ESA INTERNATIONAL SUMMER SCHOOL ON GNSS 2013
JRC SUMMERSCHOOL GNSS

Lab on GNSS Signal Processing
Part I

Daniele Borio

European Commission Joint Research Centre

Davos, Switzerland, July 15-25, 2013

INTRODUCTION

Goal of the lab:

provide the students with hands-on experience of the various signal processing stages of a GNSS receiver

✓ Acquisition

✓ Tracking

Software Defined Radio (SDR) technology: signal processing operations are performed in software

Given the limited time available, some aspects of acquisition and tracking have been significantly simplified and the lab does not aim at enabling the students to write a complete GNSS software receiver

OUTLINE

Getting started: Matlab and Octave

Part I: the Acquisition process

Part II: Signal Tracking – Principles of tracking loops

Part III: Signal Tracking – Tracking accessories (Bit synchronization and C/N0 estimation)

NOTES

The lab is a compressed version of the material presented during the course

ENGO 638 2012, “GNSS Receiver Design”, University of Calgary
http://www.danieleborio.altervista.org/engo638/engo638.htm

For this lab:

Instructor: Daniele Borio

Assistant: Michele Bavaro (http://michelebavaro.blogspot.it/)

Useful Links:

- GNSS-SDR (http://gnss-sdr.org/)
- Fastgps (http://www.gnssapplications.org/chapter5.html)
- A Software-Defined GPS and Galileo Receiver (http://ccar.colorado.edu/gnss/)

LAB MATERIAL

For the lab a USB key with the following material will be distributed:

- The dataset “Gps+GalCplx8bit10MHz0if.bin”
- The “octave” folder containing installation files for the **Octave software**
- The “acquisition” folder with the code for the first part of the lab
- The tracking folder containing the code for the second and third parts of the lab

MATLAB AND ITS CLONES

The lab was originally conceived to run under the Matlab environment

Matlab is proprietary software (usually adopted by Universities for engineering courses)

Luckily there are some freeware alternatives:

GNU Octave

http://www.gnu.org/software/octave/

FreeMat

http://freemat.sourceforge.net/

Code adapted and tested under Octave

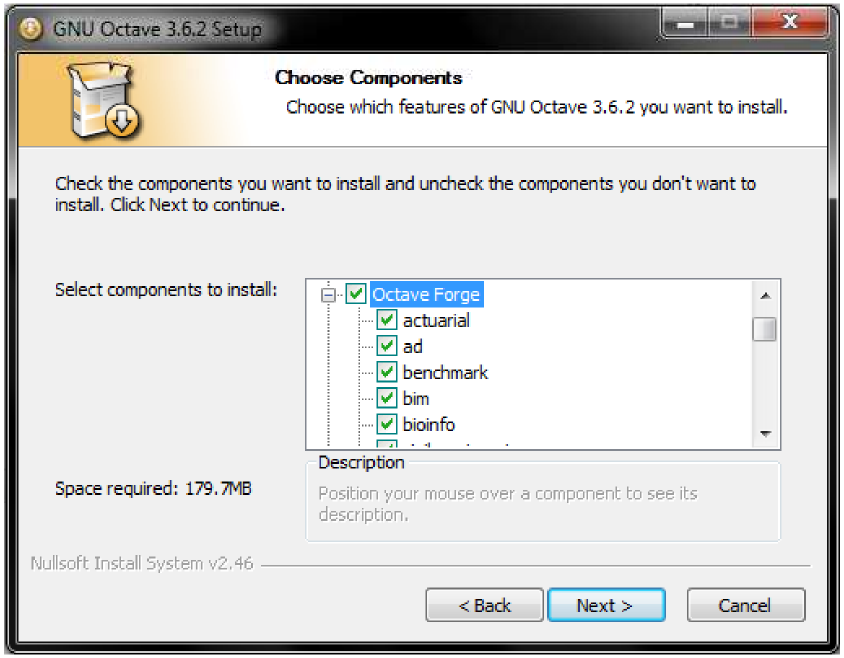
INSTALLING OCTAVE

In the "octave" directory of the USB key:

1) Install the VS2010 redistributable package (run the vcredist_x86.exe executable)

2) Run the octave installer: octave-3.6.2-vs2010-setup.exe

3) When installing octave, be sure to select the additional packages indicated in the next slide



A copy of Notepad++ is also provided and it can be used to edit the scripts

ADDITIONAL PACKAGES

- communications
- control
- general
- image
- miscellaneous
- missing-functions
- optim
- plot
- signal
- specfun
- struct

When Octave starts, type the following command to load all the packages:

```
pkg load all
```

GNSS DATA

"**Raw**" GNSS samples provided in a binary file:

Complex Sampling

Sampling frequency

Gps+GalCplx8bit10MHz0if.bin

GPS L1 C/A and Galileo E1b/c Signals (Spirent Simulator)

Data format: integers on 8 bits

Intermediate frequency (baseband samples)

The file contains the signals at the output of the Analog-to-Digital (A/D) of the receiver front-end (see lecture from Dr. Hegarty)

The samples need to be correctly loaded respecting the file format

DATA LOADING

```
fid = fopen ('../Gps+GalCplx8bit10MHz0if.bin','r');

if( IsComplex ),
    if( Is16Bits ),
        [data, cnt_data] = fread(fid, 2 * secondOfData * fs, 'int16');
    else
        [data, cnt_data] = fread(fid, 2 * secondOfData * fs, 'int8');
    end
    data = data(1:2:end) + 1i * data(2:2:end);
else
    if( Is16Bits ),
        [data, cnt_data] = fread(fid, secondOfData * fs, 'int16');
    else
        [data, cnt_data] = fread(fid, secondOfData * fs, 'int8');
    end
end
```

Open the input file in read mode

If the samples are stored in complex format, they need to be de-interleaved*

I	Q	I	Q	....	I	Q	I	Q

*I/Q complex data are usually stored in an interleaved fashion

GNSS samples stored in the vector ‘data’

DATA INSPECTION

Good practice:

inspect the data to verify that everything is fine

Histogram: evaluate how the samples are distributed (in a statistical sense)

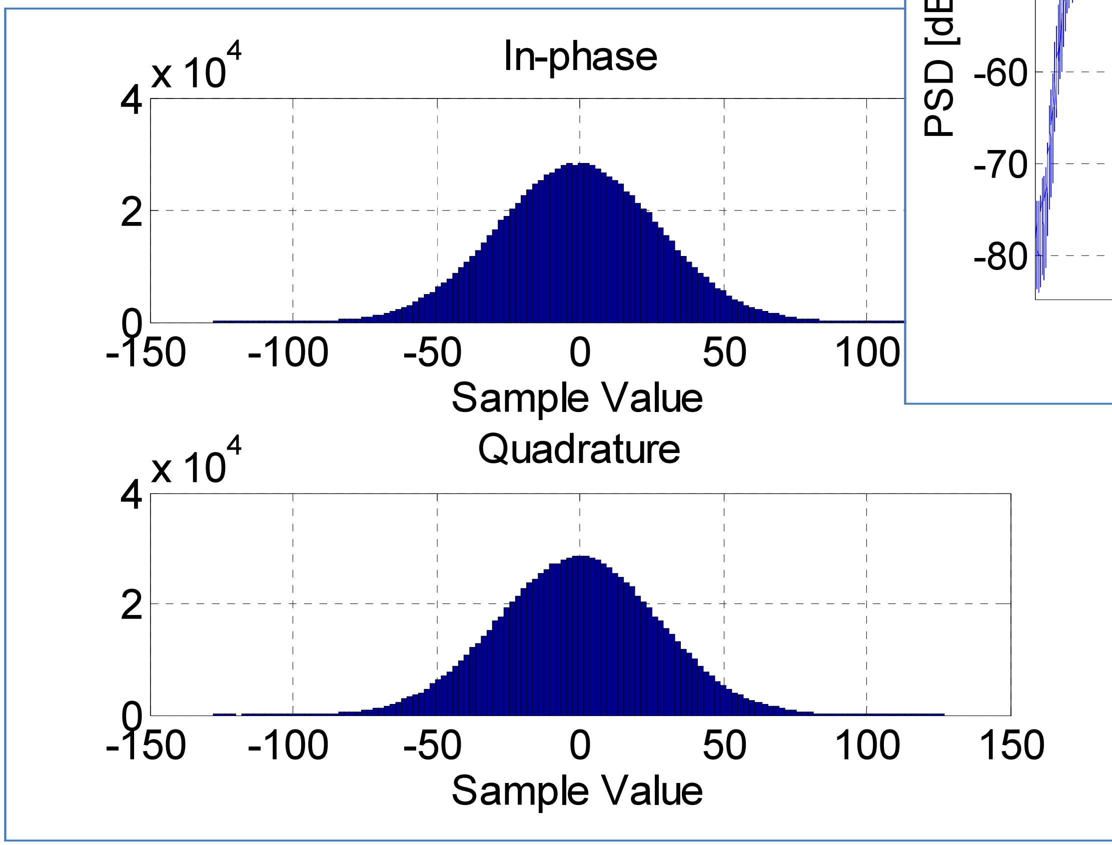
Number of bins

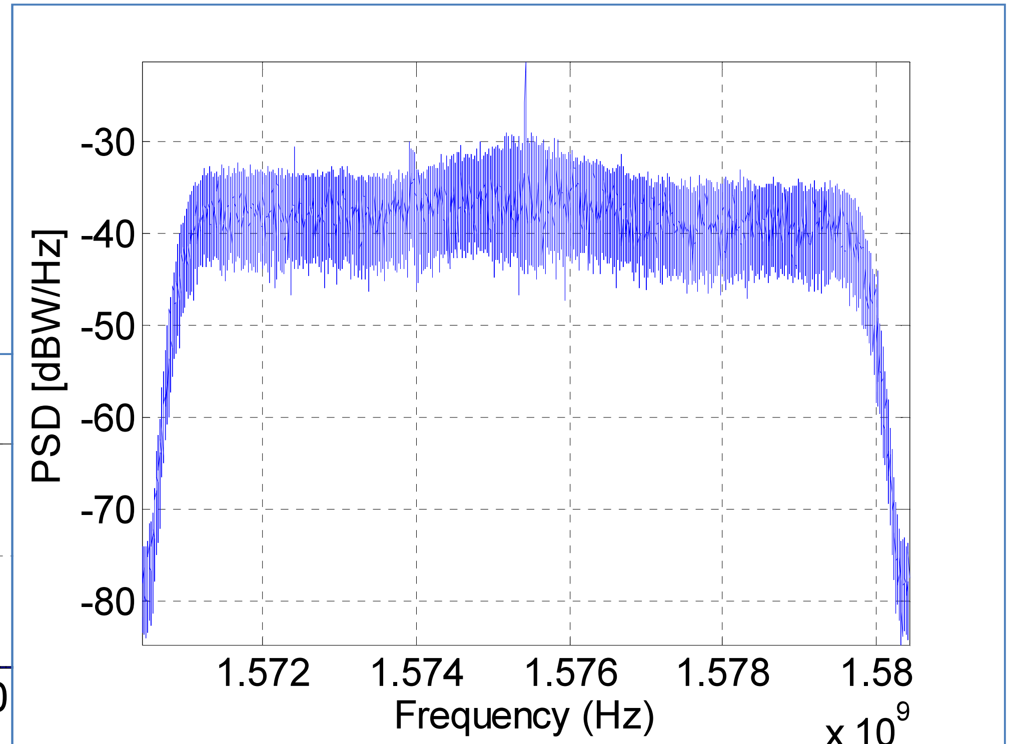
```
hist( real(data), 128 )          hist( imag(data), 128 )
```

Power Spectral Density (PSD): how the signal power is distributed as a function of frequency

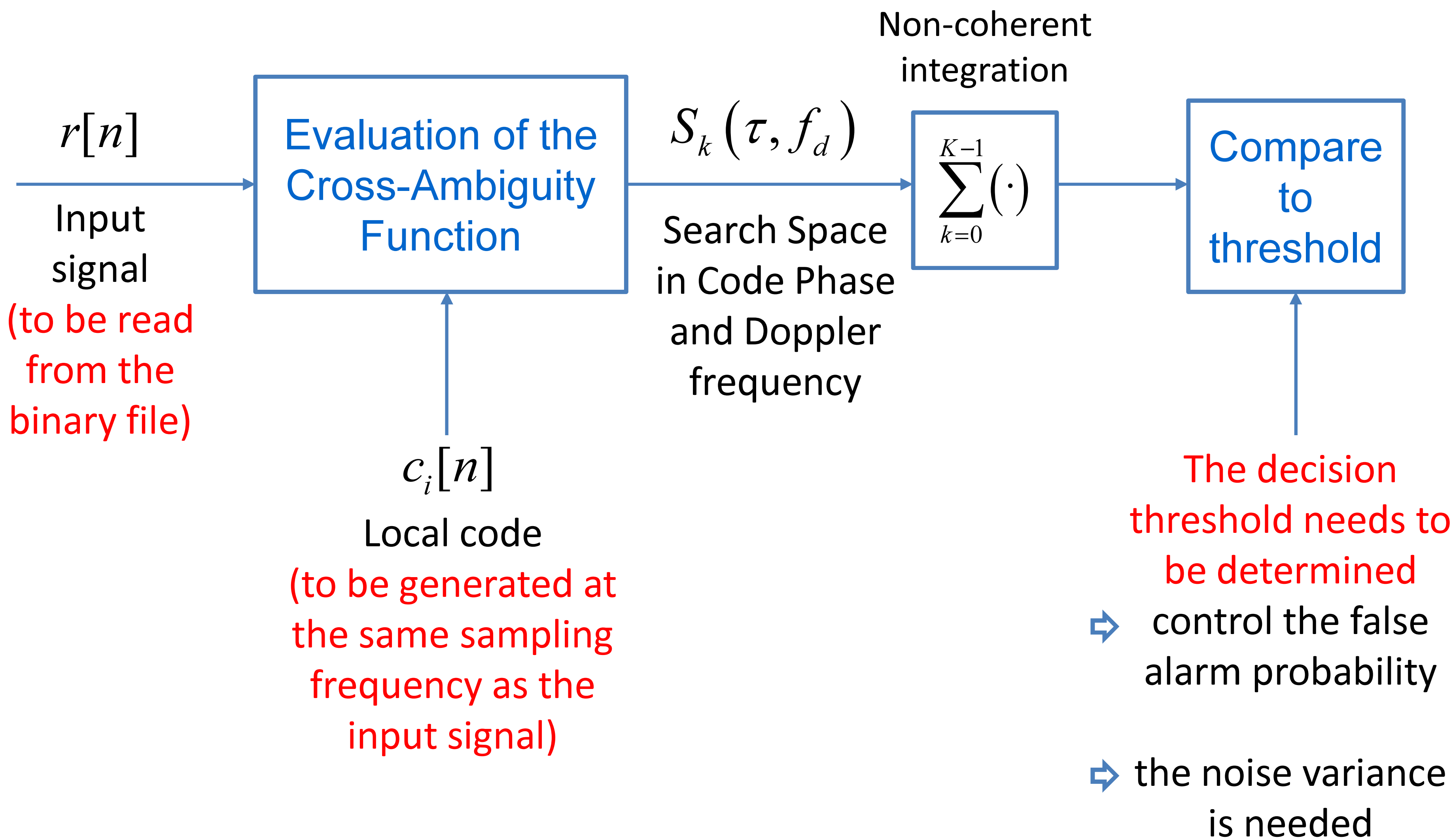
```
[pw f] = pwelch(data, [], [], [], fs );
```

HISTOGRAM AND PSD





ACQUISITION: FUNCTIONAL BLOCKS



LOCAL CODE GENERATION (I/II)

GPS L1 C/A signal: **Gold Codes**

generated by the function

vector with the local code
(1023 elements)

function code = GpsCode(sv, nchips)

satellite code id
(each satellite has its
own code)

number of
chips (1023)

Galileo E1c signals: **Memory Codes**

stored in the file

GalCodeE1c.mat

Need to be loaded

load GalCodeE1c;

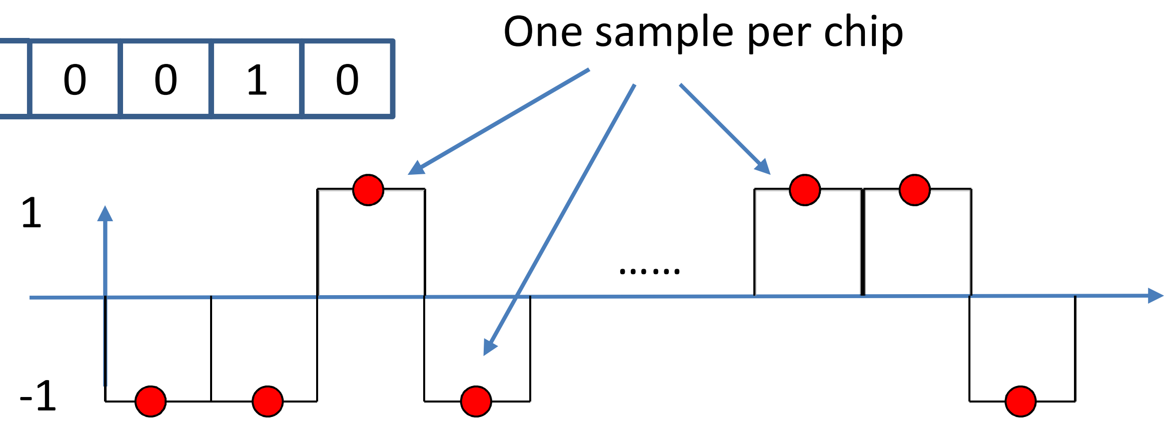
LOCAL CODE GENERATION (II/II)

The codes are provided in **binary format** and sampled at the **chip rate**

1) From **binary** to **bi-polar** format

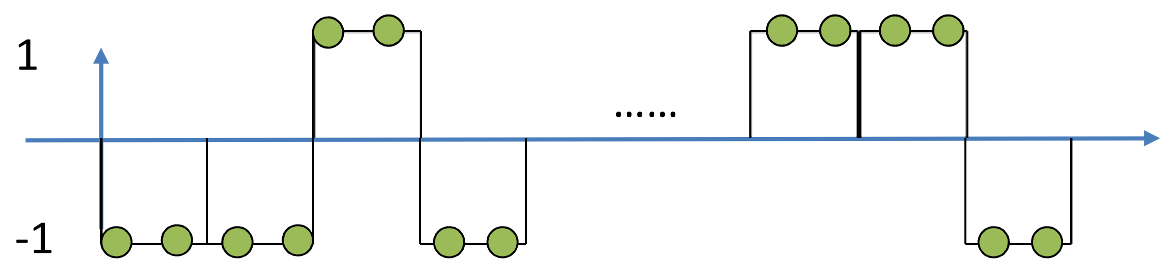
1	1	0	1	....	0	0	1	0

$$\begin{cases} 0 \rightarrow 1 \\ 1 \rightarrow -1 \end{cases}$$



2) From the **chip rate** to the signal **sampling frequency**

`New' code samples are obtained by assuming that the code is constant (for the BPSK case) over the chip duration



And for BOC signals?

PLAYING WITH THE CODES

Re-sampling, i.e. change of signal rate, is performed by the function

```
function resampledCode = ResampleCode(code, numPoints, …
                                      fs, tau0, fc )
```

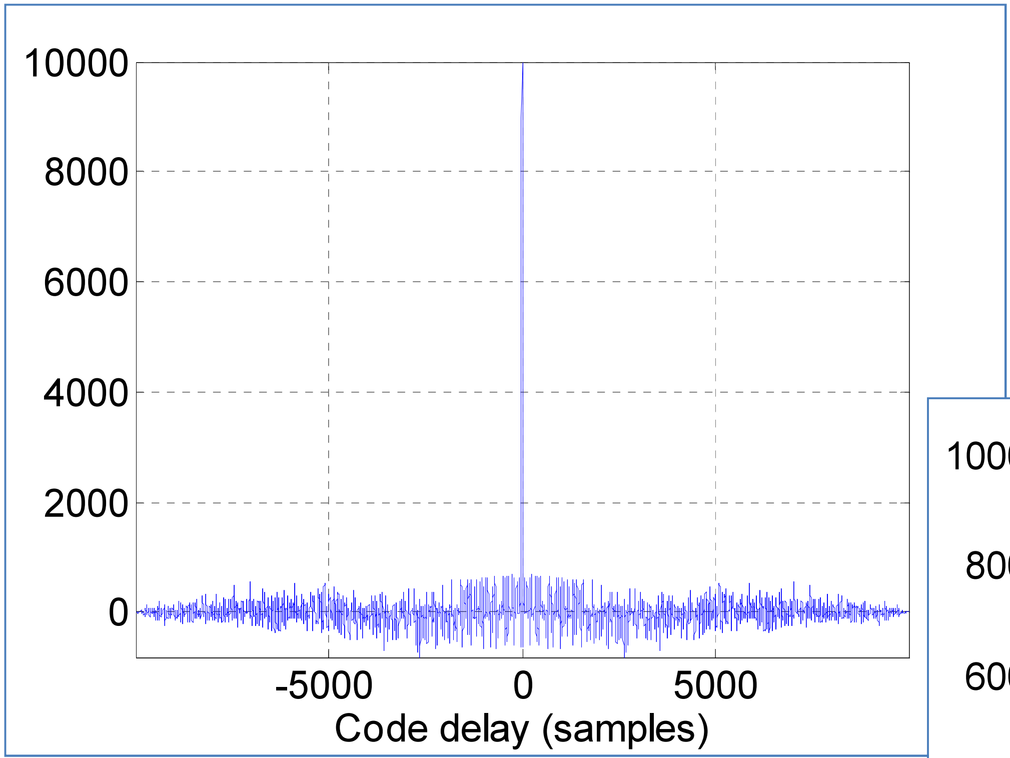
Now we have the local code (locC)

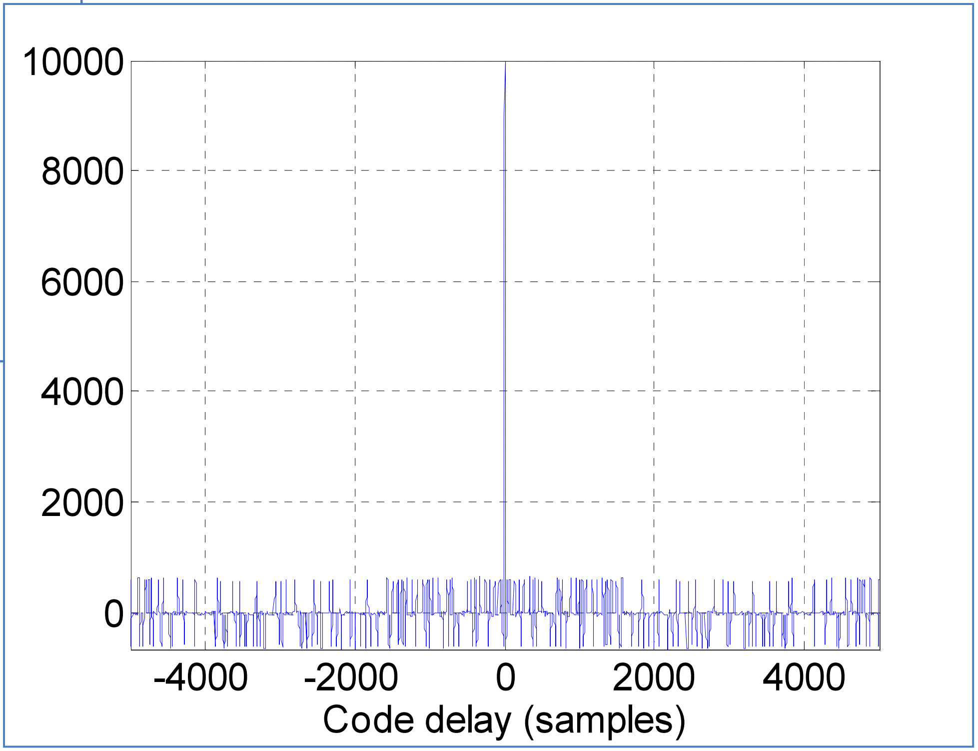
Investigate the property of the code with the following commands:

```
lcor = xcorr( locC );                              % linear correlation
ccor = real( ifft( fft( locC ).*conj( fft(locC) ) ) );    % circular correlation

plot( lcor );
figure;
plot( ccor);
```

LINEAR AND CIRCULAR CORRELATION





CROSS-AMBIGUITY FUNCTION (I/II)

Evaluate the similarity between the input signal and locally generated replicas

$$S_k(\tau, f_d) = \frac{1}{N} \sum_{n=kN}^{(k+1)N} r[n] \underbrace{c(nT_s - \tau)\exp\{-j2\pi f_d n T_s\}}$$

Delayed and modulated versions of the local code.

Compensate delay and Doppler shift on incoming signal

The square modulus of the cross-ambiguity function is evaluated by the function

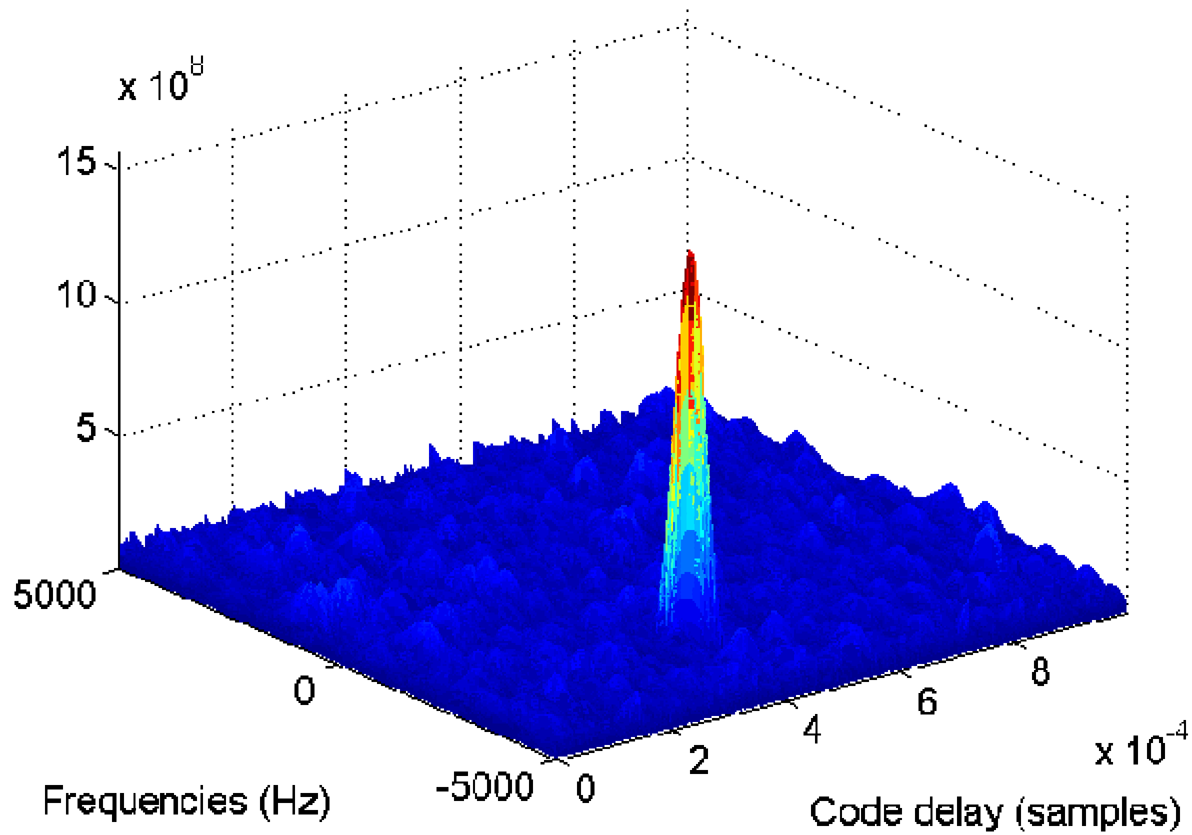
function sspace = DftParallelCodePhaseAcquisition(sig, locC, N, Nd, DopStep, fs, fi)

using an FFT based approach

CROSS-AMBIGUITY FUNCTION (II/II)

All the cells of the search space (each cell is defined by a code delay and Doppler shift value) are evaluated efficiently using the FFT algorithm

The impact of noise can be further reduced by using non-coherent integration (see exercise 1)



A DETECTION PROCESS

Acquisition is a **detection process**: we need to determine the signal presence

There are two hypotheses and two possible decisions

H_0 (Null Hypothesis): the signal is absent

H_1 (Alternative Hypothesis) the signal is present

D_0: the signal is declared absent

D_1: the signal is declared present

The signal is declared present if

Decision Threshold

$$\max_{\tau, f_d} |S(\tau, f_d)|^2 > T_h$$

N.B.: other detection criteria may be used!

DECISION THRESHOLD

When the signal is erroneously declared present (D_1 is selected but H_0 is true), a **false alarm** occurs

The decision threshold is selected in order to **control the probability of false alarm**

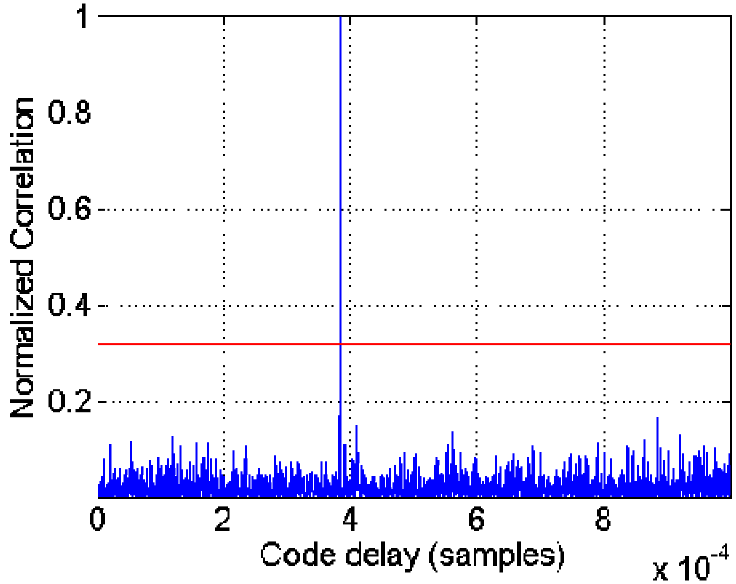
$$P\left(\max_{\tau, f_d} |S(\tau, f_d)|^2 > T_h \middle| H_0\right) = const.$$

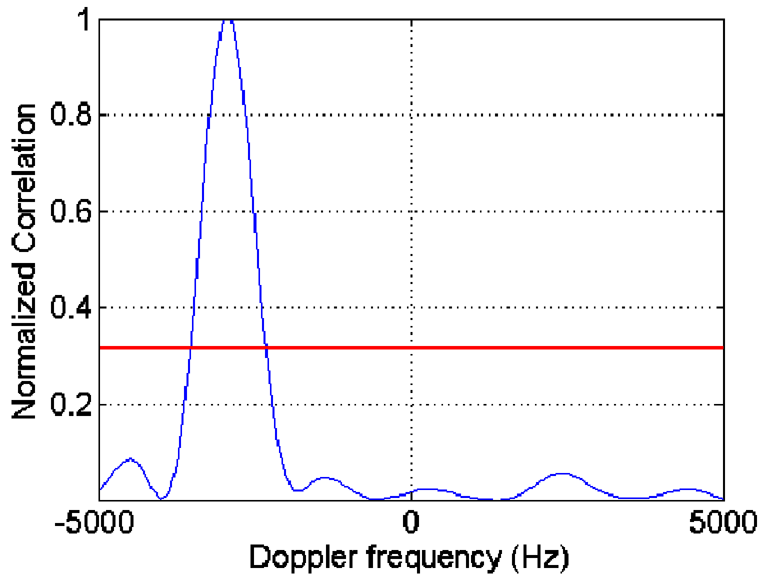
In the code: the decision threshold is computed in the 'decision logic block'

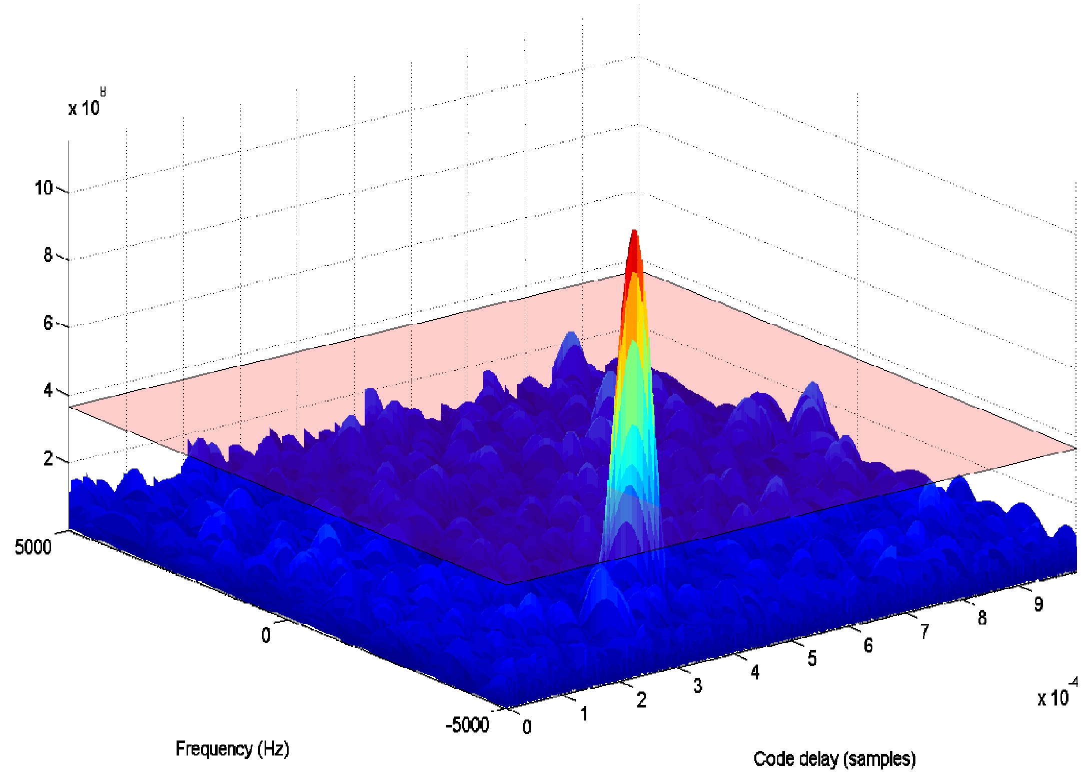
The false alarm probability is at first fixed and the decision threshold is computed by inverting the above equation

N.B.: the false alarm probability depends on the variance of the correlator output. This variance is determined by the 'NoiseVarianceEstimator' function

EXPECTED RESULTS







EXERCISE 1

Non-coherent integration

The basic acquisition script provided for the lab also implements non-coherent integration.

Experiment with the number of non-coherent integrations and observe the impact on the decision statistic used for acquisition

EXERCISE 2

Acquisition of BOC(1, 1) signals

Modify the acquisition script for the processing of BOC(1,1) modulated signals.

The Galileo codes are stored in the file GalCodeE1c.mat

Suggestion:

Consider the BOC(1,1) as a ‘modifier’ for the local code. In particular each sample 1 is replaced by two samples (1, -1) and each sample -1 is replaced by (-1,1)

What is the effect of the BOC(1,1) on the Cross-Ambiguity Function?