814246B62E9CA68533D2EC403B4FB94
88467FF9758B0D15A8CEF89187A1D5897880040B6C3C5244E85A2AD14BCF2F5ABC44A7B1D4A87E8B
DA05766218773ED4F70F8D1D07CBB1E8CA6298E64EE6DC5886D37495BA2EDB3E0B0B68AD9F300310
B88898DDEEFD484538C31A9BCAA76ECAD0C16607D32189058B0862EE9D70CEA9D304755CE8037BA4
C46C2573181748A212E4B2BDD04F9BC240518273DC17CBAFF21A03E9120FA7DCA18D56DD1D9A7E51
0C90CF219104385F531F2EFAFD185ECB6B911F9B7809D98D86F15516FFDDBE9BD1CF8662EB777C3F
94EA3F962D7B79449FAAD39935429E92CAE5637E9BCF4E94D413D27934952409AB536BE4055AFBC4
330CD1E4B5509EFE5F8EFC9ECBE9EF377DE7E37C479BB9D3EE7745E4609B0A6D2C5D92EB3C9E2278
C1F2221FF907596AA5E096ACF8990EBA907E43AD320F8019CB6355A2BA8670EE5A4F463E8E56F8F1
D3E7F4922510FB668E32C4CF23AD8496399638B095B47833E0CBB34977EB3E4242EAF870D86660D6
A73F83E45D6E8A41EDCA3815079649544597C5C43B6C93FEBAD5700D22EDAF431FD340'

'A64F94BB47BD4033C76D4924305907EC1F618B43C7535F3CFC093E5AF5DDD5C4339F3BB6D835B5C
2C2053CD3D5693368D4E1A7CAC59425D1FD96809C67285CFD3FC05B01053CB0773221D7205778022
F487BF99D1650566BE287FD7AE882AA8E8F52E5D4E3C0C2F971C9FF70AA378691EBD8ADE45CF2138
22D09FD05243F9726F6C69893845E57C37A7643E16B770E26F431FF69D437271905D270EB85D8D22
9D7D87662121F0BEEB1E895ED9589A9CF5833408A04197AC9025D8570AD9B75DB7E192EA0A089504
996E9DC652975D83633619CFF80667D8B519536B3475248BA8213C8A4C66DE69B4B3774BF9142425
C57F34A27B1E288119E3FFCC6AF6A21087F9394F09DDFBD42F32D059B8CD4104A519BA640765D5CD
E490E62F10E695FBFD33CBC9D2208A532C8EC25DA28B8CC1B6850AB43D9B5C00B6E74B7A148791AB
07B328D347058C7E6233E18C5ED172C9F9E9ACF29D913E2A1614BFC0893D4967ED033B2B9AE6B51F
908F1CED57C14FEEA85CD4D9711216BE7F79FA6721B7DCCA033C80127AC6E5FCF58EB4005EC24CB4
886D787355362D5E7031B9B2AC2A86D730AD734181E723A811FF510A4DF868001973FE83288D78E6
F9B9441DAF5BE2974A2848FD917C3BCD346A431922246EC852E4AAD467E60C15D61DD3BF4A207BB5
7DB45DCADEFE3210BE74B9DACC918A394469F2E2C95AD1E211947948FE24F5E4'

'FD1F6976002C39C87187C44E3D224ED4DF0B67750105944C651A5E57798F168A136AC0FB5979C4E
847A82B20A2E6C45DB42EF2B930A80D3257BCCC53EDA966F5DCD9AD47CFB226EED9B62A874E9F640
4D4087798A1005F4131171D3A47907A3CD602B83DABE094D2CB031867DF4595F3ED59FD8C4D76EDE
EE59E422CE5C7D0A5F720BE94FA24DF05F758348EADD5EFE9197C6BB2292E2B14DB8C6DB24AA94C5
FF0F5106D2B566058D32C58B63A150784F7B02478D9973DD4CFD2E84059AE0F4F1320754B7EE83F0
4A51C67EFFC2EB1C301C0C58DBAEBE95474E3484A76500103C14C40BB0B7D3A04D8BDABB605C1EF9
FD4A6564934DEC50BD5878243AEE80F9796EED70CE1B1E8B55725DF76472D12D4A7A487989F42E67
05818B1F7E149E97153A7B05A82FA3FBE51763E61171A4E12931472E94CCBA74CC09483DF93623FC
60945070FDDF3A00B561650427E4BD64D675B1EB398B35EF057A66FD0B48EDBABBDCD57C32ABAE46
```

```
F5CDD0CB1FCF17765258236F3DE40BD5D0A3C5C978D81DEB07367AB20B2CAA9834B9576161C4F20F
B9C184A01DC9021A4E92B71333354E05BBEA9015E5AC4C66312E8B79F0B92279AC7EF1936BCC3080
2B83DB3D113BEF64452CAD7ACF6674FDA44023A661019841A101BE80FDA4E3210AE774E433A9ABD9
7F2755259AECE21F7A8C3B1A3D471F874D2EEC85B9B21BC0C2E2EC9016F847C6'

'EE38BAF6F61704B01509B5210A0534E4702F93190C392E749869B5572BB7AC4D7120E2BECD6618C
D376C4C1B4965F7D9D73400824E88A5C7B5B66BA88C3E0065F9628A9AC6B91A1882192FC553E3140
349934D20698C9F291B5370948AF6CC90C837B9F3607F13CAFD492CEF1723376E6A5B813A56301B8
8A8799519CB7646F33F91C44CDBE7F768D7DD9B323A5002D2F784C4101AF90D6E4C5ADE7D085C79E
827D43E10DF63AC70BCDF13DCE0471B487C5ECB752B9C3E20F75DBD243790C91355ADFD7199081BF
EA03D80E82445EC2831FB5014B85EFC2A52748A8ABFAC1BA3904E178DFBAB26C1750228C9A031104
F58BB3B91905EDB9EADF7B0F6DF22ACEB0DE944E277809D77507D18EAEDAA1767697398421115D04
AB2EBFC466E99F0AA540482A49C6AC8FF95E3F962734B03EF39873A93B70470B46FFFDFDC15C89F8
FE2F4637B59F9BF9C5752D9F8AE7EA75D1EAF1C22CA27E5D5C9499624105D61BE2A691F9194D2774
14532A5E6C63875F7F20DD13C6EE73B0C3568392B14A5042843926472ABA343D2C427792199B543B
E1D43A178FAA7ECF53B98AB7528D8E1B8B82C52D973CA0427636505837F94284E8D6B4F496FC5A48
B7958D4681DA00651B8A7BC56EC859C071E4396A05F33588B8087EFE9635E565E6B5A8A70DA70F50
ECAD1A85E6E36FF07B4FB3B9119EDE0B611CFA91D9D4C58C1F4815B07B9EB1DE'

'CD37D0FB0043D03444A939E93676B9DAF5F2D19A2615E3D97D624E62ACAC8098099FDB9A5A2F4B3
ACF20F75B6807A5A3F157C2C0F479158F4A10FB4972855F3AE2FDCBDEEC00A4D470AADF5F5E57181
8AD6E872D897E2DDC402006965ADF16582B1E06B1861BF7D0C7E7BA491C79E86224AF6B246317F72
5FA74DD8376D63D7993FE2F2BBBB2F1DA9238C6F3FFCAEC50FF61E645FADEB6E03F883892C42CCCF
904708B123C9271A670D4DCFCD602951D12F5213937CA2C05ADDE9EE3908E99AAE8DA31951C36D36
D671CD7BF15DF60B707F00BF6EBBE5476926D015628A85758BFF35C4AC540F39E761B2ED3CA9116E
8680E28BC387058E0F69345CC6AB3AD160E9F2BC4D6047A1934E15D3D7A242A296333C09296981BB
F3B8577E4B8ED2A3624866111F6638F8955431195B60C5C089F9897DDF0D34A3DC627CE337AC8128
C28B63A394908E4C083BCC4522DB8CE5720C45EF76B2716225E53405FCAAAA72AC198226575D5225
195F106C1249E4B87AC05287A3ABE6C51A2A41E07F56ECDC46E989A8568D35669B525A6FFCA90DC9
1D3013967F6A5F4C022FFCC17751B68FB0D8F16FC9229851DFDCC060838F923BD44C1AD70A993E8E
BAC1667DA80F91B66F8F5B375D35275188E3C7702C2312CEAC5B20D67BB34400401BDF1DBFE79DFA
0EB73F173A04807215DA5CE8E1D28F2126424C3DB44ADCD7A961260FDBCAB31E'

'CAA02DD19DB9C721EB35AB7D64B8A387796427242698A47D832C3F1AD4DDA0B5926FFCE9319EEED
A1565ECB0FA1EEDB424414120AAE8CFD0BE88D4D248899A0BCE31F9BEE7A4DC4DB3C3B10444FAD6A
DCCE28F0EDF7B808536ACF5EB05AADAE92693EE02C9512B3EEF000844BA35E24620A2E8935354B84
32C07C8FD615534BCFD0D8E3B572BF2CF06AD343997590FE8B244A32BBE69125B5D7C5E513A49372
4EEA8DA6CB0FFF3ACF1C5085A8120694CBC40FAE1A6326FD71487CC3BE7C10A34315CDFFA8C618B6
8EA93D330945586B080381F0076351B888087F56B969E6D6A311AE03CC79FF6861E715C9DA9AEE75
1F1220661581C75DCEC0515A1C9259B9CF8E944CEC4B1754E5809E985D6F43FE45710893242ADE0D
3B84F1E1942B7A95648611595FED13F546CA11DB8E5A55A3C3C78C3793C6689E1B3AFB5F67526A48
0DF923A586A779F94A09CF963594FF4B0A387876EBB3E8FAB888C97F6773E7F0317B038E47DD7D10
9545BB07263B1AA84284B86E47FFB9784A171D101E7B0A6D38BCAE7E63D827C999BF551728FFC642
EE690B01D486CB6EBEEB9D5C888112589EA5CBC9BDF49E675965223416D6DA02D2333BFD4614706B
F13373973207C849A0DE41EBA137FDF79A1EB25D74E30CF60B577C2787DF04740BA8CADE3F9DA55D
3F0084F02809E37543239E0A71E99751EEB21CB3B41488244193A4868CBA9276'

'FB227530F82BD527E648619E532D7646A5ABBD15DB91A6E7033DFECCC65D095A3D83AB77EDD2F3F
EC52659CB3AD1BEB009D7A1C9BFB544291EC1C67B75DD6DAB06E70C32C714983139DE4A41EE07B4F
3C03BF566558484F19A3BB674B6795F0D8537BC31BC8D7A38B2FF1B2EC8B78539B2251D0E385DE48
4B05A411477681A3AE7527AC98BC2943AF1CF7F09ACF2DDE4530AE896BDE1266FE916E833A1C0CAA
2B2D2F5985AD47B2D0D1D3AFB6E50D4B3DA7DEEC4385E6CA8FE22760F92807AC55556AAF7973E801
6ADFD43A3919088B768351B1057498D2D668D7C1E8C63438055FDF7D36C5E7DF02FCAFCBD9291A21
49E7B429B3202D329E47CED51EA5771772E308C5BEBA7B934597540D83DBEC6C3BC61A96EA4CB2D7
530D9D760AA9403338CD95B829F17547C5A90D161F7B8CE0037EBF403C91C0D0C70C589BA87CAE8D
F26CF14281E235A686CCD10E2D520A76265C4C2780EDFD0705E89EFE3C953FE760DE45A8CF1F2D3F
36DE3164D5BC2CF32204228ADD7C182EC55F1158AFA9358BE179C722ADAF1D0BF1306A0B56218857
FC5C21001499F61E273442281E585B3E6DCE148AA97B6622B23BDAECF983BF186F1B34962764758A
C3C20C84036061D49CA33B3C3FCDF03F47F7E53B940DBB6E1E4A26702A118E525A9A0EC229085C92
5D133750ED0B200CB28A113289DE143D1D5839D2AF8B0525E0027F34FF32106A'
```

```
'9E5DA18A19514CCC849E9697AE4BD1B317BB34927D0461A96A7AF4A5D6C13107FFB9DE38C5E8CB7
C5682827F57D94ED2E77D36F9F1CB05E4C2C62B1DE254C7B1CB236FC4ED70BF8DD1F43AC773C16A3
7392B895F8B157578C477C85E53FA7CA58BE70D9187AF5F7A18D5A1E5642335E46C2F8F4691AEEE6
A9692E21B9668E2C083D9F45C2DB3E991588BA87A0A23808732EE39E8B3C876BE79227C782F07EE3
FB3086AF913D71D71910A0F56D62B5DE5E224F7856A42A4A1B2AFE380827BE86E381FCE486FD08A9
1B22BD91D09615F417E178C5593E41B0917E075133960AD28B4DD4096D1E84BEF1363098DDE92C29
CD508C40BA7E785F46C1E0DC72E729D394911DA919EA6F94D14567FFADC61CEB8DCA2821B1CF0484
77E2433E9DC718DE618EDEEF302CDCB5DE472656D6687DC41EA34C2BB4DF1CA08DCB933BE3EF4B41
9158BA0B68AE82A64ADD58559214FD88A4CB34D99F646310697DA982C2FD4EE069DC1CB102125C34
A89AB20F17B6EF648A834627320410FF6881C7919AE4E71CBAE5F8200E523934D84BFA897C44B89B
9BC6BC0129F7F97EE0EC049BA1AFD67D00CD624A75FF5A30514399BE4801CED057B498B9DBBF0EB9
944295D5B6AE968C4B8BBD2B9A9E17A3039C5FA35A0D30AA54CA426C58353943DDDD3FD185895C0D
AEE950455FC131F520B46AE118C7406D0A72BE6127C5307730AD441B6FC3D1E0'

'8589F8396F5B1C54CAF2B17D4C152CEF347E66EC7903C878F2823D4ADB9E7CCFAFEBB926B7EEB4A
E1BECA339A027CE8EF997957532FA871F356E0326ECE0BCE3399F81179BF78C5C7D135018ABC340C
0BE58D3063DD7CDA4C1918A0187BACF830C8B6900D43B62E04DF6E831CFEFA13BDB5E873A527F243
27C95DB4BBDB65C81A20F959F828F5DAE4DC13E5CAC7417EE089401FB497ABE10144E28EA383E61D
4A9B63B618AA7CEA4588B2911EC581F506062B05E7BEF723A5A465C9FBE70E313753BDE3102845A7
9A206BF7D996F49A21752D534B73EE83B48C1A225F85F5103DDB9B6B8380F61AAF26E5CA643EB62E
AF58AFEE0D3494E4F7A4F642A3454F4F56A406A264148FF5DAC9DF5F151C12E89ED9D4FDCC04EC5F
0022DF8CBAF3CBC67CED2853FB4F8C5894C96CD00550950E7EA2A26C80A72DF533270A0E23EDBAA4
D0BE935D62CC885E1CCE653D66C51E49C43952042E1B2D043BDA1CFFC1E98A3F806EB587A4EC9AE2
99BD838C68B9BBF7C420C12B23AA2793FA0248C932A91BCDD641DCB38F0B2D7187D8986928DF4602
B381BA13B263291134628FC91C8EDE92594B39650B877D9A91DAAA05295457DFB2C5D8207BBCDFE1
6AC5B93600E33BC970B38E18808B1A732889320352B524B109560136E605D32784CA01F8B11D077C
81EAD6B7A5741C82D76CEEF764FD07E361D531B75106AF1572AD1375B2BBAB68'

'A3E17A4CAD2ABE76E32D18501899F8D60D293BB1AC3ADB64F81148AF56741790F87F8B7A2D9A6E7
645EA50B75514C394508884CBF9E320B24D41D8246EB3C163B9101240776C312DB63C33889E3C121
8435850471C454486DF7FF4D2DC0AAA14980F394CC8EB7B828A60C53A2FEC3315BEAEB30045B3E65
006C6EBB23B47A8A069EAD45E32E771B9C467B4359EBB681AB48C891ABB796544169178203BCC4BC
6B4A278DCEFACE5E9385C059346A23DCCA001FC9E47CFEED4BCBDD947B12A3F7E5FF8B9372D9497E
E1A508D8BD3392BF3CFAD58F0191B18F6A300FF9CB8D914FDF37B48BF24C2C5CA76ABDFCCF833D51
D48FC90E06E7B972944BCBAD169232A8429B6100BA562F7F3C55A625A1870A7C7D7BC9BD4C478327
8CD95D07F89E8010E78876547F9AEC44322B0029A922B2922634ECCF2BBB47BF87909C494049550F
1E6D03BB5354DEA7E777F499D2D6239BFA5C1CFA536F8CB16F4DB9EAD96F83A4AD34AE2C6893ECD6
994C89E7F4FE426D95A18F93B88CB357996B8E5A34C43533EDB1F28A8162FCBEF03704FCCCD80C32
874F345D34E81EE813DF5CC9B9C299362F8443AABE91BD0EAB9746E431804B6129FD32916303A570
323FA121F7AEB2829F2A50A82CACCF6D273FFBD7AC6FFC5807771D216F50742F7091946F91460115
989C87E8BBBC8402B4C8B95C102CAB53843D581FA9F16C0ECCE8944E5FC4BF4C'

'9D7B1CF0029261D65AE1F021DAFA81CF1673C9E0B47FF2C37D1B1AF46E7A91BC5E529C8F93EE3BC
74E92B2743AAB1EDE16A6523B5B8A591C617C1FD0150E63F3B7EF0494162437B0FD555A83A3BDB51
9B3BB209EF7924D6BCDE5992BA6248690442E72CD5EB64B4C3D3F7DA339108A18B61AD88ABE87BB7
C85A3A352D7B882FD683B2637A17A2D9CB0B7F41456DCFA66D62913F145600BAAEEE7EFA5071C3C9
E6FDD0A6779A737071FA6965978CBC89776386B108DD7216FCE962FA87A26B29FE0E732309C0124B
0C1E99E5642E5EAE670005B078C097D16C58B8923633C18FDB0E8FF8C4610B789387ACB5A2DD0B6A
E7E0DF43A6A9E8C3B89C7E5D628D59759C58D07E0687812AEDAEEDBC63B4FEE8524D10E4B4676969
57E6791C1E94B13CADCD0ED60752C2DB1B65E035EA72F89FC679138D3609FD2A30E4DD1A94641825
3C67AA69B07EBB95D4973F562CE3773430007A6DB77271D5F2B342CC5E76E115178F9C7B1600554F
5C794961BAE81A5E9B621BA17851008BED9B556E461A553FE9BE00A40891750E4EA4B475216283B5
30CB8D479DC70B026E07889229F6017552AB9E01EDE6703FD1E2D59AF0B71E0F1DC9A42ACC582332
4BEFC52CA0DCD25FE8B10C999152AA3676A30602D3506F78751477033DB7AB1A2EDC21A6FE51273B
6B2890088703CEFE74F9EA89881896E5BE124B1FC9430B92F0C0568F5A068A80'

'F23088E3EAA0A6BA04D0633AAFE85203E8B1829223FA6B730F6DEE6799B521F2E8323B8793D0F7F
2BB9305B3EF4F5B4F1CB822836E4D92C8E4928A851BCE688329DECA6F7285DCC85195E5BDA3B503B
8AEE6F1CD7FBB158444E7DE8BF6A9A3CDA311787755A827BCAD3DA5621908EA913C0316B9B52BFB0
7ADADEFF17D3766BB450DD71328A0353B09DC24DE93CF83A2E5F98BA9D612187B601157D6B140E67
5228B58C9398618C3BF0D11A226E489366102B9C35A916653F0DB36711ACBA5F32B327F5789F3EF4
```

```
8A338E4676F4BC2C6A1308597171903D2AA299CE7E523C2ABE4B15AA4FC48954187E0097583EB099
419047244B4931326E5923B6313DE08423DB00866374ABBF5C31A00542CB97CDFB8F71046AA2A6DB
FD7E1A71C068ED70E8D7C3268EA3E0EEF2262BD7991B6C59FF471F73A4E85F4FA015E164F9C15FE0
AA5F4772BF2D62B26D3EAA25CE83EAEC5EB3577CA83A68168FB64C40A7A155905CBA6E64159E55EB
C928D125E55165C639F545B0071EE3CF1A3F58B4994BB4BF50C2B24F2E06E4ADC90BC1C0954A257D
88444347AAECF136C15242633463DCF984BB6736666E38F1A45150B1B7D1C31DE06EB9C2F4097E9D
9B4D21EBC9F3A918000DE2449DCB3F5FDDC3C773A645DF560F7E013E847E2356D33EFF1E21578263
8F58034B09F4739F98915BFB0B1DC124681492F58021670D03CBF5E8F962351E'

'EB07F9EDF03596ADC2A3B7EB6DB1CFC911E9A4C42336A57309F7B6C3389282E557D94BCC71827D7
C5737B1C530D2A087E3F507242F3DA5BD1BBCA4DF8B78BEEC1DBF7EBB2EA1CF1DFA79E60785BAFA2
3658490C9A64AC61C45779DFAFC6C55CB5C9FE457BF47E45A3FEF092E178ED4495C0357B459E95AA
C82132FF1C8044F4EC84EB882DC195D9CE996B1CCF523098E9E1A57C37C2E2D0ACB0EAA34B0B56FE
5A0747130B1E75AA923F6F94C0D024A7FCD22E7A4ED8B201966C417AE864420767AB3223BFF56C64
D4F8F557DD950F7C50D9A39AB2C742CE686C8F92B35711904C600A9D4D3DD83F3DF1ED7DB8042C76
B0B7D5D9BCD6E0B5524184BF99D8D0B4F14967FA48A93A2F44E2275ED7E59F3991EFB0CBF2E26AC1
F8D9A41AAE4563179254BA37028867E68C8179454B8B71FAB49DBD1F889104CFB64C8121151364BD
B64BAF854B0DA22B8620BD7EE3D4302A88A115F8BFBA649CAA9EE7EF5BC95CFAB26503A9D2603337
0A4EF3CB8A5D094C63305A833387B4F8371C6FE1987514BB458C571E6CB5DF5FC900631652D3FA44
44F8F1F0312204340FDB2092F709FDC51D2680753131ABC33712B4F1067EA1CC87C40B281E69209E
DEC42C22A88950E9C1CE8130DA9291897BF2D8D1D106911743E7A9DA36220FA90A02A34EB0B28543
217839374EBE79F40B3B612236C902E4CD05CE2E1C07F3DA10E2AEE8E387494E'

'E9D537A821DEDE526B441BA4252785779B54DE76F82747F8607B8952DF990F268C039CC792883B1
C76C297D81C6C0CF17DA8BA2C71110B16741728725839D33B5942BC0A5614A3650675FDA5D70F291
54A429A42819D6EDE324C64596F93E84CC9B2C9DA3717AA6DFFCD03B75AC96543020A9F202462035
3E1364E4320FD44933799FFF083E73F5D20B83BF77EC2247964ECE442C3213DE99026F8FAF0E9630
2EC60067EA38C5CA0CD989475205FA38869E349FC7F79EB81F8457CA3D1A875A8D166C96EBAF1F39
C88815E2258EA1A14943298DA39EB9B738AAA4E0035F9567A0A9D572785594496316D56EB3D39E1F
3F243D4F16111E194FC537A635FAEB2FB4401CAA9EE0091CF3CB28B366CB5446A6D3B10AB86B4B1A
0714D107FCCBBD50EAE520D56A1161E03849192F5096346FBE5150B6D04025A564A43A3D22BD4B7E
10DD4061CE20FA2ECDD36F66BAAD7EA96CDBAA0F063B814707718F47278F8570F77F3B15799D0E35
4CCA50DAA38C31C746B17482297D9C089FF379454FCCCB8730D89B1462AD95426370AC37DE50B775
B952663B97AFBC403F6F729BB9CC1D21DD89EE78AF09DF8558F7E68B3711A7D9075DD4754174802F
52CB9683FFE746471C7E543FF388D024327D1866CC5CA6775C58A14D70A3ECCD3EFAB52F9AE6CCE1
46766A8419FB546E39EB604F43B15AB88C72741F8C7D0A7FE2F462D360676D6E'

'D79D916241BBE52B61BE8210A02543F75A47032E9C0CC128524A675E94D8F79A69B6842B0C5CFF5
C1AC98D2085299BDBAEA67A41C724CA36B6275A80D377DC3A6EB4C8D0B6B88241334A95300B53FFB
546163D2889D7C85F1D1397924F126DA76085BEF131A65C7DDF60DDF4086BD33B44D25025D689FF4
1E0C256EA12F4353D9E722EE37907AA8BED0A5A606333A031AC6B9A16614250916759B72FE6C1828
BC6C1966C9EBCD51413A77F41F808BCA2534AC49DB1D32D37878DF5CC0BEFCC099C56CAF50D8B92E
7CE616AA026EA1D81DC7ABC17C4705F9B57A0F99FA749F30F93DFA982A083EAE6582C8461A11ABA7
4B11663ED7D66EB4F8DE14F090EB1CA6D8D81CB6B063A391FD354DCEAF7DB71C277D0E92B4B46387
3DCBEFFB698BDCA17F80845EFD5F0FF150ADDC9D7797E21E4279B54BDD4B7C9D403D9FA6101604B7
9AC377780A5461499714082942313CF74AD1147CD10571A31D82871B6B3A055D50C6CDA4BDDF3871
F41EFDAEBE8ABB995344DB6366E35C6E506907AD7FC76632F99124A58A32C86360FD6DDBF50324D8
6694518AC44F1FA19662C0EF0C0860811B5B976A96EC2A1449E53A7E4A07923E9F85794F228E441D
92903922E5783F2FA21C677251B6B8DB02AC2E242C0C8652E0C17C9E3858E52DE78DC712B2DD5D2A
F9A42DB2E2BEB3FB6E0FFF13DB9A1E02C8F84FCEF3F7C4D2DDC09F2A2813E8C2'

'F8E2DACDD88277D482951555C657B3E3C5DB79E5A43500F7A2C8B30C854DBE611FAC1087FA03D43
9AC4635D39211E234B82A91248DEE5D4FE67A02D5AE25C676E64C4843E419EBB3C4D81FB606B9CA0
836F8207CD19D106C0E287EFD8F8DB5C1A3A22886C2765FED26B5189153657B7C47D5590F11C6340
067B800669B05A0849BCD2005DFEE6DF95833C9E94328D72F931D69CFBB2BDA81AC83DD660B3B17D
2BA4023491DED324FC4F22510ECA4A5194B1245F4F3FE334DA9C1E6BF83A3FB30897BE54C688D2A7
C5845F425866F25DD0A9852BA6DAAF8437DD80BCC72B3E258A906DE079A2D33EC5C5F6927503BA13
158305DFFD3F86345524394151AA557D6242060F276BB6BB25586F632942ACF5E0883CD3F8393688
F360323A000B82BD89414E9C807994B0234D730BC6D7CD0A2BF75D9F510786E83EE98D4CACF20EFF
86EE9C38B8D52455D8A694B689F0D9A632E7A6AC6675E190A12ADD716D2C6322657B878FA97267C1
BA4631584356768EBBD1F13FD2F37EBDCD1DF96FB943942E8A5188666235B455BE2F770C9759A8F0
```

```
70971CBA49789744FD2F64DC4DC6E003B3F9BEC7617C7EEDF6BACA94D374400499CA6813C90A03DF
E2C537261DA93A1C0F6D8BA93D1EB5FB17255DF28B78737582FD675D056A4C474A71CA8EF0D77BAE
BE5637711AEA3FF2B014700448C3D74E3DF264D773360F45CCC3342987169C9A'

'94741D7F05B0CA50908E6BC14801A28E353551F01769451B1482FAD0043D5C72331246D9AC3344F
0FA2E28FD00E86B38F5E0452F46CA111E92D01B37E966455DF1374883DB8B055C4DF25B42182280F
86D0D825C096018D2949B4BFCEB7BB2C8A5BFA2C79E27F11A7F9B43A50AF928D81FA95CEC86A1142
22B99786072311025672AB04B2593C5AF50100B71D052AE268FBA992BF7868E58EFCD07A24D21117
74A36115C1C527B5192EA955722EAE849EF83817FE8595C96EA2D76FECF6476D89F65A262D94B3F5
E89A5DE8B1A7333EFCDFDED17FE1CCADEBA0D1E7B73E67491B413A862E34A308D5C211787E6ED868
3C6E1DDEB8EE2D281166C03E7A72D7D7BD8B878D07D2216C21B855CCDA76B7B75DD1B2CB876E59F9
1F040D42B97050043499DCFFC65AF803E2F7455C9669DD9896FE1F62227936DF905835A644D31130
A39479DE75B4DC4361E41202D51D50E0E4B4B218AF7F5CAF264DCD060C296E777DF1EED6AE8147E9
B6CA73184C345FBDD89DE4A999C42AB4681D9EA3B86DD75031A33DCDC807F8FB14EE0CE61B16068A
F01CCE7378C9D965943476AD21A469D8B0CAE15BA8FE04971FE1EC61D3AAD3386DF71B33FD0B4F32
4F3DA518F0CC0353182B3D76CF4EF5AB150FB9E74C28234CB3D907AC81CB6D3B99D510B481E1F042
3D6F4987F5517ABBBEEC07F46AECEBA5F15D91AEB0FE91490E91F739D465225C'

'839A01464B473A64A3D1EA24EB363EAAA590F4BD0E4492FEC4E3D4DB5883E4873BBA17595FF4813
4893F16F5C4A43659C46484A268C3303B2DC345E8C98FBBA6D06946F997074AE15680EC9423D6464
585D98804B3541662E183F6540503BEC204749D58E3DB9ECF11C80CD3A38F8D66FFE6CC8A003BDD3
5F547E5039DE9A21F70A8A07B2DD89B68E43B42C2E021A11909817C543F839E6862268E38DCE712B
4D49C39A5035F3D6BA19AE028AE70CCF557720794FEF6442999E740CD6AFE6235F165515FDC24AB6
F578DB2549C8065E008577FCF8B8DD8A3BA679BABBC9A747A4E2DABD91501424E4191097E689A741
EB6644A771CABDBFE6B74ED3ED171DF8DE641C1D42213B9D0F8CAD1E11FF63670F5587F1FB7FF922
76AB48F31751E7A591AF4F0966F3909883EE6015639671BDC3D1378750F66F5DD165912CFF1A54ED
463905404EB7D3412EE2B0F0D9E6B99EC81678ABCD1789BD8F1D72D3DF8754A16DC2106B83B32580
7E27BCBD22A25DAC32F27EACAB6A4CB6CBA4CC90D5302BE5E9827B7AB48BB696B2902975C48B3A4B
A4630B14E0FD8A050B0718C2829371BEC597387172B0B3192EF958BD1F7977EF9A3A6C80D53BC961
315F97B714253B9731A017BE2CA1D43024F75E26BBE989C4D514D01538956FE4B90BE17B3407B55B
D08BA50FA807D0E448B7CAC65EB3FF856772A933F0C5F3E6F41E051015C6F9B8'

'BDA2B72F0BB0265269F198207FB061DA29DE43E30847E7C062A581A7EB53491EA51B51EDD36F991
D15AF89AB53198537988350FD5FDF8E003019BE115840B9BA55C238C3CBC72C0E24E25090A3D6A59
BEA9FED0FAC9EAD40451A95649638FE0BB0F8FFE61AF5B9A8AB84BE84C65EA1E12E9F6650ADB59A8
24E608E80D1FC3AC19F418169B3879CC946165511D5AA280AE644AF360C42F7A3EEDF27E368E4648
0E3353E67F536E02B33505341BAF3941069567B723D7C125C8F066F9A6255436AAFDCAA8C554FDAF
B0A9AAD91F1263DC62EF91A748FFB29F57E325D65A38ECB4F2851923DC6E9B7296064148A9BA2D93
8116266C597D9E1F11A46BE0EF526225BE750F0F3E5B0AEB7DC2140FA3A48B7238D0F5A872000782
CB6F7751443EC6A1B7FA1ED02B9ABCD1C1DE4FC85E9B405C7851913C60F85582B1529276AD475AE5
2BD8115B6E73A53506E7A0244E1C29BCEF4CF20CFDF883392BB3990BE2A11B3213B68EC4A166C77D
724CFAEBDC34C45ED09848A994BCE1FF6A9BB80C7F5CA8FD44D3FDF8DEC8BA6552C234EF8DC52382
D52D2B01BB23404FC453725C7C9269A785FE09C712D4ADE7072B66295CA0C6405D9859E134FBBD37
37F2956DD1D718A9F8242CE95BDB1E49F265EBF19976BC46E29F7DE0EE5C89A43AF2E107588A46E1
B6762E6F8E48B8FC4F4FF93EC60938B8E5C3719022C750C4309FC62ADA4E9028'

'D240216C5C4A70742CAA03AE910E8859C92E5A90A352CB8B45847BAC7793E1F75720D44919E896A
D4581E1FD83986FF235C9834BEECAA1556794BE49033E79D4CCDB4DC67C5200E8B6A3EE891E700B3
48CBF092E4D3FA5E648B620E34E491D7B628A1FE7E2C45586B6577E50788687F0858C10F78F371B2
5C712ED2760C3D605D4ED4F052E8B66FC308D3ADD4A9B86F00CE4257EED085EAE95FBB1E113FCB42
CE12BB6076178A20903C55DA570EF8A25BA7AC8B7E134B8D4E35AB172CA33CC97294A5E7E579B936
1B92B49B63BB1982740015DFEC16882989C917F50D5FDD9166FE1001F3282D3C54A28AC7FD773CCC
0634AF7CDF225F94107C169D2F2BB757EEB55933CCE0FF116D7FFBA992F9A075A2439CCB369D5B5D
E460CADC9F8C81D98E71651AEBFC2A918C551082D85F75675CDC8CCA1D3E486CFFB3B025D27C8D67
C451FDFCF59C3BFA163EB791152390E9488C604B9B8116C329453A98F7A104527BC677411034CC49
686108E569B7595E1DDC85918D90BBCB337855860D6E4718C0679DAB6982D23FCB6648E8561F44BC
F9B052D8B58384523BC592C9B7F824B96AD1A39AEBD2232D6D34DC171E8FBF933900960F207B5559
7759D23E1E794507586114228A2FC100CC200D2B862DF3F26E6D1C9370373FE165C326D8C29FD2F0
B3071AFD5215781BFB589F605263FF065B7A5CA3F6AA9DE3FD8BF5589BDE3526'

'8E7752C52805DD0A723D61F0BBE0122DF576A42B5AFDF9F196A766C9B3BFE296DC16A892FAECEED
```

```
D8256D2B1AE6BFE5437D4A2691803043B59862B30D68E4FF94A0700D735CFE967299724DA9D68020
0C898EED1C785E7B8CEB14F1DCDC73FC625F9678B407603587220C2FDFE0A47E82ADF36C26F94279
7D608BA6B38A3AD1A967315E1F2D665B27D51E350F075531A179DB2EED55547EA61761CD2B3962FC
B347279117D1C7A7574B49FFE0991AF572A2B0C962A8A79800CFD524AAF9E6401C44569600F41F04
422DB891D25B9F714713086BBFD0FB268E66A4FB10C0ABEEB31D0FBFBA20B0E4FFF404051596FC6F
6C8093AD01807FA52041CD33007B205D15D47AF733966411A36F4C7B846D0BE049ADC21B89EA4CE0
FBA414C005E66F36FACF3C43B474D47DAD78AC114D0171C031DFBE4A15FE1A22603CD79B6BB448B6
7A4DEDC97262F7B869C54F385F3682C744ED5AD6C0B6E16793920E6B45A024010896D5FECFA111CC
9F0C34E728B32F2C4D45B8AA69B621AB9AC3D9D79B38BF205E8D0D19FAC44A76B9F5644526E06858
F76B3EE2D74AEB1971D6B6E68B83773399AC32203164564B102B26C370A9FEC673C285AE0D1D3DF2
39D48B6492B89846EBED4618AEC940DC62AF4C3FF0D56FC9FBE23EE3B0A4890BA2665A88E9F40C4B
6A770F9630234ED10A3A7FF3C5BCCBA836F3EDC8B821AB18D4B1D51D9962C328'

'E682E9D8E92A7837823C9B7714D267F9CE290E9FA6CC0A8432D3F7507DAF6CF681246AA4C2323C6
B53BCC6E53B31F49742EE5F4E6F79DC36727E98B06D0300ED21F0CF5F2B51D8304A51D0B498F4BFA
39C0049B8117DAD334D4B2E37676EC42DFE0EED63B3726872CCF9A10223A8A4563BE8AC266E06970
04921DCCEEA5DD80C62567FDEBF2AFDF030192831A6FD871F63D5DADA4B270AA9EC0ACE47E75BD19
018CB809B548D4F2C24831C384DD2B807852F596BD4FE32CAB3A16899D0B100E9F96D06AACB8DA8D
51DB0B0F600F3B614461F5238188B5EDA68EA753B6ACC58569E841BAF92CEE04E6E2626B1FBD01B9
B67D1311B1C3D67427298E2D193F0647EA17D16FD7FD6A40A1BDBB320A1F5FC64B97759AF4EA92AA
EB759B5DD30A726E9B8EAFA372FBD83CBFF0000CA75F219A95D6A3CDE38B8DFA9281609A20EE39B7
3FEBDF6A155359476D073E7153BC918C1191C9BAAF0E0F161384DAD8AFC31A3FC1E9EAFA495E22D1
8C05194EB85298AB0F042E447DD627904B73E6E505712DF010531C88E695F6510C78B443C731D7FD
CD62EB7C4015AB5D530BD09CE5229FA4DC5642AF176C39D60FE070DF635CC5435136C7BB9C4DC83B
0D382B9BB636A6C2B3838542904D53B862585FE6EC8960A9A77783D17B2D90506F5D60998602AE54
30E86025C8864883CECD7CE51B49CC2953A2A41D7EF8027F1A83815BBEF6F6B2'

'F6BD4204243CBA14DAA15A256FBCD138B5D875E28BCC0BA36855E648434CD04F49935C3D074DD5B
A2EB82AB14E82C30991A1159E990D1D36DAF794853A23C499AB6B3DC02A89F014310372813643F78
6BF19D3FA8C463EE50D9FA87107E91C461AD2E5DF2FC99630D2005894CB7698123111FAFC0C5BC9D
1E8E84FCCA5179A6C9AFE3E369222D66854F90D2668A57FDEE00C300AEA4E88F03F05C4D7695B206
DE9F7E1D429E5E6B65DFE05D4C861F4E7844DDB9062C0B6DB46B27AD0368992F54A44829DD11A05A
B97BA8AD854E428B87F20C4E5E4BB1FF3803809A81F2E4C109572006729A5E490E0AA40BA55F4391
C9FB758EFA79B97E6D413BCB02D33A00DA6705BFBADED66CFC21291C494B7C3293810012ECC61415
E609DD97AAFFDEB795DE36026B4602DD546A1AD937F1A6DEACD3393F5530C48A7974E2882CB327AE
600C05A535BDE5D15AC524859582EEE2D62194B73E01643359E7B2625F3EB9FE7137514ED549A319
6FFCBC8072B4F6C18CC67AAFA0ED6029A805EF0987E2F27A3260F849C68F3EF91DAA9E579AA16FDA
698CC18AE8706E28C6D84CB3F593273D763C2969933D8EFA564E8C06C427809E6A5A6F76DE7C8B07
FF4EDDF6CF2B7595066DFB15F5C6F3839DEE642FC86BC1F3AED7ED2E65B665198AA034817DBBBE0F
E30E662B2161276CBD969FDA05AFD6D6A570C1E3CF7E324634441983F257E2BA'

'A9366308475F2D8D0C2D451C4A65A01EE58A0AF19B791D97382EC59A52616C7480B86EB1D0A83E9
3224B0DF73DE1D7EE6D51088F3B20B7937E6C0144E0DACA6324F0C8E5F9D93A8CBA1045E5B509D7D
F98619FDDFDD7892C3082D69008D9D3ED6C9C1367D9DB7C04621D7CDD8A5A2599EE45B87A82F8CE8
D60293E7A71D11700CA9AF117D630C5D8B876A9DCE519BD653114448C68B265813C608435B96CD64
2A420A15FBAB467692931BCA74F1F9D23F5BFDDC5B8651139B5A73F04FEF3DA64B7BD56E49235069
EE5E8A136B921051F1D1C7D5993E6EEEEA2D58583152ADCD87AA89CF5962BC8341EF99CEB3682A2D
0686602CE140ABC2FDF79A778A9D75AFFDBBA00C0BD6A8A8AFF9B5D1F30C8373572C81BD95948901
02F46B5A393ED126C36AEF6A66E231A246FDFCBD3DED198ABC54CF357ABC67AC83680C048932D7C9
02AB7DB16952B3C95DF4E845B46A362FFE1A27CD1388483FFA41AA563933371C0180848F9E3C03AF
C1F00D6ABA29A953327A4E3D9FAD4616C8546C9AF89FB4D08D4256923B736A8F68FEA5A097E0640C
16E0F7F942E6A6F5CBA76BB00D81C606C7FED908789A63F01F9B5FC7B7BE434E85A0A44B2070BE71
AB2BA0132D9D7B32E2D2FE229619F85643E75B4141D355386D1A09F45738455BC21607086C7BBCD4
B73F87DD83E905BCE8FC6C5BF1824E904C4F5C26518B2FEBF8EB06B22437270C'

'92D87BF3F54B0445C05E508E80F9CBC0502F0897D717CA232004362F394A023BFBFE3322C1D331A
FC6454FC756FB48768693FD5C46DDB40DCBF14C726C24ED67D8F3EB613BA80B0E39CF0747DF62D25
8613640D881E085C377DE1C3D149C8359407C2C6ABC0D2718A2D42439A8E7B38CD7DCED72AE750B2
BE88D0069FBE94BD69A9A4B4AD42FEC5E651A31F86B90DC2FEBAA6FA6E5F6368B620C1750278DF39
3F7C5035D47897FC05FBC419A61330135F24365F13D653D77CA2930DBB05A3815FE83F75BB1BD8B2
DE12A2FAADCD1ED62329C55B87FB32CC8F3B42D888981B4192480D1F57CEB0C55897BDA6B9C0ACE1
```

```
E7E4595E30C736830624320844FCF4574C47B07725B25EC2E28F4C50B744B3860B361DDDD22D949
AA94EBA4F97606FCAD91394B6FC0E634BD15E099E697403B2AE84CDF5DBDF36D91FB82C0BC12B984
FEE83CA9E97C194CADF8382CECAAF49EB3BD446F660F94C188C074CC312E186BEE0F6585535B050C
226659A94B4C4974DA32CDFF30DBEB4DEA588C6F490F7432DA5FA2408BBC931EAF60EADD7B891A61
C157147B8DDE7A45F909BD20D5B12009783DE410940245FE4E91ACCF72942E486AE773CD66591217
3EA29875A1722F8658C414CD08CBFDFE1DD356E167A9D7B20BF7441562EE816435A78BAE7E5A5EB4
DA6AAAC36F594C93E2851D76B6A18B0B03B30CD38B97E38109C494C557643D58'];
```

```
function [bmemcodE1B,bmemcodE1C]= memseqL1(mat1,mat2)

% Transforms into binary mat1 and mat2 which are in hexadecimal (signal E1)

sa=size(mat1);

binmat1=zeros(sa(1),sa(2)*4);
binmat2=binmat1;

for i=1:sa(1)
    binmat1(i,:)=htob(mat1(i,:));
    binmat2(i,:)=htob(mat2(i,:));
end

bmemcodE1B=binmat1;
bmemcodE1C=binmat2;
```

```
function [binmat]=transf(vec,i1,i2,i3,type)

% Transforms into binary either an octal or hexadecimal vector element by
% element

bin=zeros(i3,i2);
fil=[];
j=1;

if type=='o'
    for i=1:i1
        aux=otob(vec(i));
        fil=[fil aux];
        if length(fil)==i2
            bin(j,:)=fil;
            j=j+1;
            fil=[];
        end
    end
else
    for i=1:i1
        aux=htob(vec(i));
        fil=[fil aux];
        if length(fil)==i2
            bin(j,:)=fil;
            j=j+1;
            fil=[];
        end
    end
end
binmat=bin;
```

```
function bin=htob(hex)

%Transforms every element of an hexadecimal number into binary

L=length(hex);
b=zeros(1,4*L);
j=1;

for i=1:L

    switch hex(i)

        case '0'
            b(j)=0; b(j+1)=0; b(j+2)=0; b(j+3)=0;
            j=j+4;
        case '1'
            b(j)=0; b(j+1)=0; b(j+2)=0; b(j+3)=1;
            j=j+4;
        case '2'
            b(j)=0; b(j+1)=0; b(j+2)=1; b(j+3)=0;
            j=j+4;
        case '3'
            b(j)=0; b(j+1)=0; b(j+2)=1; b(j+3)=1;
            j=j+4;
        case '4'
            b(j)=0; b(j+1)=1; b(j+2)=0; b(j+3)=0;
            j=j+4;
        case '5'
            b(j)=0; b(j+1)=1; b(j+2)=0; b(j+3)=1;
            j=j+4;
        case '6'
            b(j)=0; b(j+1)=1; b(j+2)=1; b(j+3)=0;
            j=j+4;
        case '7'
            b(j)=0; b(j+1)=1; b(j+2)=1; b(j+3)=1;
            j=j+4;
        case '8'
            b(j)=1; b(j+1)=0; b(j+2)=0; b(j+3)=0;
            j=j+4;
        case '9'
            b(j)=1; b(j+1)=0; b(j+2)=0; b(j+3)=1;
            j=j+4;
        case {'a','A'}
            b(j)=1; b(j+1)=0; b(j+2)=1; b(j+3)=0;
            j=j+4;
        case {'b','B'}
            b(j)=1; b(j+1)=0; b(j+2)=1; b(j+3)=1;
            j=j+4;
        case {'c','C'}
            b(j)=1; b(j+1)=1; b(j+2)=0; b(j+3)=0;
            j=j+4;
        case {'d','D'}
            b(j)=1; b(j+1)=1; b(j+2)=0; b(j+3)=1;
            j=j+4;
        case {'e','E'}
            b(j)=1; b(j+1)=1; b(j+2)=1; b(j+3)=0;
            j=j+4;
```

```
        case {'f','F'}
            b(j)=1; b(j+1)=1; b(j+2)=1; b(j+3)=1;
            j=j+4;
    end
end


bin=b;
```

```
function bin=otob(oct)

%Transforms every element of an octal number into binary

L=length(oct);
b=zeros(1,3*L);
j=1;

for i=1:L

    switch oct(i)

        case '0'
            b(j)=0; b(j+1)=0; b(j+2)=0;
            j=j+3;
        case '1'
            b(j)=0; b(j+1)=0; b(j+2)=1;
            j=j+3;
        case '2'
            b(j)=0; b(j+1)=1; b(j+2)=0;
            j=j+3;
        case '3'
            b(j)=0; b(j+1)=1; b(j+2)=1;
            j=j+3;
        case '4'
            b(j)=1; b(j+1)=0; b(j+2)=0;
            j=j+3;
        case '5'
            b(j)=1; b(j+1)=0; b(j+2)=1;
            j=j+3;
        case '6'
            b(j)=1; b(j+1)=1; b(j+2)=0;
            j=j+3;
        case '7'
            b(j)=1; b(j+1)=1; b(j+2)=1;
            j=j+3;
    end
end

bin=b;
```

---

3. ACF and PSD of BOC

```
function comparative
```

```
%Compares PSD and ACF of different BOC(fs,fc) modulations.

fo=1.023e6;
fsamp=120*fo;

B=5;
Rb=250;
D=B/Rb;

N=fsamp*D;
k=-(N-1)/2:(N-1)/2;
df=fsamp/N;
f=k*df;

%***********************************Stadistics of a BOCsin(1,1)

n1s=sin((pi*f)/(2*fo));
n2s=sin((pi*f)/(fo));
q1s=pi*f;
q2s=cos((pi*f)/(2*fo));

ns=n1s.*n2s;
qs=q1s.*q2s;

fracts=ns./qs;

Gbocs=fo*(fracts).^2;
Rggs=ifftshift(ifft(ifftshift(Gbocs)));
Rggs=N*abs(Rggs);
Rggs=Rggs/max(Rggs);

%***********************************Stadistics of a BOCsin(6,1)

n1sf=sin((pi*f)/(2*6*fo));
n2sf=sin((pi*f)/(fo));
q1sf=pi*f;
q2sf=cos((pi*f)/(2*6*fo));

nsf=n1sf.*n2sf;
qsf=q1sf.*q2sf;

fractsf=nsf./qsf;

Gbocsf=fo*((fractsf).^2);
Rggsf=ifftshift(ifft(ifftshift(Gbocsf)));
Rggsf=N*abs(Rggsf);
Rggsf=Rggsf/max(Rggsf);

%***********************************Stadistics of a BOCsin(10,5)

n10s=sin((pi*f)/(2*10*fo));
n20s=sin((pi*f)/(5*fo));
q10s=pi*f;
q20s=cos((pi*f)/(2*10*fo));

ns0=n10s.*n20s;
qs0=q10s.*q20s;

fracts0=ns0./qs0;

Gbocs0=5*fo*((fracts0).^2);
```

```
Rggs0=ifftshift(ifft(ifftshift(Gbocs0)));
Rggs0=N*abs(Rggs0);
Rggs0=Rggs0/max(Rggs0);

%***********************************Stadistics of a BOCsin(15,2.5)

n10z=sin((pi*f)/(2*15*fo));
n20z=sin((pi*f)/(2.5*fo));
q10z=pi*f;
q20z=cos((pi*f)/(2*15*fo));

ns0z=n10z.*n20z;
qs0z=q10z.*q20z;

fracts0z=ns0z./qs0z;

Gbocs0z=2.5*fo*((fracts0z).^2);
Rggs0z=ifftshift(ifft(ifftshift(Gbocs0z)));
Rggs0z=N*abs(Rggs0z);
Rggs0z=Rggs0z/max(Rggs0z);

%***********************************Stadistics of a BOCsin(15,10)

n10d=sin((pi*f)/(2*15*fo));
n20d=sin((pi*f)/(10*fo));
q10d=pi*f;
q20d=cos((pi*f)/(2*15*fo));


ns0d=n10d.*n20d;
qs0d=q10d.*q20d;

fracts0d=ns0d./qs0d;

Gbocs0d=10*fo*((fracts0d).^2);
Rggs0d=ifftshift(ifft(ifftshift(Gbocs0d)));
Rggs0d=N*abs(Rggs0d);
Rggs0d=Rggs0z/max(Rggs0d);


%***********************************Stadistics of a BOCcos(1,1)

%the only different thing is the first factor in the numerator

n1=(sin((pi*f)/(4*fo))).^2;

n=2*n1.*n2s;
fract=n./qs;

Gbocc=fo*(fract).^2;
Rggc=ifftshift(ifft(ifftshift(Gbocc)));
Rggc=N*abs(Rggc);
Rggc=Rggc/max(Rggc);

%***********************************Stadistics of a BOCcos(6,1)

n1i=(sin((pi*f)/(4*6*fo))).^2;

ni=2*n1i.*n2sf;
fracti=ni./qsf;
```

```
Gbocci=fo*(fracti).^2;
Rggci=ifftshift(ifft(ifftshift(Gbocci)));
Rggci=N*abs(Rggci);
Rggci=Rggc/max(Rggci);


%***********************************Stadistics of a BOCcos(10,5)

%the only different thing is the first factor in the numerator

n10=(sin((pi*f)/(4*10*fo))).^2;

n0=2*n10.*n20s;
fract0=n0./qs0;

Gbocc0=5*fo*((fract0).^2);
Rggc0=ifftshift(ifft(ifftshift(Gbocc0)));
Rggc0=N*abs(Rggc0);
Rggc0=Rggc0/max(Rggc0);

%***********************************Stadistics of a BOCcos(15,2.5)

n10z=(sin((pi*f)/(4*15*fo))).^2;

n0z=2*n10z.*n20z;
fract0z=n0z./qs0z;

Gbocc0z=2.5*fo*((fract0z).^2);
Rggc0z=ifftshift(ifft(ifftshift(Gbocc0z)));
Rggc0z=N*abs(Rggc0z);
Rggc0z=Rggc0z/max(Rggc0z);

%***********************************Stadistics of a BOCcos(15,10)

n10d=(sin((pi*f)/(4*15*fo))).^2;

n0d=2*n10d.*n20d;
fract0dd=n0d./qs0d;

Gbocc0dd=10*fo*((fract0dd).^2);
Rggc0dd=ifftshift(ifft(ifftshift(Gbocc0dd)));
Rggc0dd=N*abs(Rggc0dd);
Rggc0dd=Rggc0dd/max(Rggc0dd);

%_____________________________________PLOTS_____________________________________

figure(1);
hold on
plot(f,Gbocs,'color','r');
plot(f,Gbocsf);
plot(f,Gbocs0,'color','g');
plot(f,Gbocs0z,'color','k');
xlim([-40e6 40e6]);
legend('BOC(1,1)','BOC(6,1)','BOC(10,5)','BOC(15,2.5)',4,'Location','NorthEast')
;
xlabel('Hertzs');
title('PSD Sine');
hold off

figure(2);
hold on
```

```
plot(f,Gbocc,'color','y');
plot(f,Gbocci,'color','c');
plot(f,Gbocc0,'color','m');
plot(f,Gbocc0z,'color','r');
xlim([-40e6 40e6]);
legend('BOCcos(1,1)','BOCcos(6,1)','BOCcos(10,5)','BOCcos(15,2.5)',4,'Location',
'NorthEast');
xlabel('Hertzs');
title('PSD Cosine');
hold off

figure(3);
hold on
plot(f,Gbocs0d,'Marker','d');
plot(f,Gbocc0dd,'color','r');
xlim([15.3445e6 15.346e6]);
legend('BOC(15,10)','BOCcos(15,10)',2,'Location','northeast');
hold off

figure(4);
hold on
plot(f,Gbocs,'color','c');
plot(f,Gbocsf);
plot(f,Gbocs0,'color','g');
plot(f,Gbocs0z,'color','k');
plot(f,Gbocc,'color','y');
plot(f,Gbocci,'color','r');
plot(f,Gbocc0,'color','m');
plot(f,Gbocc0z,'color','r');
legend('BOC(1,1)','BOC(6,1)','BOC(10,5)','BOC(15,2.5)','BOCcos(1,1)','BOCcos(6,1
)','BOCcos(10,5)','BOCcos(15,2.5)',8,'Location','NorthEast');
xlim([-40e6 40e6]);
xlabel('Hertzs');
title('PSD Sine & Cosine');
hold off
```

```
function dumacf

fsamp=300;
t=0:1/fsamp:299/fsamp;

%____ BOC(1,1)_____________________________________________________

n=2;                    %Number of half-periods in a chip time.
sc=[1 -1];              %Sub-carrier chips

xsc=expan(sc,fsamp,n);

cxsc=zeros(length(xsc),1);

for i=1:length(xsc)

    if ((75<i)&&(i<151)) || ((226<i)&&(i<=300))
        cxsc(i)=-xsc(i);
    else
        cxsc(i)=xsc(i);
    end

end
```

```
[Rsc,lag]=xcorr(xsc,'coeff');
[Rscc,lagc]=xcorr(cxsc,'coeff');

%[Gsc,fsc]=shcenteredFFT(Rsc,fsamp);
%[Gscc,fscc]=shcenteredFFT(Rscc,fsamp);

%figure(21); hold on
%plot(fsc,abs(Gsc));
%plot(fscc,abs(Gscc),'color','r');
%xlim([-50 50]);
%hold off

%____ BOC(10,5)_____________________________________________

n1=4;
sc1=[1 -1 1 -1];

xsc1=expan(sc1,fsamp,n1);

cxsc1=zeros(length(xsc1)*0.5,1);

for i=1:(length(xsc1)*0.5)

    if ((37<i)&&(i<76)) || ((114<i)&&(i<=150))
        cxsc1(i)=-xsc1(i);
    else
        cxsc1(i)=xsc1(i);
    end

end

cxsc1=[cxsc1(1:150) ; cxsc1(1:150)];

[Rsc1,lag1]=xcorr(xsc1,'coeff');
[Rscc1,lagc1]=xcorr(cxsc1,'coeff');

%[Gsc1,fsc1]=shcenteredFFT(Rsc1,fsamp);
%[Gscc1,fscc1]=shcenteredFFT(Rscc1,fsamp);

%figure(22); hold on
%plot(fsc1,abs(Gsc1));
%plot(fscc1,abs(Gscc1),'color','r');
%xlim([-50 50]);
%hold off

%____ BOC(15,2.5)_____________________________________________

n2=12;
sc2=[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1];

xsc2=expan(sc2,fsamp,n2);

cxsc2=zeros(length(xsc2)/6,1);

for i=1:(length(xsc2)/6)

    if ((12<i)&&(i<26)) || ((38<i)&&(i<=50))
        cxsc2(i)=-xsc2(i);
    else
        cxsc2(i)=xsc2(i);
```

```
    end

end

cxsc2=[cxsc2 ; cxsc2 ; cxsc2 ; cxsc2 ; cxsc2 ; cxsc2];

[Rsc2,lag2]=xcorr(xsc2,'coeff');
[Rscc2,lagc2]=xcorr(cxsc2,'coeff');

%[Gsc2,fsc2]=shcenteredFFT(Rsc2,fsamp);
%[Gscc2,fscc2]=shcenteredFFT(Rscc2,fsamp);

%figure(23); hold on
%plot(fsc2,abs(Gsc2));
%plot(fscc2,abs(Gscc2),'color','r');
%xlim([-50 50]);
%hold off

%____ BOC(6,1)_____________________________________________________
% Parameter n is the same than BOC(15,2.5)

Rsc3=Rsc2; lag3=lag2;
Rscc3=Rscc2; lagc3=lagc2;




%____ BOC(15,10)___________________________________________________

n4=3;
sc4=[1 -1 1];

xsc4=expan(sc4,fsamp,n4);

cxsc4=zeros(length(xsc4),1);

for i=1:(length(xsc4))

    if ((50<i)&&(i<101)) || ((151<i)&&(i<201) || ((251<i)&&(i<=300)))
        cxsc4(i)=-xsc4(i);
    else
        cxsc4(i)=xsc4(i);
    end
end

figure(3);hold on
plot(t,xsc4);
plot(t,cxsc4,'color','r');
hold off


[Rsc4,lag4]=xcorr(xsc4,'coeff');
[Rscc4,lagc4]=xcorr(cxsc4,'coeff');

%[Gsc4,fsc4]=shcenteredFFT(Rsc4,fsamp);
%[Gscc4,fscc4]=shcenteredFFT(Rscc4,fsamp);

%figure(24); hold on
%plot(fsc4,abs(Gsc4));
%plot(fscc4,abs(Gscc4),'color','r');
%xlim([-50 50]);
```

```
%hold off

%_____________________________PLOTS_____________________________

figure(1);hold on
grid on
plot(lag,Rsc,'LineWidth',2);
plot(lagc,Rscc,'color','m');
plot(lag1,Rsc1,'color','r','LineWidth',2);
plot(lagc1,Rscc1,'color','g');
plot(lag2,Rsc2,'color','k','Marker','*');
plot(lagc2,Rscc2,'color','y','LineWidth',2);
plot(lag3,Rsc3,'color','c');
plot(lag4,Rsc4,'color','g','Marker','h','LineWidth',1);
plot(lagc4,Rscc4,'color','m','Marker','^');
legend('BOC(1,1)','BOCcos(1,1)','BOC(10,5)','BOCcos(10,5)','BOC(15,2.5)','BOCcos
(15,2.5)','BOC(6,1)','BOC(15,10)','BOCcos(15,10)',9,'Location','NorthEast');
hold off

figure(2);
subplot(2,1,1),hold on
grid on
plot(lag,Rsc,'LineWidth',2);
plot(lag1,Rsc1,'color','r','LineWidth',2);
plot(lag2,Rsc2,'color','k','Marker','d');
plot(lag3,Rsc3,'color','c');
plot(lag4,Rsc4,'color','y','LineWidth',2);
title('Sinusoidal subcarrier');
legend('BOC(1,1)','BOC(10,5)','BOC(15,2.5)','BOC(6,1)','BOC(15,10)',5,'Location'
,'Best');
hold off

subplot(2,1,2),hold on
grid on
plot(lagc,Rscc,'LineWidth',2);
plot(lagc1,Rscc1,'color','r','Linewidth',2);
plot(lagc2,Rscc2,'color','k','Marker','d');
plot(lagc3,Rscc3,'color','c');
plot(lagc4,Rscc4,'color','y','LineWidth',2);
title('Cosine subcarrier');
legend('BOC(1,1)','BOC(10,5)','BOC(15,2.5)','BOC(6,1)','BOC(15,10)',5,'Location'
,'Best');
hold off

%____ MASTER PLOT_____________________________________________

lags=[lag' lag1' lag2' lag3' lag4', lagc' lagc1' lagc2' lagc3' lagc4'];
acfs=[Rsc Rsc1 Rsc2 Rsc3 Rsc4, Rscc Rscc1 Rscc2 Rscc3 Rscc4];
lags=lags';
acfs=acfs';

for i=1:5
    figure(i+2);
    hold on
    grid on
    plot(lags(i,:),acfs(i,:));
    plot(lags(i+5,:),acfs(i+5,:),'color','r');
    legend('sine subcarrier','cosine subcarrier',2,'Location','NorthEast');
    xlabel('Samples');
    hold off
    switch i
```

```
        case {1}
            title('BOC(1,1)');
        case {2}
            title('BOC(10,5)');
        case {3}
            title('BOC(15,2.5)');
        case {4}
            title('BOC(6,1)');
        case {5}
            title('BOC(15,10)');
    end
end
```

```
function subcarrier(k)

%Shows the PSD of a square sub-carrier

fsamp=20e6;
t=-10e-3:1/fsamp:(10e-3)-(1/fsamp);
f=1e3;
T=1/f;

if k=='s'
    sc=sign(sin(2*pi*f*t));
    scs=sign(sin(-2*pi*f*t));
else
    sc=sign(cos(2*pi*f*t));
    scs=sign(cos(-2*pi*f*t));
end

rsc=xcorr(sc,'coeff');

[SC,f1]=shcenteredFFT(sc,fsamp);
[sSC,f3]=shcenteredFFT(scs,fsamp);
[Ssc,f2]=shcenteredFFT(rsc,fsamp);

X=sin(T*0.5*f1*pi)./(T*0.5*f1*pi);

ReSC=real(SC);
ImSC=imag(SC);

%-----------------------------------PLOTS-----------------------------------------

figure(1)
grid on
plot(t,sc,'linewidth',3); axis([-2e-3 2e-3 -1.15 1.15]); title('sub-carrier');
xlabel('sec');
grid off

figure(3);
plot(f1,abs(SC).^2,'linewidth',3);title('|SC(f)|Â²');
xlim([-11000 11000]);
xlabel('hertzs');
hold on
plot(f1,abs(X).^2,'color','red','linewidth',2);xlim([-1e4 1e4]);
hold off
```

```
function BOC(sct,m)

% Implements a signal BOC in time and frequency and shows all the steps and
% why the spectrum is as it is. The arguments are the subcarrier, the
% chiprate and type of subcarrier -sine or cosine-.
% It will also be shown how affect to the plots the statistitcs of bits,
% giving them mean equal to zero or different to zero.

fo=1.023e6;
fsamp=100*fo;
fs=10*fo;                               %subcarrier
fc=5*fo;                                %chiprate
Rb=1000;                                %symbol rate
D=10/Rb;                                %Duration
t=0:1/fsamp:D-(1/fsamp);                %timeline

if m==0
 B=[1 1 -1 -1 1 1 -1 -1 1 1];           %data stream mean=0
else
 B=[1 -1 -1 -1 1 -1 -1 -1 -1 1];        %data stream mean~=0
end

B=expan(B,fsamp,Rb);

[CA,CC]=cdmaE6;                         %cdma codification
CA=expan(CA,fsamp,fc);
CA=fulltime(CA',10);

if sct=='s'
    sc=sin(2*pi*fs*t);
else
    sc=cos(2*pi*fs*t);
end

sc=sign(sc);                            %subcarrier

signal=B'.*CA.*sc;


%_____________________Transforms |X(f)|^2_____________________________

[tfB,f1]=shcenteredFFT(B,fsamp);
[tfCA,f2]=shcenteredFFT(CA,fsamp);
[tfsc,f3]=shcenteredFFT(sc,fsamp);
[tfs,f4]=shcenteredFFT(signal,fsamp);


figure(1)
subplot(2,2,1), plot(f1,abs(tfB).^2);
xlim([-1100 1100]);
xlabel('hertzs');
title('PSD data');

subplot(2,2,2), plot(f2,abs(tfCA).^2);
xlim([-6*fc 6*fc]);
xlabel('hertzs');
title('PSD cdma');
```

```
subplot(2,2,3), plot(f3,abs(tfsc).^2);
xlabel('hertzs');
title('PSD subcarrier');

subplot(2,2,4), plot(f4,abs(tfs).^2);
xlabel('hertzs');
title('PSD signal');

%_____________________Transform S(f)_____________________________

B=xcorr(B,'coeff');
CA=xcorr(CA,'coeff');
sc=xcorr(sc,'coeff');
signal=xcorr(signal,'coeff');

[tfB,f1]=shcenteredFFT(B,fsamp);
[tfCA,f2]=shcenteredFFT(CA,fsamp);
[tfsc,f3]=shcenteredFFT(sc,fsamp);
[tfs,f4]=shcenteredFFT(signal,fsamp);

figure(2)
subplot(2,2,1), plot(f1,abs(tfB).^2);
xlim([-1100 1100]);
xlabel('hertzs');
title('PSD data');

subplot(2,2,2), plot(f2,abs(tfCA).^2);
xlim([-6*fc 6*fc]);
xlabel('hertzs');
title('PSD cdma');

subplot(2,2,3), plot(f3,abs(tfsc).^2,'linewidth',3);
xlabel('hertzs');
title('PSD subcarrier');

subplot(2,2,4), plot(f4,abs(tfs).^2);
xlabel('hertzs');
title('PSD signal');
```

```
function impactstd

% simulates the spectrum of a BPSK signal when the data are with mean equal
% to zero and when it is not equal to zero. It will be appreciated how
% while the spectrum with mean zero will be two deltas, the spectrum with
% mean different to zero will have more harmonics along all the band.

fsamp=1000;
sr=50;                              %20 samples/bit
symb=100;
D=100/50;

t=0:1/fsamp:D-(1/fsamp);

b1=sign(randn(1,symb));

b2=[];
```

```
for i=1:symb
    if mod(i,2)==0
        b2=[b2 1];
    else
        b2=[b2 -1];
    end
end

b1=expan(b1,fsamp,sr);
b2=expan(b2,fsamp,sr);

%_____________________________mean!=0_____________________________
me1=mean(b1);
mes1=me1.^2;
display(mes1);
c1=xcorr(b1,'coeff');
cv1=xcov(b1,'coeff');

m1=me1*ones(1,length(b1));
mea1=mes1*ones(1,length(cv1));
R1=cv1+mea1';
d1=c1-cv1;
dr1=c1-R1;
S1=fftshift(fft(fftshift(c1)));

[Ss1,f1]=shcenteredFFT(c1,fsamp);

%_____________________________mean=0_____________________________
me2=mean(b2);
mes2=me2.^2;
display(mes2);
c2=xcorr(b2,'coef');
cv2=xcov(b2,'coef');

m2=me2*ones(1,symb);
mea2=mes2*ones(1,length(cv2));
R2=cv2+mea2';
S2=fft(fftshift(c2));
[Ss2,f2]=shcenteredfft(c2,fsamp);


subplot(2,1,1),plot(f1,abs(Ss1));
xlim([-40 40]);
title('PSD with mean!=0');
subplot(2,1,2),plot(f2,abs(Ss2));
xlim([-40 40]);
title('PSD with mean=0');
```