DEPARTMENT OF TECHNOLOGY AND BUILT ENVIRONMENT

A Signal Detector for Cognitive Radio System

Aldo Buccardo

June 11, 2010

Master Program in Telecommunications Engineering
Examiner: Magnus Isaksson
Supervisor: Niclas Bjorsell

Table of Contents

Chapter 1

Introduction

1.1 Overview

At present, the radio spectrum support a wide range of business including narrow and broadband mobile telecommunications, medical and scientific research, cultural activities, aeronautical and marine communications, defense and emergency services. Thus communications and radio services are increasingly important to economic and social development and the radio spectrum became an important natural resource. With the appearance of new wireless systems, the demand for radio spectrum is increasing and the scarcity of this resource is becoming obvious. Radio spectrum management to ensure the efficiency of radio communications equipment and services is necessary. Clearly, the regulatory process of ensuring the optimal use of the spectrum needs to be flexible and responsive in order to adapt to changes in technologies, demand, markets and public policy. Efficiencies may to be limited for safeguarding the provision of public services such as safety, defense and public broadcasting, or it can to be compromised by old

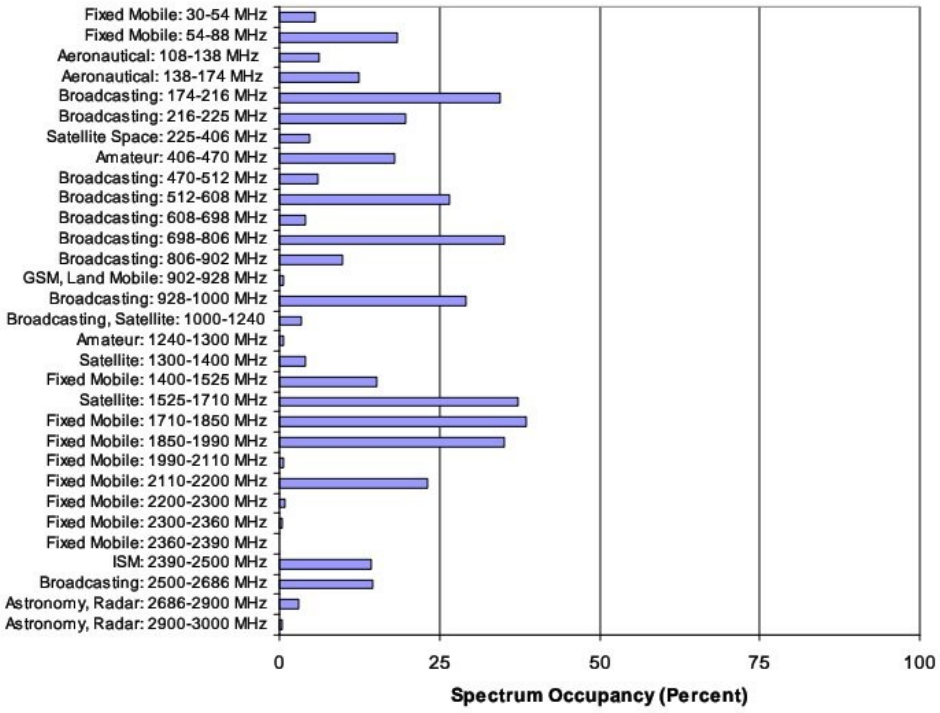

Figure 1: Measured Spectrum Occupancy in Dublin, Ireland

commercial and government policies. Actually every wireless system got its own license to avoid interference and for new technologies is difficult to operate in their spectra because these are already occupied from commercial operators and government.

To address the scarcity of radio spectrum, the Federal Communications Commission (FCC) published a report[1], prepared by Spectrum Policy Task Force (SPTF)[1], where you can read that some bands are crowded while others are fewer used. For example Figure 1 shows the spectrum occupancy in Dublin in 2007 measured by Shared Spectrum Company[2]. Dublin is the most densely populated city in Ireland and it is shown that the spectrum occupancy during a high usage period in a normal work week is only 13.6% in

1: The Spectrum Policy Task Force is a Team of high-level, multi-disciplinary professional FCC staff for identifying and evaluating changes in spectrum policy that will increase the public benefits derived from the use of radio spectrum.

mean. To ensure the efficient management of radio spectrum new policies and technologies are necessary. For example, an actual measure turns off the analog television broadcasting and switches digital television system. In any case, it's difficult to reclaim and release spectrum bands already licensed, so a new dynamic policy of spectrum licensing is employed by FCC and PTS (Swedish Post and Telecom Agency). They define two types of licensed user: Primary User (PU) and Secondary User (SU). To decrease the unused spectrum the PU could share spectrum with a SU under special negotiable conditions without releasing his own license. This project needs an intelligent radio system to manage the shared spectrum, sense spectrum used by neighboring devices, change frequency, adjust output power, and even alter transmission parameters. According to the SPTF, Cognitive Radio (CR) is one way to improve the efficiency of the radio spectrum utilization[3]. Actually, CR finds unused spectrum to insert a SU transmission without interfering with PU. Thus this technology can solve the problem of the unused spectrum, but it can be used in many others applications: emergency communications services, broadband wireless networking, multimedia wireless networking, and more. In these cases, the Cognitive Radio system can make decisions about its own behaviors and operations according to its working environment to improve the quality of service. CR is a very important technology for the new generations digital communications and its development is fundamental to guarantee an optimal use of the radio spectrum as well economical and social benefits.

1.2 Motivation

The Cognitive Radio can automatically analyze its radio spectrum environment to identify temporarily vacant spectrum and use it. Moreover, CR is a flexible system because it can change the communications parameters to adapt to channel conditions. Automatism and flexibility are possible

because CR includes Software-Defined Radio (SDR). The basic concept of the SDR software radio is that the radio can be totally configured or defined by the software so that a cheap and simple hardware (a PC with sound card is sufficiently) can be used in many applications. With SDR many hardware components like mixer, filters, modulators, demodulators, are implemented via software. Thus, simply modifying the software, CR can completely change its functionality, or improve its performances, without replace hardware[4]. This characteristic allows to add new features to CR and improve its existing ability. In particular, it's important to study and develop a signal detector for CR that new methods and algorithms can make fast and accurate. Indeed the detection ability is a key functionality for CR, but the signal detector can be used in other projects of spectrum sensing where to identify the presence of a signal is request. An interesting device is the signal classifier[5] that determines the modulation scheme of an input signal, corrupted by noise and interferences, without requiring any knowledge about it. In this application, the detector finds a signal in the spectrum and routes it to the classifier. At present this new instrument is included in a Real Time Spectrum Analyzer[6], However several benefits, such as faster execution time and extensibility to meet new requirements, are given by SDR[7]. Thus, it's possible include a signal classifier module in CR to obtain a better spectrum management and to know which kind of signal is present in a given band and in a given time, but however a signal detector is necessary.

Most research currently focuses on spectrum sensing in CR, but theoretical detection algorithms are not enough. A hardware test platform is required to test and check the performances of the CR networks. An interesting testbed platform for radio network has been introduced by Eric Blossom: GNU Radio. It's a free software (open source) toolkit for building software radio[8] that respects the SDR principles. Moreover GNU Radio is designed to work with an inexpensive hardware device called USRP2 (Universal Software Radio Peripheral) that includes acquisition board and RF interfaces. Thus,

GNU Radio is a perfect platform for testing the signal detector in real transmission conditions, but to implement a signal detector module is also an opportunity to contribute at GNU Radio improvement.

1.3 Purpose

Spectrum sensing plays a main role in CR because it's important to avoid interference with PU and guarantee a good quality of service of the PU. In adding a detection device is useful for signal classifier that improves CR ability. The project of a signal detector has been developed from University of Sannio (Benevento, Italy) in collaboration with University of Gavle. In the Laboratory of Signal Processing and Measurement Information (LESIM) [9], the device has been implemented by means a PC, an high quality signal generator and antennas [10]. In this work, the detection will be checked making the wireless link between two devices of lower quality. In this way, the detection is nearest to the real world. Thus a GNU Radio platform including a signal detector prototype is presented. The detection method is eigenvalue based [11]. The SVD algorithm is used (Singular Value Decomposition) to estimate the eigenvalues of the correlation matrix. Exploiting the relationship between University of Gavle and Ericsson to realize a CR system, the detector prototype has been made and tested in company laboratories. To support the detection method and the implementation choices this thesis reports experimental results and explanatory conclusions.

1.4 Outline

This master's thesis consist of seven chapters and begins with an overview of the spectrum management. Moreover objectives and motivations of this work have been presented like possible solution of the limited spectrum problem.

The second chapter describes Cognitive Radio and it studies in deep the Spectrum Sensing function, SDR capability and IEEE standard. In the Chapter 3 the detection method eigenvalues based improved with SVD algorithm is presented. Chapter 4 describes the instruments and the architecture of GNU Radio system that allow to realize the signal detector. Chapter 5 reports the detection test results made with simulated and emulated signals. Comments of results and final conclusions are finally insert in the sixth and seventh chapter respectively.

Chapter 2

Cognitive Radio Technology

2.1 CR Approach

In August 1999, Joseph Mitola officially introduced the cognitive radio paradigm[12], which has a built-in programmable and optimized new wireless system. Mitola, in his dissertation[13], described this innovative approach in wireless system as: *the point at which wireless personal digital assistants (PDA) and the related networks are sufficiently computationally intelligent about radio resources and related computer-to-computer communications to detect user communications needs as a function of use context, and to provide radio resources and wireless services most appropriate to those needs.* In summary, a full CR, called Mitola Radio, is a fully re-configurable radio device that can "cognitively" adapt itself to both user's needs and its local environment. CR has a great potential to provide more benefits to telecommunications systems. It's predicted that CR identifies portions of unused spectrum to transmits in it to avoid interferences with others users. Moreover, CR has a lot of capabilities that can improve the spectrum access

and communications services. CR could incorporate some spectrum control module to check and select the appropriate operating frequency ranges in function of the country in which they are used. This ability could be improved incorporating in CR the signal classifier proposed in [6]. CR could facilitate interoperability between different devices in which carrier frequency and/or transmission systems are different. For example, CR can work like a bridge between two systems receiving a signal in one format and transmitting the same information in different mode. Using the Dynamic Frequency Selection (DFS), CR improves the spectrum management changing dynamically the operating frequency. An important CR feature is the Transmitter Power Control (TPC) which allows transmission at the allowable limits when necessary. For safety reasons a TPC installed in a mobile device could decrease the power when it is in airport. These are few examples of CR potential applications, actually there are many possibility.

2.2 CR Architecture

This section summarizes of the ideal Cognitive Radio Architecture (CRA) that is more developed in [12][14]. The fundamentals design rules by which sensors, Autonomous Machine Learning (AML) and SDR may be integrated to create aware, adaptive and cognitive radio are presented.

2.2.1 CR functional components

A simple description of CRA is the set of six functional components showed in Figure 2.

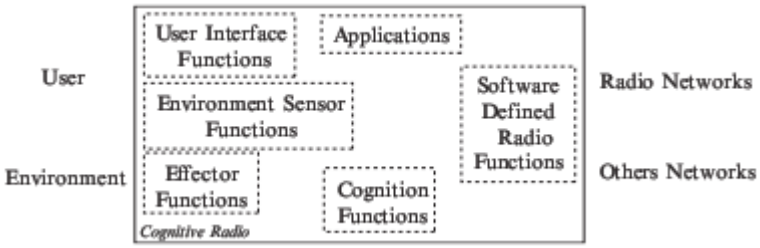

Figure 2: Functional Cognitive Radio Architecture (CRA)

A specific hardware to implement these function is necessary, but CR is a emergent technology so the boxes can be considered functional, not physical. These functional components are[14]:

1. the User Sensory Perception (User SP) interface includes audio and video sensing and perception (observation) function;
2. the local environment sensors (location, temperature, accelerometer, compass, etc.);
3. the system applications (media independent services like playing a network game);
4. the SDR functions (which include RF sensing and SDR radio applications);
5. the cognition functions (symbol grounding for system control, planning, learning);
6. the local effector functions (speech synthesis, text, graphics, and multimedia displays) .

Each block is independent, but relationships between them exist so they cooperate together. The user interface consist of buttons, microphones and video devices; it encodes and interprets user and environment's inputs to have high performances in transmission. In adding, local sensors check the environment situation to perform an efficient access to wireless network. User's data and others informations are elaborated by system applications component to guarantee continuity, integrity and coherence of multimedia

streams. Different information systems can interoperate because they are managed by the cognition functions. Obviously the first afferent of system applications is SDR to transmit the multimedia information on wired or wireless channel. SDR behavior depends of environment conditions. The cognitive system may select the optimal parameters configuration processing the environment informations. In general, the cognitive system is the main block for CRA; it has the full control on the blocks interfaces via sensory perception and AML technologies. Finally, the local effectors synthesize speech along with traditional text, graphics, and multimedia displays .

2.2.2 Learning: The Cognition Cycle

The structure in Figure 2 and the functional components do not deepen the intelligent aspect of CR. In a ideal CRA a machine learning is integrated to provide the fundamental capabilities required for CR: to observe (sense, perceive), orient, plan, decide, act, and learn. The rules, by which these capabilities are related, are included in a reactive sequence, illustrated in Figure 3, called cognitive cycle[13][14].

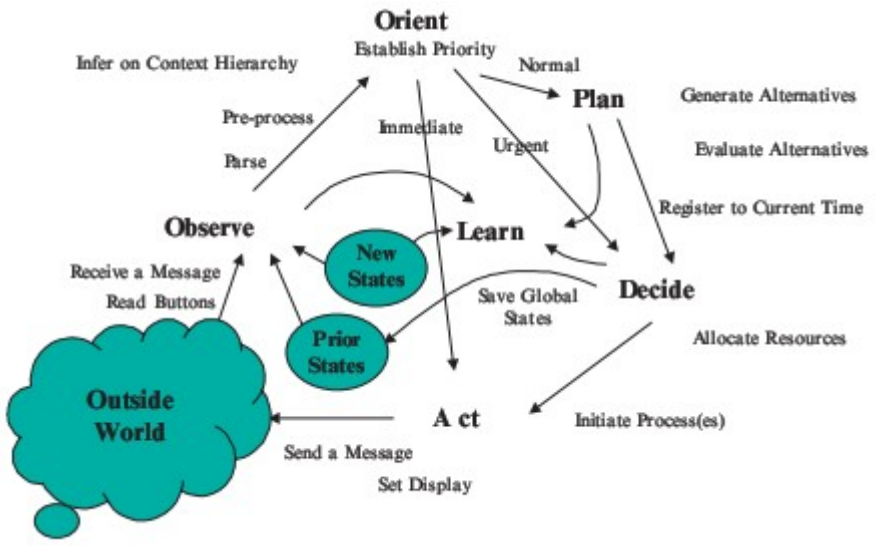

Figure 3: The cognitive cycle

Through User Interface and Environment Sensors, the CR system continually observes (senses and perceives) own environment. External stimuli are elaborated in the Orient phase to determine their meanings and their priority. The system is trained to respond to a stimuli, like particular RF conditions, signals, and more, with a immediate reaction or a planned activity. Typically, the reactions and CR behaviors are defined by a network or planned with specifics tools (Cognitive Functions). Typical actions are to save data or to search a new RF channel. In any case, the actions are not always unique. It's possible that the current stimulus is common to more plans so the system should decide. The choice depends by CR experience and user's application, but, in critical situations, to alert the user is possible. The perception, observation, decision and action functions cause the CR learning: the system counts the occurrences of the events (it makes a statistic) and remembers its past decisions based on the external stimuli and user's preferences. The learning also occurs when new decision models and planning tools are embedded in CR system.

2.2.3 SDR: Software Defined Radio

The ideal Cognitive Radio is a system in which Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC) convert signals to and from RF and the modulator, demodulator, mixers, filters and other electronic devices are replaced by software. SDR transforms the analog world in a digital world so that the radio system may be processed by computer. This means that all modulation schemes are handled digitally with performances difficult, or impossible, to obtain in analog circuits. This digital radio allows great flexibility so that it can adapt itself to the environment. In this case, SDR has a RF front-end able to access more that 2.5 GHz for supporting all kinds telecommunications services in television band, GSM band, air traffic control band, and more. For RF frequencies, ADCs and DACs consume many

electric power so CR needs a SDR that conserves the life battery for the portable devices. SDR flexibility is possible if the hardware architecture can change its internal software: FPGAs and DSPs can be used, but the ideal SDR needs infinite memory for learning, so that the hardware complexity grows inevitably. At present, the ideal SDR is not be practical and affordable so a real CR prototype is far.

2.3 Spectrum Sensing Cognitive Radio

The definition of Full CR implies Intelligent Signal Processing (ISP) at all layers of the OSI model and, at the moment, this is not possible. There are two principals problems: to design a hardware with the ability to intelligently make decisions based on own local environment; to develop SDR technologies to enable full reconfigurability. Because the cognitive science is in its infancy, these problems may not be solved for many years. For example, a device capable of operating in any frequency band up to 3GHz without the need for rigid front-end hardware (excluding the antenna) will not be available before 2030[15]. Anyway, true cognition and fully flexible radio technologies may not be needed: a simpler intelligence at the physical layer coupled with lower degrees of ISP, could provide significant benefits over traditional types of radio. This CR is classified spectrum sensing cognitive radio[16] because its main function is spectrum sensing. In other words, CR focus its functions to increase the spectrum efficiency in specific bands. At present some elements of CR are used in the current radio systems: Adaptive Frequency Hopping (AFH) is included in the Bluetooth standard to avoid interference with other wireless technologies in the 2.4GHz unlicensed radio spectrum[17].

2.4 Challenges in spectrum sensing

To realize a spectrum sensing cognitive radio, as well as bring benefits at old radio systems, is a fundamental step to make a full CR so spectrum sensing function is an important objective for research world. Spectrum Sensing is not a easy problem to solve. Actually, it can be formulated as a class of optimization problems that arise in CR networks to maximize the spectrum efficiency. A sensing algorithm design should optimize the implementation complexity, the interference measurement, the power absorption, and more. Anyway, it must also tackle others specific problems of the communications systems. Multipath fading would normally be expected to interfere with the signals between the target under detection and CR, so it's difficult to understand if a signal doesn't exist or it's reduced by a bad channel. This problem is called "hidden PU"[18][19] because a SU could transmit in a apparent hole of spectrum that on contrary hides a PU signal. Other important feature of CR is the sensing time that may be as fast as possible. Hence, the detection method should be able to recognize the PU within a definite time. Finally, we can not forget that CR network consists of multiple SU and PU, thus interferences between SU can occur so that detection system becomes unreliable. Cooperative sensing is recommended in literature for solving these problems[18][19], but definitive solutions don't exist yet. Actually CR is an open yard so, to uniform the building of CR, a coordinated work is necessary, so the most important challenge is to make a CR standard.

2.5 Cognitive Radio Standard: IEEE SCC41

A lot of companies and research groups in numerous university are focusing their efforts on CR. Since many technical and economical aspects are associated with CR, there is a need standardize the processes, terms and so on. To address the issues about the development of new generation Radio system, IEEE has created the Standards Coordinating Committee 41 (IEEE

SCC41) [20] to develop standards related to dynamic spectrum access networks. IEEE SCC41 is divided in six work groups each responsible for evolve the standardization of a particular aspect of CR. They are called IEEE 1900.x where .x is working group number[20]:

1. *Standard definitions and concepts for spectrum management and advanced radio system technologies*. The working group number one is responsible to produce a standard that provides technically precise definitions and explanations of key concepts in the fields of spectrum management, policy defined radio, adaptive radio, software defined radio, and related technologies
2. *Recommended practice for interference and coexistence analysis*. In a world where the spectrum is shared, the accurate frequency measurements become a crucial requirement. To measure the interference between radio systems operating in the same frequency band, the working group number two to recommend the interference analysis criteria.
3. *Recommended practice for conformance evaluation of Software Defined Radio software modules*. Because SRD is very important component for future radio systems, a set of test methods for conformance evaluation of SDR software is necessary. The third working group will realize a standard with procedures and recommendations to provide guidance for validity analysis of proposed SDR software.
4. *Architectural building blocks enabling network-device distributed decision making for optimized radio resource usage in heterogeneous wireless access networks*. IEEE 1900.4 was published on February 27th, 2009. It gives the methods to improve overall capacity and quality of service of wireless systems in a multiple radio access technologies environment, by defining an appropriate system architecture and protocols which will facilitate the optimization of

radio resource usage. At present, the fourth working group works on two projects: 1900.4a and 1900.4.1. The first facilitates cost-effective and multi-vendor production of wireless access system, including cognitive base stations and terminals . The second provides detailed description of interfaces and service access points defined in the IEEE 1900.4 standard.

5. *Policy language and policy architectures for managing cognitive radio for dynamic spectrum access applications*. IEEE 1900.5 will define a set of policy languages, and their relation to policy architectures, to manage the features of cognitive radios for dynamic spectrum access applications.
6. *Spectrum sensing interfaces and data structures for dynamic spectrum access and other advanced radio communication systems*. Recently proposed advanced radio systems use a proprietary architecture so the interoperability with other similar systems could be difficult. To support the interoperability, this group develops a standard that defines the interfaces and data structures required for exchange of sensing related information. Obviously, the logical interface and supporting data structures used for information exchange are defined abstractly without restricting the sensing technology, client design, or data link between sensor and client.

Chapter 3

Signal Detector

3.1 Background

The objective of new technologies is to grow the capacity and speed of transmission data. For this new transmission systems high performances and important features are required:

- broadband hardware to use different bands and channels at the same time;
- flexible software to elaborate different kinds of signals and modulations.

To satisfy these needs, the idea of a CR was born. This new telecommunications system improves the spectrum efficiency and adapts itself at the transmission conditions. Spectrum sensing became an essential functionality to detect spectral holes and to opportunistically use these frequency bands without causing harmful interference to legacy (primary) users. Moreover. signal classification is very important in a scenario in which several technologies and services can be present at the same time. In this

case, an effective spectrum monitoring could be achieved by recognizing the signals which occupy the spectrum. Both the CR functionalities, spectrum sensing and signal classifier, need a device able to sense the spectrum and to estimate the signal presence. The Signal Detector does this giving a binary output: it's 0 when a unused frequency band is found; it's 1 when a signal is detected. In Figure 4 is showed how the detector could be used in a real CR system: when a signal is found it activates the classifier; if the analyzed band is free the detector enables the transmission for SU.

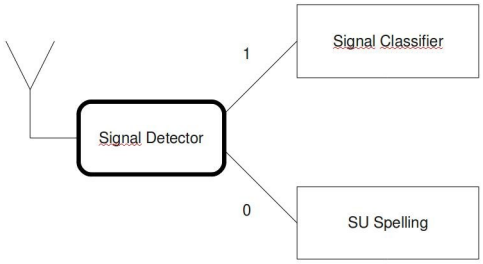

Figure 4: Signal Detector in CR system

The Signal Detector implemented in a generic radio platform is a functional block with a user interface and a system interface (Figure 5). Through the first interface the user has interaction with the detection method so he can set some parameters to change the detection performances; the second interface provides to the detection method the data stream under test, and gives to the user some informations about the transmission conditions.

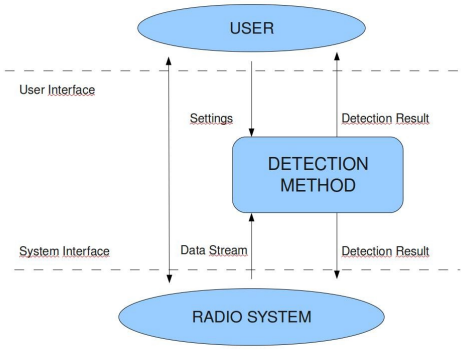

Figure 5: Signal Detector Interfaces

The detection result is sent to the system to enable the signal classifier or the SU Transmission. Moreover the detection result is showed to the user for two reasons. He can use the result for others applications; with the environment informations and the detection result, the user can give new instructions to the system to improve the transmission performances. In this way, the user adds new laws for the learning of CR. Anyway, the core of the Signal Detector is the detection method. This algorithm should be flexible accurate and fast in execution, so the objective is to make a good trade-off between computational complexity and knowledge a-priori of the signals under test. To understand how the method has been chosen, in the next paragraphs, the most common detection methods are presented and compared.

3.2 Detection methods in literature

The detection method should overcome many obstacles that make the detection difficult. These factors are: the typical low signal to noise ratio (SNR) in the transmissions; fading and multi-path in wireless communications; the instable noise level in the channel; the need of a low sensing time, and more. Many detection methods exist, but not all of them are independent from these factors so they can not be efficacious. In this section the most important detection methods are presented and how these are limited by the negative factors is explained.

3.2.1 Matched Filter

The best way to detect signals with maximum SNR is to use a matched filter receiver[21]. Its most important skill is the low execution time, but to know the signal proprieties is needed. This method includes the demodulation of the signal. This means that the receiver should agree with the source, estimate the channel conditions and to know the signal nature. This method is useful for dedicated receivers like in TV transmissions.

3.2.2 Energy detection

This is a basic and common approach to spectrum sensing since it has low computational and execution complexity [22]. An energy detector sets a threshold according to the noise floor and compares it with the energy of the data stream in input. This detector (Figure 6) only requires minimal information, such as the signal bandwidth and carrier frequency. The input signal selects the interesting bandwidth by a band pass filter, than it is sampled. The digital implementation of this method use the Fast Fourier Transform (FFT), so the absolute value of the samples are squared and integrated over the observation band.

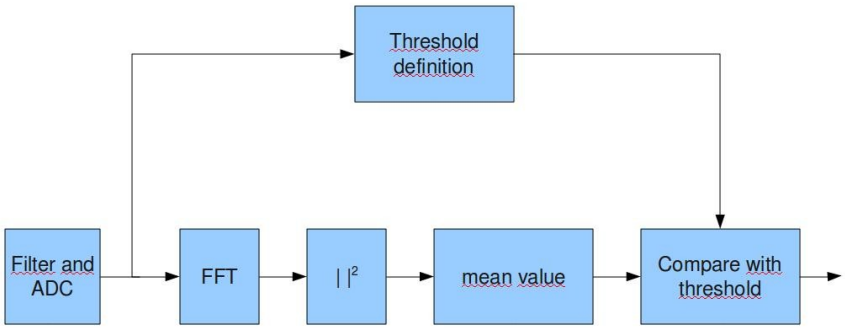

Figure 6: Energy Detector

Finally, according to a comparison between the output of the integrator and threshold, the presence or absence of the primary user can be detected. In this architecture, the size of FFT provides the resolution bandwidth and good detection of narrowband signals, but this has to consider a costs: long FFT needs many hardware resources and long sensing time. Moreover the threshold depends of the noise level so it's very difficult to fix it in changing conditions. Finally, the detection doesn't recognize the modulated signals from interference frequencies or high noise.

3.2.3 Feature detection

Most signals have statistical proprieties that vary periodically with time, which are called cyclostationary features. Hence, more accurate detection can be achieved by exploiting the inherent periodicity of the autocorrelation function of the signals. In this case a Spectral Correlation Function is defined:

$$S_x^{\alpha}(f) = \lim_{\substack{T\to\infty \\ \Delta t\to\infty}} \frac{1}{\Delta t}\int_{-\Delta t}^{+\Delta t}\frac{1}{T}X_T(t, f+\frac{\alpha}{2})X_T^*(t, f-\frac{\alpha}{2})\,dt$$

where $X_T(f)$ is the Fourier transform and α is called cycle frequency that means the frequency separation of the correlated spectral components. Modulated signals have a SCF with specific characteristics so, comparing them with a list of typical features, the detection is possible. Anyway this method needs a great complexity computational and the knowledge of some signal parameters of the signal under test like the carrier frequency[22].

3.2.4 Eigenvalues detection

This is a new detection method presented to research community in September 2007[11]. The core of this method is the ratio of the eigenvalues of the covariance matrix of the received signal. Actually there are two types of detection method eigenvalues based: maximum-minimum eigenvalue detection, which compares the ratio of the maximum eigenvalue and the minimum eigenvalue with a threshold; energy with minimum eigenvalue detection, which compares the ratio of the average energy and the minimum eigenvalue with a threshold[23]. The first method doesn't need a-priori knowledge like the second that needs to know the SNR value. Thus the maximum-minimum approach can overcome the noise uncertainty problem and also retains the advantages of the energy approach. In this way the

method detects signals with unknown source, unknown channel and unknown noise power. The drawback is its complexity.

3.2.5 Comparison

Each method presented has advantages and disadvantages in signal detection. These are summarized in Table 1 where several aspects of the methods are compared[22]:

- Execution time. The ideal signal detector should work in real time so that the execution time is the shorter possible.
- Noise rejection. Skill of the method to be immune to the white noise.
- Knowledge a-priori. How much information the method needs to detect the signal. In CR this information should be minimum.
- Computational complexity. Capacity calculation required to detect the signal.
- Interference rejection. Skill of the method to be immune to the disturbs different by white noise.

	Execution time	Noise rejection	Knowledge a-priori	Computational complexity	Interference rejection
Matched Filter	GOOD	MEDIUM	HIGH	LOW	HIGH
Energy Detection	HIGH	LOW	NONE	LOW	LOW
Features Detection	LOW	HIGH	MEDIUM	HIGH	HIGH
Eigenvalues Detection	MEDIUM	HIGH	NONE	MEDIUM	LOW

Table 1: Comparison of methods

In conclusion, the method that offers the most flexibility is the eigenvalues based method. For this reason it is implemented in the core of the signal detector proposed.

3.3 Detection methods proposed

In this section the core of the signal detector is presented. The block diagram in Figure 5 can be exploded for a deeper analysis. In Figure 7 the main parts of the method are showed: the sampled signal comes from the System Interface to build the covariance matrix or the Hankel matrix; the eigenvalues of the matrix are calculated with a specific algorithm to make the ratio maximum-minimum; with the user's settings the threshold is defined and the comparison with the eigenvalues ratio detects the signal presence.

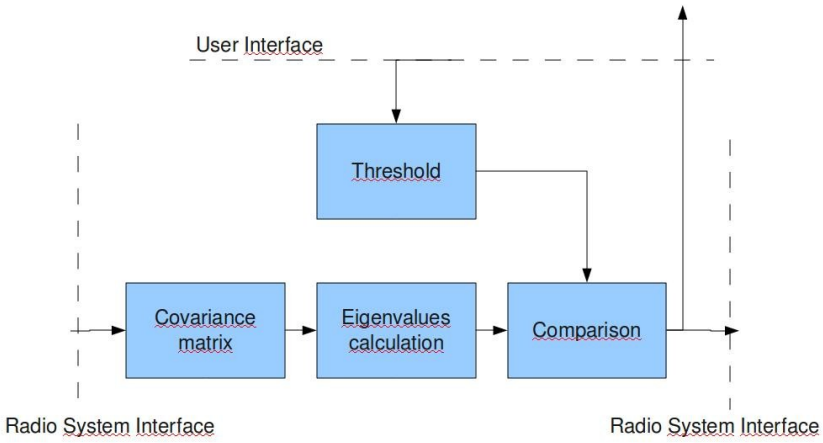

Figure 7: Detection Method

The discussion is divided in three subsections to highlight the main parts of the method: why the eigenvalues ratio can be used to detect signals; how to find the eigenvalues with a single source and a single receiver; how to define the threshold.

3.3.1 Eigenanalysis of the autocovariance matrix

To explain the detection algorithm, the eigenanalysis of the autocovariance matrix is necessary. It's assumed that the random process $x(n)$ is wide-sense stationary and it is linear combinations of p basic components $s_i(n)$

$$x(n)=\sum_{i=1}^{p} a_i s_i(n) \tag{1}$$

Since sequence observed is $y(n)=x(n)+\eta(n)$ where $\eta(n)$ is a white noise sequence with spectral density σ^2, the $L\times L$ autocovariance matrix for $y(n)$ can be expressed as

$$\Gamma_{yy}=\Gamma_{xx}+\sigma_\eta^2 \boldsymbol{I} \tag{2}$$

where Γ_{xx} is the autocovariance matrix for the signal $x(n)$, $\sigma_\eta^2 \boldsymbol{I}$ is the autocovariance matrix for the noise and L is the length of the covariance. Note that if $L>p$, Γ_{xx} which is of dimension $L\times L$ is not of full rank. Now let us perform an eigen-decomposition of the matrix Γ_{yy}. Let the eigenvalues be ordered in decreasing value with $\lambda_1 \geqslant \lambda_1 \geqslant \cdots \geqslant \lambda_L$ and let the corresponding eigenvectors be denoted as $\{\boldsymbol{v}_i,\ i=1,\ldots,L\}$. We assume that the eigenvectors are normalized so that $\boldsymbol{v}_i^H \cdot \boldsymbol{v}_j=\delta_{ij}$ (H denotes the conjugate transpose). In the absence of noise the eigenvalues λ_i, $i=1,2,\ldots,p$, are nonzero while $\lambda_{p+1}=\lambda_{p+2}=\cdots=\lambda_L=0$. Thus, the eigenvectors $\boldsymbol{v}_i$, $i=1,2,\ldots,p$ span the signal subspace. These vectors are called *principal eigenvectors* and the corresponding eigenvalues are called *principal eigenvalues*. In presence of noise the eigen-decomposition separates the eigenvectors in two sets. The set $\boldsymbol{v}_i$, $i=1,2,\ldots,p$, which are the principal eigenvectors, span the signal subspace, while the set $\boldsymbol{v}_i$, $i=p+1,\ldots,L$, which are orthogonal to the principals eigenvectors, are said to belong to the noise subspace. It follows that the signal $x(n)$ is simply linear combinations of the principal eigenvectors. Finally, the variances of the projections of the signal on the principal eigenvectors are equal to the corresponding eigenvalues of the covariance matrix[24]. So the principal eigenvalues are the power factors in the new signal space.

3.3.2 Maximum-minimum eigenvalues based detection

The method presented in [11] is more generic and it assumes that there are many receivers and sources. Anyway the signal detector presented in this thesis is designed to work with a only source and a only receiver. In this way the method is easier.

The detection problem can be formulated with two hypothesizes: no signal and only noise; signal with additive white noise. The binary hypothesis test can be replaced by:

$$\begin{aligned} H_0&: x(n)=\eta(n),\ \ n=0,1,\ldots \\ H_1&: x(n)=\sum_{k=0}^{N} h(k)s(n-k)+\eta(n) \end{aligned} \tag{3}$$

Where the discrete signal at receiver is denoted by $x(n)$, $s(n)$ is the source signal, $h(k)$ is the channel response, N is the order of channel and $\eta(n)$ are the noise samples. Considering a subsample L of consecutive outputs and defining

$$\begin{aligned} \hat{x}(n)&=\left[x(n), x(n-1),\ldots, x(n-L+1)\right]^T \\ \hat{\eta}(n)&=\left[\eta(n), \eta(n-1),\ldots, \eta(n-L+1)\right]^T \\ \hat{s}(n)&=\left[s(n), s(n-1),\ldots, s(n-N_1-L+1)\right]^T \end{aligned} \tag{4}$$

we get

$$\hat{x}(n)=\mathbf{H}\,\hat{s}(n)+\hat{\eta}(n) \tag{5}$$

where $\mathbf{H}$ is a $L\times(N+L)$ matrix, defined as

$$\mathbf{H}=\begin{bmatrix} h(0) & \cdots & h(N) & \cdots & 0 \\ & \ddots & & \ddots & \\ 0 & \cdots & h(0) & \cdots & h(N) \end{bmatrix} \tag{6}$$

The following assumptions on the statistical proprieties of the transmitted symbols and channel noise are assumed:

A) Noise is white

B) Noise and transmitted signal are uncorrelated.

Let $\mathbf{R}(N_s)$ be the sample covariance matrix of the received signal, that is,

$$\mathbf{R}(N_s)=\frac{1}{N_s}\sum_{n=L}^{L-1+N_s}\hat{x}(n)\hat{x}^H(n) \tag{7}$$

where N_s is the number of collected samples. If N_s is large, based on the assumptions A and B, we can verify that

$$\mathbf{R}(N_s)\approx\mathbf{R}=\mathrm{E}(\hat{x}(n)\hat{x}^H(n))=\mathbf{H}\mathbf{R}_s\mathbf{H}^H+\sigma_\eta^2\mathbf{I}_L \tag{8}$$

where $\mathbf{R}_s$ is the statistical covariance matrix of the input signal, $\mathbf{R}_s=\mathrm{E}(\hat{s}(n)\hat{s}^H(n))$, σ_η^2 is the variance of the noise, and $\mathbf{I}_L$ is the identity matrix of order L.

Let $\hat{\lambda}_{max}$ and $\hat{\lambda}_{min}$ be the maximum and minimum eigenvalues of $\mathbf{R}$, and $\hat{\rho}_{max}$ and $\hat{\rho}_{min}$ be the maximum and minimum eigenvalues of $\mathbf{H}\mathbf{R}_s\mathbf{H}^H$. Then $\hat{\lambda}_{max}=\hat{\rho}_{max}+\sigma_\eta^2$ and $\hat{\lambda}_{min}=\hat{\rho}_{min}+\sigma_\eta^2$. Obviously, $\hat{\rho}_{max}=\hat{\rho}_{min}$ if and only if $\mathbf{H}\mathbf{R}_s\mathbf{H}^H=\delta\mathbf{I}_L$ where δ is a positive number. If there is no signal, $\hat{\lambda}_{max}/\hat{\lambda}_{min}=1$; otherwise, $\hat{\lambda}_{max}/\hat{\lambda}_{min}>1$. Thus the ratio can be used to detect the presence of the signal.

3.3.3 The algorithm to find the eigenvalues

There are many methods to estimate the eigenvalues of a matrix, but in this case the Singular Values Decomposition (SVD) is used because it's reliable and accurate. Thus, this subsection explains how SVD is useful to calculate the eigenvalues[25] of the covariance matrix. For a time series $x(n)$ with $n=1,2,\ldots,N$ we can fill a Hankel matrix with $N-L+1$ rows and L columns illustrated as follows:

$$\mathbf{Q}=\begin{bmatrix} q(1) & q(2) & \cdots & q(L) \\ q(2) & q(3) & \cdots & q(L+1) \\ \vdots & \vdots & \vdots & \vdots \\ q(N-L+1) & q(N-L+2) & \cdots & q(N) \end{bmatrix} \tag{9}$$

using the SVD, $\mathbf{Q}$ can be factorized as

$$\mathbf{Q}=\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^H \tag{10}$$

where $\mathbf{U}$ and $\mathbf{V}$ are an $(N-L+1)\times(N-L+1)$ and a $L\times L$ unitary matrix, respectively. The columns of $\mathbf{U}$ and $\mathbf{V}$ are called left and right singular vectors, respectively. $\boldsymbol{\Sigma}=diag(\lambda_1,\lambda_2,\ldots,\lambda_m)$ is a diagonal matrix whose nonnegative entries are the square roots of the positive eigenvalues of $\mathbf{Q}\mathbf{Q}^H$. These nonnegative entries are called singular values of $\mathbf{Q}$ and they are arranged in a decreasing order with the largest on in the upper left-hand corner. It's noted that $\mathbf{Q}\mathbf{Q}^H=\mathbf{R}$ so this method allows to calculate the eigenvalues without to build the covariance matrix.

3.3.4 Threshold definition

To find the threshold for the statistical test it's important to study the statistical distribution of the covariance matrix. Since a finite set of samples is considered, the sample covariance matrix $\mathbf{R}(N_s)$ may be well away to the covariance matrix $\mathbf{R}$ and its distribution becomes hard to find. Moreover there is not information about the signal, actually we even don't know if the signal is present or not, so it's difficult to set the threshold on the detection probability and usually the false-alarm probability P_{fa} is used. The threshold and the false-alarm probability can be found with the latest random matrix theories[26]. We consider the sample covariance matrix of the noise:

$$\mathbf{R}(N_s)=\frac{1}{N_s}\sum_{n=L}^{L-1+N_s}\hat{\eta}(n)\hat{\eta}^H(n) \tag{11}$$

this is a special Wishart random matrix. Recently, I.M. Johnston and K. Johansson have found the distribution of the largest eigenvalue [27][28] for real and complex matrix, as described in the following theorems:

- Theorem 1: Assumed real noise, and defining the quantities

$$\begin{aligned}
&\mathbf{A}(N_s)=\frac{N_s}{\sigma_\eta^2}\mathbf{R}_\eta(N_s)\\
&\mu=(\sqrt{N_s-1}+\sqrt{ML})^2\\
&v=(\sqrt{N_s-1}+\sqrt{ML})\left(\frac{1}{\sqrt{N_s-1}}+\frac{1}{\sqrt{ML}}\right)^{1/3}\\
&lim\ \frac{ML}{N_s}=y,\quad y\in(0,1)
\end{aligned} \tag{12}$$

  M is the number of receivers that for this detector is one; then the quantity

$$\frac{\lambda(A(N_s))-\mu}{v} \tag{13}$$

  converges with probability one to the Tracy-Widom distribution of first order.

- Theorem 2: Assumed complex noise, and

$$\begin{aligned}
&\mathbf{A}(N_s)=\frac{N_s}{\sigma_\eta^2}\mathbf{R}_\eta(N_s)\\
&\mu'=(\sqrt{N_s}+\sqrt{ML})^2\\
&v'=(\sqrt{N_s}+\sqrt{ML})\left(\frac{1}{\sqrt{N_s}}+\frac{1}{\sqrt{ML}}\right)^{1/3}\\
&lim\ \frac{ML}{N_s}=y,\quad y\in(0,1)
\end{aligned} \tag{14}$$

  then the quantity

$$\frac{\lambda(A(N_s))-\mu'}{v'} \tag{15}$$

  converges with probability one to the Tracy-Widom distribution of second order.

- Theorem 3: Assume that

$$lim\ \frac{ML}{N_s}=y,\quad y\in(0,1) \tag{16}$$

Than

$$\lim_{N_s \to \infty} \lambda_{min} = \sigma_\eta^{\,2} (1 - \sqrt{y})^2 \tag{17}$$

Based in the theorems,

$$\begin{aligned} \lambda_{max} &\approx \frac{\sigma_\eta^2}{N_s} \left(\sqrt{N_s} + \sqrt{ML} \right)^2 \\ \lambda_{min} &\approx \frac{\sigma_\eta^2}{N_s} \left(\sqrt{N_s} - \sqrt{ML} \right)^2 \end{aligned} \tag{18}$$

Let F_1 and F_2 be the cumulative distribution function (CDF) of Tracy-Widom of the first and second order respectively, they are the limiting law of the largest eigenvalue of particular random matrices. It is difficult to evaluate the distribution functions so usually they are calculated numerically.

Using the theory, we can analyze the false-alarm probability:

$$\begin{aligned} P_{fa} &= P\left(\lambda_{max} > \gamma \lambda_{min} \right) \\ &= P\left(\frac{\sigma_\eta^2}{N_s} \lambda_{max}\left(A(N_s) \right) > \gamma \lambda_{min} \right) \\ &\approx P\left(\lambda_{max}\left(A(N_s) \right) > \gamma \left(\sqrt{(N_s)} - \sqrt{(ML)} \right)^2 \right) \\ &= P\left(\frac{\lambda_{max}\left(A(N_s) \right) - \mu}{v} > \frac{\gamma \left(\sqrt{(N_s)} - \sqrt{(ML)} \right)^2 - \mu}{v} \right) \\ &= 1 - F_1\left(\frac{\gamma \left(\sqrt{(N_s)} - \sqrt{(ML)} \right)^2 - \mu}{v} \right) \end{aligned} \tag{19}$$

Hence the threshold should be chosen such that

$$\frac{\gamma \left(\sqrt{(N_s)} - \sqrt{(ML)} \right)^2 - \mu}{v} = F_1^{\,-1}\left(1 - P_{fa} \right) \tag{20}$$

With the relations in 10 and $M = 1$ the threshold can be written

$$\gamma = \frac{\left(\sqrt{(N_s)} + \sqrt{(L)} \right)^2}{\left(\sqrt{(N_s)} - \sqrt{(L)} \right)^2} \left(1 + \frac{\left(\sqrt{(N_s)} + \sqrt{(L)} \right)^{-2/3}}{(N_s L)^{-1/6}} F_1^{-1}\left(1 - P_{fa} \right) \right) \tag{21}$$

For complex signal F_1 is replaced by F_2, the CDF of Tracy-Widom distribution of second order.

Chapter 4

GNU Radio

4.1 Introduction to the testbed

To test the technologies proposed, the signal detector needs to be integrated in a radio system for making a real devices and evaluating its experimental performances. Thus there is a need to find a testbed platform able to implement advanced algorithms and realize a real wireless communication. It's important that the testbed is extensible, to support different peripherals and interfaces expansions to satisfy future applications, and that its hardware is able to support high speed data transfer. Actually different kinds of testbed exist, all extensible, scalable and programmable, but each one with different characteristics. The choice becomes a trade-off between complexity, performances and costs. In the subsections below, a short analysis of the most important platforms for cognitive radio is presented. Finally, the reasons to use GNU Radio are reported.

4.1.1 GNU Radio platform

GNU Radio is a free software toolkit for building software radios. Software radio is the technique of getting code as close to the antenna as possible so that it turns radio hardware problems into software problems [29]. The fundamental characteristic of software radio is that software defines the transmitted waveforms, and software demodulates the received waveforms. GNU Radio provides a library of signal processing blocks and the glue to tie it all together. The programmer builds a radio by creating a graph (as in graph theory) where the nodes are signal processing blocks and the arrows represent the data flow between them. The signal processing blocks are implemented in C++ and the graphs are constructed and run in Python. GNU Radio is reasonably hardware-independent and the performances depend by the host PC, but, in any case, today a serious digital signal processing is possible on a common computer desktop. The Universal Software Radio Peripheral (USRP) is the recommended device for interfacing GNU Radio with the real world. It is a inexpensive and minimal hardware that, connected to the PC, executes radio front-end functionalities, AD and DA conversion [30].

4.1.2 Wireless open Access Research Platform

Rice University's Wireless open Access Research Platform (WARP) is a wireless platform built to prototype advanced wireless networks [31]. The project of the Rice University is to create a community to enable high-performance research. In this case a custom hardware is necessary: the WARP infrastructure is a complete system with custom hardware, support packages, design tools and application libraries. The hardware uses FPGAs for DSP-optimized development, where the number of FPGAs can be scaled as necessary to deliver increased computational power as needed. The FPGA board uses a Xilinx Virtex-II Pro FPGA as the main processor, which is very

well suited for high performance wireless algorithms. The software used to control the hardware is Matlab and Simulink. In any case, to maintain and grow the WARP, an open-access repository, which contains building blocks for a wide range of wireless systems, is available. The open-access repository allows researchers to integrate the standard building blocks with custom components to assemble complete systems [32].

4.1.3 Berkeley Emulation Engine 2

The Berkeley Emulation Engine 2 (BEE2) system is designed to be a modular, scalable FPGA-based computing platform with a software design methodology that targets a wide range of high-performance applications, such as Cognitive Radio [33]. This FPGA design directly embeds the PowerPC 405 processor cores into the reconfigurable fabric, minimizing the latency between the microprocessor and the reconfigurable fabric while maximizing the data throughput. BEE2 is therefore well suited for real-time applications. The BBE2 board includes five Vertex-2 Pro 70 FPGAs, four for numeric computational operations and one to control the board. An important characteristic of BBE2 is its programming environment. Others development platform use sequential high-level language, like C or C++, for the microprocessor and low-level language, like VHDL, for programming FPGAs. This approach leads to complicated interfaces, specific applications and ad hoc solutions. Instead BEE2 uses a high level unified computation model, based on Mathworks, Simulink and the Xilinx System Generation Library, for both the microprocessor and FPGAs, enabling more flexibility and facility to use. With these elements, BBE2 can provide over 10 times more computing throughput than a DSP-based system, and over 100 times that a microprocessor-based system.

4.1.4 The choice of GNU Radio

The choice of the platform depends by the context where the detector works. In this case, the device is included in a Cognitive Radio system, so the choice is not difficult. GNU Radio respects in total the CR approach because it's a software platform that implements a radio system without a dedicated hardware. Moreover GNU radio is a cheap system because it's open source and the USRP is a low cost device. Probably GNU Radio is less efficient than others testbeds, but its performances depend by the host computer, so is not easy to compare the computational efficiency. In any case, its simple architecture allows a easy utilization also because it has a functional graphic interface and it's supported by a lively web community. GNU Radio is a very versatile system used in many applications, so its applicability is guaranteed in the future. In this sense, a signal detector block in GNU Radio system represents the first step for the real CR prototyping.

4.2 GNU Radio Project

GNU Radio is an open source Software Defined Radio (SDR) project that was started about ten years ago by Eric Blossom, an electrical engineer. The main idea of GNU Radio was to turn all the hardware problems into software problems, moving the complexity of a radio equipment from the hardware level to the software one, and get the software as close to the antenna as possible. Richard Stallman, the GNU Project founder, liked Blossom's idea and agreed to take the project under the GNU aegis [34]. Eric Blossom, together with his development colleague Matt Ettaus, have realized a project which can turn an ordinary PC into a good quality radio receiver: it's possible to develop a custom non commercial radio receiver just combining and interconnecting appropriate software modules. The new release of GNU Radio includes functional blocks to support various modulations (GMSK,

PSK, QAM, OFDM), error corrections codes (Reed-Solomon, Viterbi, Turbo Codes), and signal processing capabilities (filters, FFTs, equalizers, timing recovery), and, in any case, it's easy to make and add others blocks to the initial library. Each module is able to perform a specific signal processing function with a real-time behavior and with high-throughput.

To make a GNU Radio script is very easy using the graphical tool called GNU Radio Companion (GRC). This program, written in python, allows to edit and to execute a GNU Radio flowgraph connecting the function blocks and setting their parameters. To describe the main features of GRC it is showed in Figure 8.

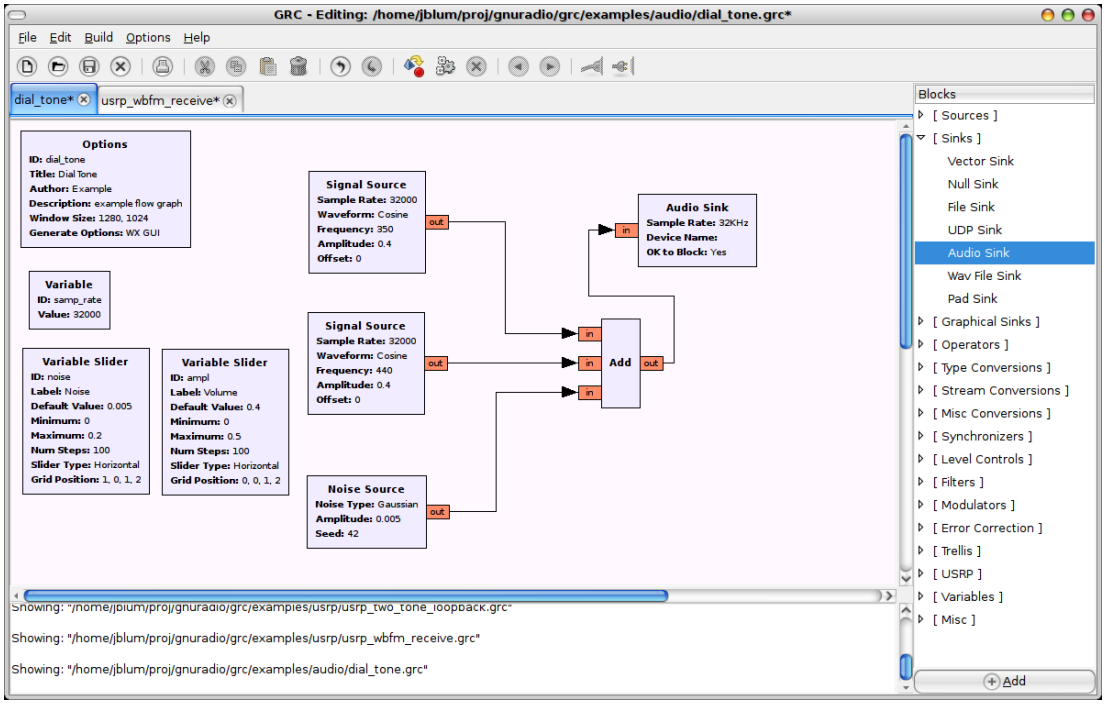

Figure 8: GNU Radio Companion interface

The interface has four important subsections: the work area where the flowgraph is build, the list of function blocks on the right, the tool bar on the top and a text window below. Users can drag and drop GNU Radio blocks from the list into the work area. The block list contains subsections for different kind of functions and new groups can be make and add. The list

count more than 150 blocks [29]. GRC represents signal blocks as colored, rectangular blocks. Each block has a label indicating the name of the block and a list of parameters, and has a number of sockets, input and output, that depends by the function. GRC represents a socket as a small rectangle attached to the graphical signal block. The socket has a label indicating its function usually named "in" or "out". Sockets are also are colored to indicate their data type: blue for complex, red for float, green for int, yellow for short and purple for byte. Drawing a line between two sockets with the same label a connection is made. If the flow graph is valid, all parameters are valid and all sockets are connected, it's possible to run the script from the tool bar or press F5. The flow graph is saved in an XML (eXtensible Markup Language) format, and a window will appear with any sliders or graphs that were added. To stop the flow graph you can close the window, press stop in the tool bar, or press F7. You can check the running state or errors in the text subsection below the work area. In any case, it's possible to check the equivalent python code for the flow graph that GRC makes when the script is executed.
GNU Radio doesn't have a dedicated hardware support, but its interface with the real world is USRP. Actually USRP and USRP2 exist, but in this work only USRP2 (Figure 9) is used because it is a new device.

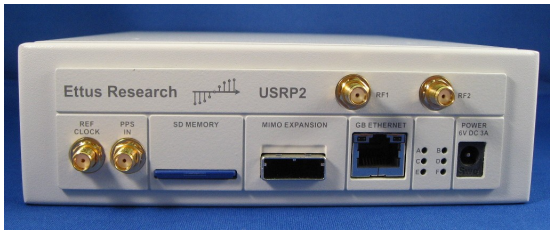

Figure 9: USRP2

USRP2 is designed to allow general purpose computers to function as high bandwidth software radios. In essence, it serves as a digital baseband and RF section of a radio communication system. The USRP2 is an integrated board which incorporates AD/DA converters, some forms of RF front end, and an

FPGA [30]. The basic design philosophy behind the USRP2 has been to do all of the waveform-specific processing, like modulation and demodulation, on the host CPU, while all of the high-speed general purpose operations like digital up and down conversion, decimation and interpolation are done on the FPGA. A typical setup of the USRP2 board consists of one mother board and up to two daughter boards, as shown in Figure 10.

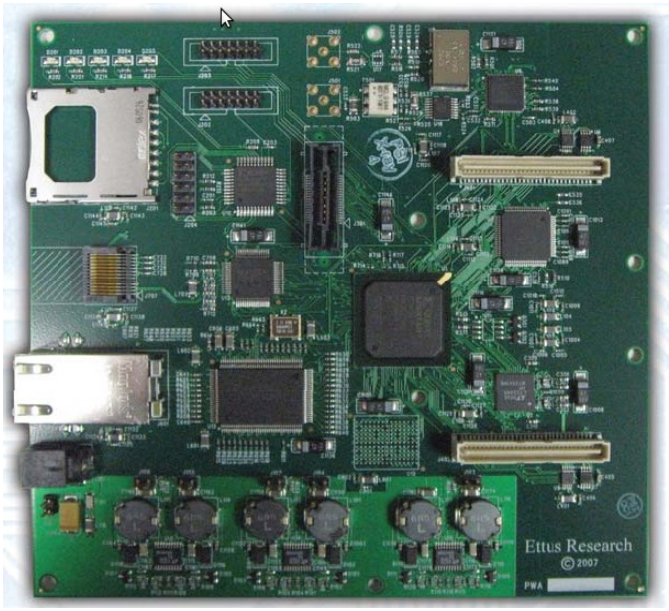

Figure 10: FPGA of the USRP2

There are two 14-bit AD converters LTC2284 with 100 MS/s and two 16-bit DA converters AD9777[29][35] with 400 MS/s. On the mother board, we can see the DC power input and the Gigabit Ethernet interface that allows applications to simultaneously send 50 MHz of RF bandwidth in and out of the USRP2. Two onboard digital downconverters (DDCs) mix, filter, and decimate (from 100 MS/s) incoming signals in the FPGA Spartan3 2000, XC3S2000-5FGG456 [29][36]. Two digital upconverters (DUCs) interpolate baseband signals to 100 MS/s before translating them to the selected output frequency. Daughterboards mounted on the USRP2 provide flexible, fully integrated RF front-ends. A wide variety of available daughterboards allows you to use different frequencies for a broad range of applications [30]. The

expansion MIMO port allows multiple USRP2 systems: it's possible to connect together more USRP2 to form fully coherent multiple antenna systems for MIMO with as many as 8 antennas. The entire USRP2 design is open source, including schematics, firmware, drivers, and even the FPGA and daughterboard designs. The USRP2 is supported on Linux, but that drivers for Windows, Mac OS X, and other systems will be developed soon.

4.3 Detector block

In this section the code for GNU Radio detector block is described. Actually, two subsections are predicted: the C++ code that implements the GNU Radio function and the XML code to import the block in GRC. Writing a new signal processing block involves creating 3 files: The *.h* and *.cc* files that define the new class and the *.i* file that tells Simplified Wrapper Interface Generator (SWIG) how to generate the glue that binds the class into Python. SWIG is an immensely useful program that allows you to interface many different interpreted languages with C/C++ code, for purposes of extension or interoperability between languages. Many C/C++ libraries have been ported to Python using this tool, and it allows anyone interested to actually write there own low level code and call it from within Python [37]. The standard procedures to make a new processing block are explained in detail by Eric Blossom in the official web site [38] so, in this chapter, only the detector code will be presented, while the complete code is attached in appendix A.

Every block in GRC has a corresponding XML file that contains parameters, IO ports, and a template for code generation. The guidelines to make it are on the official GNU Radio web site [29], but the main features will be described in the subsection below.

4.3.1 Detector Code

The detector code is in the file named *howto_detect_ff.cc* where the constructor of the class "detect" is defined. To implement the SVD algorithm the GSL (GNU Scientific Library) libraries are imported. GSL is a numerical library for C and C++ programmers. It is free software under the GNU General Public License like GNU Radio [39] so fully compatible. To simplify the code reading, three auxiliary functions are made: `TracyWidom` give the CDF value for the `pfa` (probability of false alarm P_{fa}) in input; `gamma` calculates the threshold for the test with in input the number of samples (see section 3.3.4), the length of the correlation function and the `pfa` value; `test` execute the detection test using the eigenvalues ratio and the threshold value. The prototypes are reported below and the complete functions are in appendix A.

```
1  float test (float ratio, float t)       // test function
2  float gamma (int ns, int L, float pfa)  // threshold function
3  float TracyWidom (float pfa)            // CDA function
```

The real code for the detection is the following:

```
1. int howto_detect_ff::general_work (int noutput_items,
2.                            gr_vector_int &ninput_items,
3.                            gr_vector_const_void_star &input_items,
4.                            gr_vector_void_star &output_items)
5. {
6.   const float *in = (const float *) input_items[0];
7.   float *out = (float *) output_items[0];
8.   float vett [d_samples];
9.   int lenght = floor(d_samples / d_L) * d_L;
10.  int p, j, k;
11.  float thr, lmax, lmin, ratio, TW;
12.  gsl_matrix * hankel = gsl_matrix_alloc (lenght,d_L);
13.  gsl_matrix * V = gsl_matrix_alloc (d_L,d_L);
14.  gsl_vector * S = gsl_vector_alloc (d_L);
15.  gsl_vector * temp = gsl_vector_alloc (d_L);
16.  FILE *story;
17.
18.  k=0; ratio = -1;
19.  gsl_matrix_set_zero (hankel);
20.  TW = TracyWidom (d_pfa);
21.  thr = gamma(lenght, d_L, TW);
22.
23.  for (int i = 0; i < noutput_items; i++){
24.
```

```
25.     vett[k]=in[i];
26.     k++;
27.
28.     if (k >= lenght){
29.       k = 0;
30.       story = fopen("filestory.txt", "a");
31.
32.       for( p=0; p<lenght; p++ ){
33.         for( j=0; j<d_L; j++ ){
34.           gsl_matrix_set (hankel, p, j, vett[p+j]);
35.         }
36.       }
37.
38.      gsl_linalg_SV_decomp (hankel, V, S, temp);
39.      lmax = gsl_vector_get(S, 0);
40.      lmin = gsl_vector_get(S, (d_L -1));
41.      ratio = lmax/lmin;
42.      mem = test(ratio, thr);
43.
44.      fprintf(story, "%f\n", mem);
45.      fclose(story);
46.
47.     }
48.
49.     out[i] =  mem;
50.   }
51.
52.   // Tell runtime system how many input items we consumed on
53.   // each input stream.
54.
55.   consume_each (noutput_items);
56.
57.   // Tell runtime system how many output items we produced.
58.   return noutput_items;
59. }
```

The processing block elaborates streams of samples, so it needs of a source and a sink for the samples. In this function they are the `&input_items` and `&output_items` respectively. Others parameters are the number of input items and of output items. The last one represents the buffer capacity and it depends by the RAM state of the PC host. In the variables definition, `d_pfa`, `d_sample` and `d_L` are the option inputs set by the user. These variables, defined in *howto_detect_ff.h,* are the P_{fa} value, the number of samples N used for the detection and the length of the correlation function L respectively (see Section 3.3). To use the GSL function, special matrix and vectors for computing the SVD algorithm are defined (rows 12-15). Finally a

FILE reference is declared for saving the detection results in a mass storage (row 16). The function sets the default values for the Hankel matrix and calculates the threshold for the test (rows 19-21). Then a for cycle starts for scanning the samples. This flow control is common to many GNU Radio functions and the main instructions should be included in it. Every sample is poured from the stream source (`&input_items`) in the vector `vett` to reach the length defined in row 9. Using this vector the Hankel matrix is made, the SVD is computed and the eigenvalues ratio is passed to the test function (rows 28-47). To learn more informations and documentations about the GSL functions you can see the official web site [39]. The detection result is 1 if the signal is detected, 0 otherwise. The is stored in the global variable `mem` and saved in a file. Then the vector is filled again of `length` samples before to make the follow decision. To guarantee the flow of the stream through the processing block, for every item in input, an item in output is required. In this case the output is the `mem` value. This mean that between two consecutive decisions the `mem` value is unchanged. The last rows (52-59) are system informations to know how many input items and output items are produced.

4.3.2 XML code

To import the processing block in GRC a XML script is necessary. Making the file in *.grc_gnuradio* folder, a new group of functions, named CR, is added to the list of GRC. In this group there is the detector block. The XML code, showed below, allows to define the user options like the number of samples to process, the length of the covariance function and the probability of false alarm.

```
<block>
        <name>Detector</name>
        <key>detector</key>
        <category>CR</category>
        <import>from gnuradio import gr</import>
        <import>from gnuradio import howto</import>
        <make>howto.detect_ff($pfa, $len.elle, $samples)</make>

        <param>
                <name>pfa</name>
                <key>pfa</key>
                <value>0.1</value>
                <type>real</type>
        </param>
        <param>
                <name>L</name>
                <key>len</key>
                <type>enum</type>
                <option>
                        <name>12</name>
                        <key>a</key>
                        <opt>elle:12</opt>
                </option>
                <option>
                        <name>16</name>
                        <key>b</key>
                        <opt>elle:16</opt>
                </option>
                <option>
                        <name>20</name>
                        <key>c</key>
                        <opt>elle:20</opt>
                </option>
        </param>
        <param>
                <name>Samples</name>
                <key>samples</key>
                <value>1000</value>
                <type>int</type>

        <sink>
                <name>in</name>
                <type>float</type>
        </sink>
        <source>
                <name>out</name>
                <type>float</type>
        </source>
</block>
```

In the rows 5-7, the new GNU Radio libraries are imported and the function *detect_ff* is called. From row 9 to row 39 the options for the parameters are inserted in the GRC block and the `pfa` is 0.1 for default; from row 41 to 49 the types of input and output are defined. In this way the detector block (Figure 11) has one socket for real input and a socket for real output and, with a click on the block, the user can chose and set the main detection

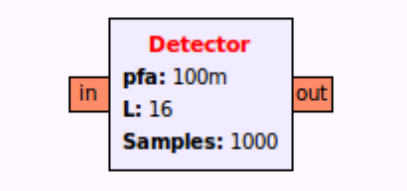

Figure 11: Detector processing block in GRC

parameters. Obviously `pfa` is a value between 0 and 1, but if the user sets a wrong number the probability of false alarm will be arranged to the default value of 0.1. The parameter `L`, the length of the correlation function, can have three different values: 12, 16, 20. Finally, the user can set a integer number of samples to elaborate.

Chapter 5

Detection tests

5.1 Overview

After the presentation of the detection method and of how it can be implemented in GNU Radio environment, the experimental validation of the detector is necessary. Thus, in this chapter, the procedures of the detection tests are proposed. The target of the tests is to verify the capability of the device to detect the typical modulated signals. These tests are divided in two sections. The first part shows the results of detections made in Matlab environment with simulated signals. The second section describes the testbed used to make a real wireless communication and the results of detections of real signals. The Matlab simulations are necessary to compare the detection results in real world. These sections are purely illustrative. Observations and comments about the results are presented in the next chapter.

5.2 Matlab tests

In this section the procedure and the results of detection in Matlab environment are presented. As already explained, the detection method is based on the ratio of the maximum eigenvalue to the minimum eigenvalue of the covariance matrix of the received signal, and the method is improved using the SVD algorithm to estimate the singular values of the Hankel matrix filled by received signal samples. The eigenvalues of the covariance matrix are equals to the square of the singular values (see Section 3.3.3).

The tests have been implemented using Matlab 7.8.0 (R2009a) and the scripts are in appendix B. The modulations used are: 2PSK, 4PSK, 8PSK, 16QAM, 64QAM. A shaping pulse can be applied to increase the performances. White noise is added to the signals and its effect is expressed by SNR. The SNR range includes negative values: 0dB, -5dB, -10dB, -12dB. The parameters of the simulation are: 1kHz carrier frequency, 10kHz sample frequency, 1000 symbols/s, 1000 samples analyzed for each signal, the P_{fa} is 0.1 and the length of the covariance function is L=16. To explain how the tests are made, the procedure for testing PSK modulations is presented:

```
1. %DETECTOR TEST : matlab simulation of main signal modulations
2.
3. clear all
4. close all
5. clc
6.
7. %-data
8. nt = 100; number of signals under test
9. %shape = 2:3; %1= rectangular, 2= Raised Cosine, 3 = Root Raised
   Cosine
10. SNR = [0, -5, -10, -12];
11. L = 16;
12. pfa = 0.1;
13. results = SNR;
14. row = 2;
15.
16. %-Test PSK
17. M = [2,4,8];
18. for i = 1:3,                %number of levels
19.     for shape = 1:3,        %shape
20.         for k = 1:4,        %SNR in dB
21.             ndetect = 0;
```

```
22.             for n = 1:nt,   %n. tests
23.                 sign = Mpsk(M(i), SNR(k), 1);
24.                 ndetect = ndetect + detect_counter(sign, L, pfa);
25.             end
26.             results(row,k) = ndetect;
27.         end
28.         row = row+1;
29.     end
30. end
```

It contains the functions to generate the modulated signals (row 23), and the functions for the detection (row 24). This is very similar to the function presented in section 4.3.1. The signal `sign` is tested using a threshold calculated with `pfa`, and `L`. Also in this case `pfa` is the P_{fa} and `L` is the length of the correlation function (see Section 3.3.2). In the detection tests, for each modulation, 100 signals are simulated and analyzed by the detector. The function `results` (row 26) fills the Table 2 with the number of correct decisions on 100 detections for different SNR values.

	0 dB	-5 dB	-10 dB	-12 dB
2PSK	100	100	71	29
2PSKrc	100	100	88	58
2PSKrrc	100	100	85	65
4PSK	100	100	65	40
4PSKrc	100	100	85	56
4PSKrrc	100	100	87	53
8PSK	100	100	61	35
8PSKrc	100	100	89	62
8PSKrrc	100	100	83	47
16QAM	100	100	67	40
16QAMrc	100	100	82	50
16QAMrrc	100	100	86	52
64QAM	100	100	70	38
64QAMrc	100	100	89	52
64QAMrcc	100	100	79	54

Table 2: Detection Test Results. The signals generated are 100 for each modulation; 'rc' and 'rrc' are Raised Cosine and Root Raised Cosine with roll-off 0.35 respectively; L = 1000; pfa = 0,1.

The performances can grow if the number of samples is bigger. The detection test is repeated using 1500 and 2000 samples with the other parameters unchanged. The results are reported in Table 3 and Table 4 respectively.

	0 dB	-5 dB	-10 dB	-12 dB
2PSK	100	100	80	53
2PSKrc	100	100	89	80
2PSKrrc	100	100	96	86
4PSK	100	100	82	55
4PSKrc	100	100	97	68
4PSKrrc	100	100	91	66
8PSK	100	100	83	58
8PSKrc	100	100	98	82
8PSKrrc	100	100	96	72
16QAM	100	100	86	56
16QAMrc	100	100	98	76
16QAMrrc	100	100	95	76
64QAM	100	100	82	57
64QAMrc	100	100	100	72
64QAMrcc	100	100	94	68

Table 3: Detection Test Results. The signals generated are 100 for each modulation; 'rc' and 'rrc' are Raised Cosine and Root Raised Cosine with roll-off 0.35 respectively; L = 1500; pfa = 0,1

	0 dB	-5 dB	-10 dB	-12 dB
2PSK	100	100	95	60
2PSKrc	100	100	100	89
2PSKrrc	100	100	98	89
4PSK	100	100	94	60
4PSKrc	100	100	99	89
4PSKrrc	100	100	99	79
8PSK	100	100	94	60
8PSKrc	100	100	100	86
8PSKrrc	100	100	99	86
16QAM	100	100	95	64
16QAMrc	100	100	100	85
16QAMrrc	100	100	98	90
64QAM	100	100	93	68
64QAMrc	100	100	99	88
64QAMrcc	100	100	100	75

Table 4: Detection Test Results. The signals generated are 100 for each modulation; 'rc' and 'rrc' are Raised Cosine and Root Raised Cosine with roll-off 0.35 respectively; L = 2000; pfa = 0,1

5.3 Detection test in the real world

In this section, the experimental activity in the real world is described. The testbed, the test procedure and the results are presented. The detection tests on real signals have been executed making a wireless communication between

two USRP2. The testbed is showed in Figure 12. It's composed by two USRP2 spaced 30 cm and two twins computers to control the devices. The antennas are in line of sight.

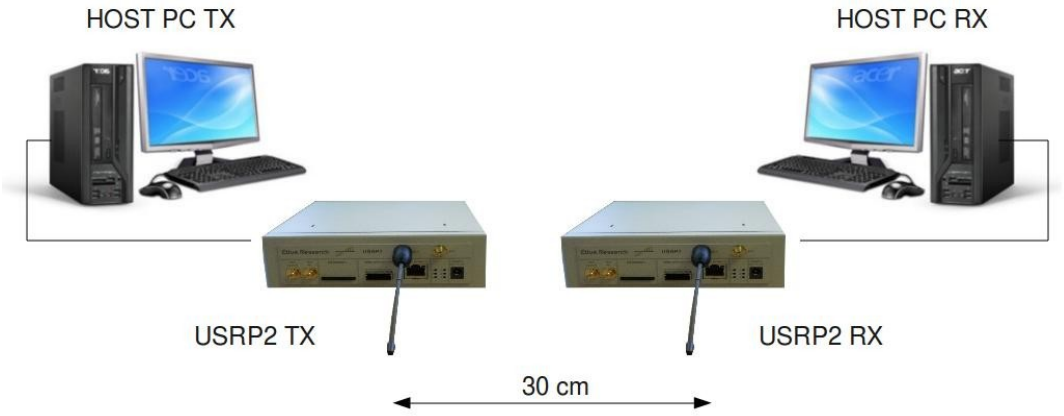

Figure 12: TESTBED for detection in real world

The daughterboard mounted on each USRP2 is RFX400. It works in the RF band of 100 MHz, from 400 MHz to 500 MHz, in transmission as well as in reception. The main features are: 30 MHz transmit and receive bandwidth, full control via software or FPGA, independent Local Oscillator (LOs) for transceiver and receiver, 70 dB of Automatic Gain Control (AGC) range. On the GNU Radio and Ettus web site you can find more technical information and schematics about this board[29][30]. The Antennas used on the USRP2 are VERT400[30] able to work with RFX400. Each USRP2 is connected to own computer by ethernet cable. The computers have a processor Pentium Dual-Core 2.70GHz and 2GB of RAM. The operative system used is the Linux distribution Ubuntu 9.10. The GNU Radio release used is 3.2.2.

To make the link two Python scripts are necessary: one for the USRP2 transceiver and one for the USRP2 receiver. These scripts control the devices and set the parameters to make the detection test. The scripts have written using GRC. The transceiver script (Figure 13) generates the baseband signal under test and uses the USRP2 to send the samples in RF band at 450MHz

with an interpolation index of 512. This script implements the typical I-Q modulator with the currier frequency of 100 Hz, amplitude 1V, and a sample frequency of 1000 Hz. The source is random so the symbols are independent. Like in simulation, the modulations generated are 2PSK, 4PSK, 8PSK, 16QAM and 64QAM, but the filters are not applied.

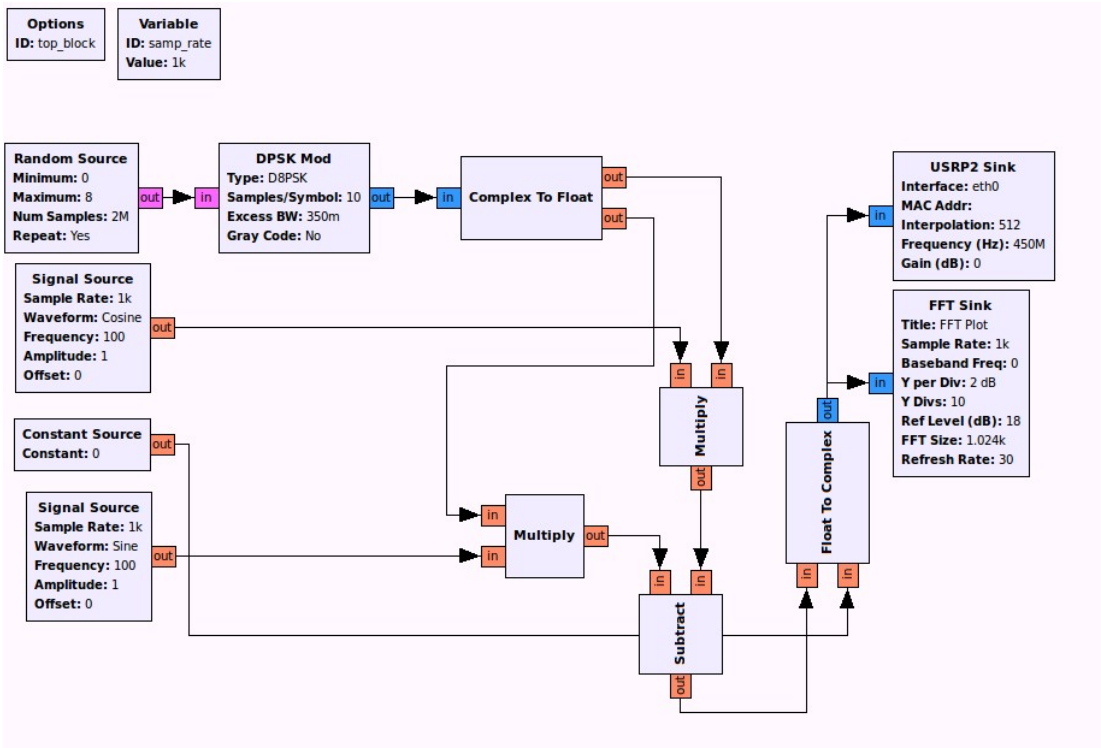

Figure 13: Transceiver python script in GRC

The signals are real, but the processing block USRP2 needs complex data in input, so a format converter is used. In the receiver script (Figure 14) the source is the USRP2 with a decimation factor of 100. In this test, the receiver is tuned at same RF frequency of the transceiver (450MHz). A format converter is necessary to use a real signal. This is important because in the detection we may use the first order Tracy Widom function to set the threshold (see Section 3.3.4). Moreover, the processing block to add Gaussian noise appears to simulate different SNRs. Artificial noise is added in the receiver script to consider the degrading effects of channel and hardware. The gray blocks in Figure 14 has been used to estimate the RMS value of the

received signal. Because this value is fluctuating, the mean is considered. Using the RMS value of the signal and the SNR definition, it's possible to calculate the `amplitude` (RMS value) of the noise block to simulate the SNR expected. The SNR values in the detection test are the same of the Matlab tests. Unlike the simulations, for each modulation only a signal is generated. It contains 2000000 samples, so analyzing 1000 samples at a time the detector can make 2000 decisions.

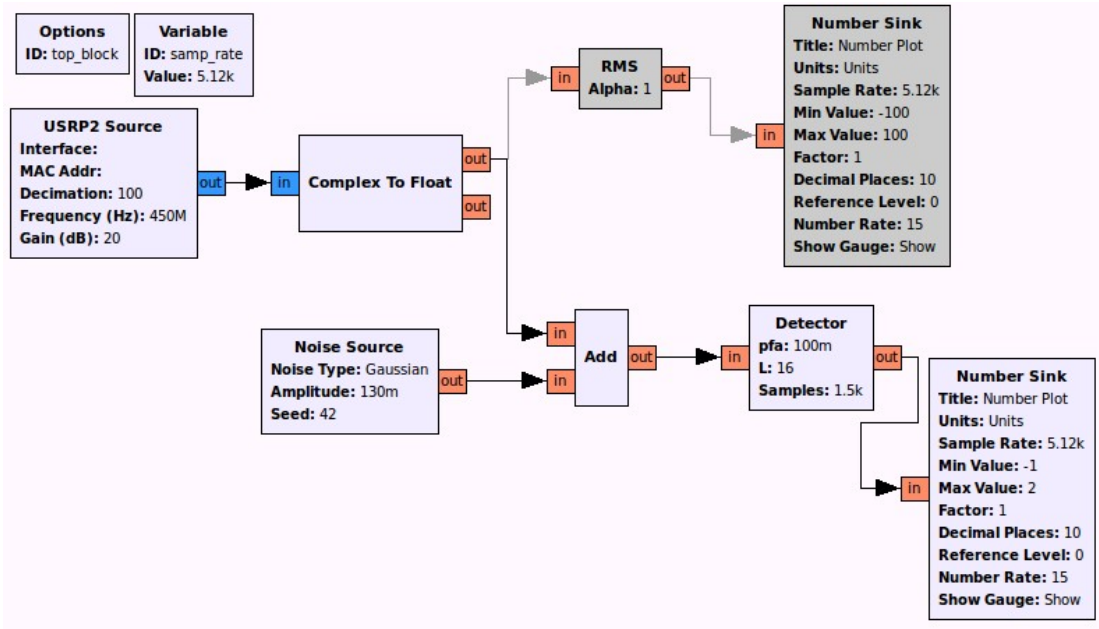

Figure 14: Receiver python script in GRC

Because the decisions are independent (can you see the Chapter 4), to have the same statistics of the Matlab tests, the first consecutive 100 detections can be considered. Thus, the received signal is degraded by additive noise and given to the detector block that fills a file of the decision results. These are a sequence of 0 (fail detection) or 1 (correct detection), so it's easy to count the number of detections. The tests have made with different numbers of examined samples: 1000, 1500 and 2000. The results are presented in the tables below. Each table show the number of detection on 100 decisions for different SNR values with L = 16 and $P_{fa} = 0.1$.

	0 dB	-5 dB	-10 dB	-12 dB
2PSK	100	100	72	19
4PSK	100	100	80	15
8PSK	100	100	75	23
16QAM	100	100	75	29
64QAM	100	100	73	20

Table 5: Detection test results on 100 detection for each modulation; L = 1000; P_{fa} = 0,1; the signals are not filtred;

	0 dB	-5 dB	-10 dB	-12 dB
2PSK	100	100	77	53
4PSK	100	100	91	52
8PSK	100	100	95	58
16QAM	100	100	99	63
64QAM	100	100	87	51

Table 6: Detection test results on 100 detection for each modulation; L = 1500; P_{fa} = 0,1; the signals are not filtred

	0 dB	-5 dB	-10 dB	-12 dB
2PSK	100	100	97	79
4PSK	100	100	99	75
8PSK	100	100	100	76
16QAM	100	100	100	75
64QAM	100	100	100	80

Table 7: Detection test results on 100 detection for each modulation; L =2000; P_{fa} = 0,1; the signals are not filtred

Chapter 6

Results analysis

6.1 Comparison

Comparing the tests results it's possible to describe the behavior of the proposed detector. The comparison is made plotting the data of the tables (Table 2, Table 3, Table 4, Table 5, Table 6, Table 7) in some graphs. They express the differences between the number of the detections on real signals and the number of the detections on simulated signals. On the x-axes there are the SNR values used in the tests. The graph has three curves that represent the detections with 1000 (blue), 1500 (red) and 2000 (yellow) samples. Zero value is the expected result. It means the behavior of the real detector is the same of the simulated detector. If the curves are upper zero the real detector finds more signals than in simulation. When the curves are under zero the real detector performances are worse than in simulation. For each modulation a graph has been made. The graphs below show the comparisons of the tests without filtered signals because these are the only comparable results.

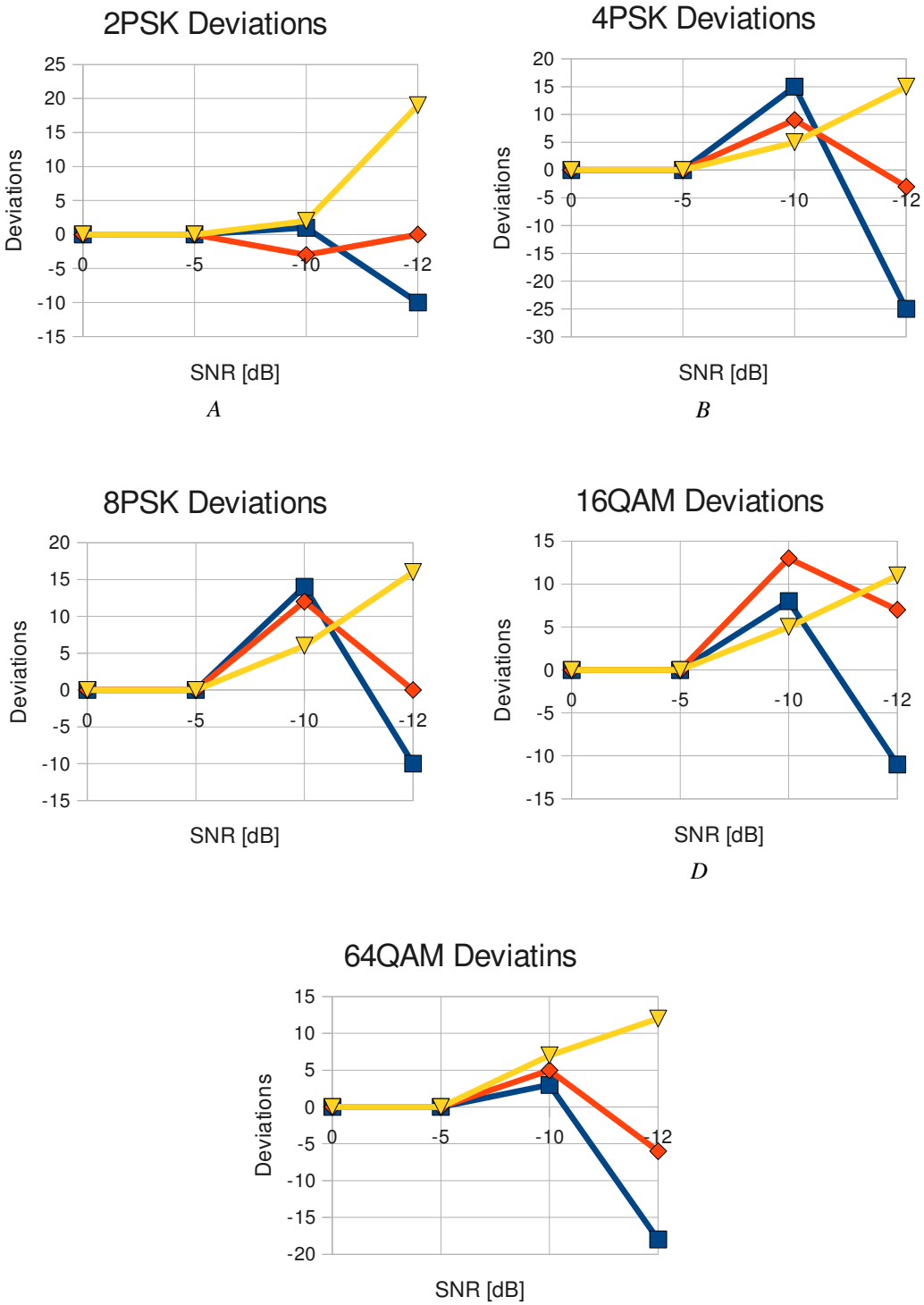

Figure 15: Comparison Graphs. 2PSK (A), 4PSK (B), 8PSK (C), 16 QAM (D), 64 QAM (E)

The Table 2 Table 3 Table 4 show an important propriety of the detector. In simulation, the number of correct detections is higher when the SNR value grows. Because the curves in Figure 15 go to zero when the SNR value decreases, the propriety is verified also for the proposed signal detector.

There are some differences when SNR is low. In simulation, with SNR of -12dB the capabilities of detection are low and very variable. This is true for the proposed detector too. In adding, unlike in simulation environment, the noise is not ideally constant. Its fluctuations alter the results in extreme work conditions. It means that, specially for longer observations (more samples kept), the probability to detect with lower noise power increases. In any case, the deviations are not so high and, for SNR low, the differences are not higher than 15% in mean. Thus in general the performances of the proposed detector and of the simulated detector are close.

The test with filtered signals has not been made because, at time, there aren't the appropriate GNU Radio functions. However the Matlab results in tables Table 2 Table 3 Table 4 show that the detection performances grow if a numerical elaboration is applied to the signals under test. Using an opportune filter on the signals, the detection performances of the proposed device can improve.

Chapter 7

Conclusions

The target of this work is to contribute to the develop of full CR presenting a prototype of signal detector implemented in GNU Radio environment. The research and prototypical activities have been made in the swedish Ericsson laboratory of Gavle in collaboration with the technical support offered by LESIM of the University of Sannio, Benevento (Italy). They are interested to develop new technologies to improve the spectrum efficiency introducing the new cognitive radio system. The importance of CR to assist the spectrum management has been explained in the first chapter. CR approach and architecture have been described in the second chapter. SDR for CR has been introduced and how it can perform a new generation radio system has been explained. A particular application of SDR is the signal detector. It is necessary for analyzing the radio spectrum and to find signals in it. In the third chapter, the theoretical detection method has been discussed in detail. The detection algorithm is eigenvalue-based and the SVD is used to estimate the eigenvalues of the samples covariance matrix. The method has been compared with others algorithms, but it has been chosen because it doesn't

need knowledge a-priori and because it has an high noise rejection. GNU Radio project has been presented in the fourth chapter. GNU Radio is a software platform that respects the CR approach. It can implement a real radio system using a common computer and a minimal hardware, the USRP2, to transmit the signal in RF frequencies. The signal detector has been implemented in GNU Radio adding a new processing block to the default library. C++ code and XML code of the detector are also presented in this chapter. The detector has been tested by means of several modulated signals. In order, simulation trials have been made to compare the results with the tests in the real world. The fifth chapter introduces the testbed, reports the test procedures and shows the results. Comments about the experimental results are in the sixth chapter. Using no high quality devices, the signal detector has been tested in not ideal conditions so the results are different from the simulation performances. However, it's fair to say that the detector works well because the deviations between simulation results and experimental results are not so high.

The prototype of signal detector proposed is an further step to make a full CR. Using GNU Radio platform, future works involve the optimization of the C++ code for a real-time detection, the implementation of specific functions for making tests more elaborate, the study and the characterization of the USRP2 performances. Finally, the realization of a complete radio system application should be made adding, for example, a function for the signal classifier to GNU Radio library.

Appendix A: C++ code

Here the main files to import the signal detector in GNU Radio (see Section 4.3.1) are presented.

-howto_detect_ff.h:

```
/* -*- c++ -*- */

#ifndef INCLUDED_HOWTO_DETECT_FF_H
#define INCLUDED_HOWTO_DETECT_FF_H

#include <gr_block.h>

class howto_detect_ff;

/*
 * We use boost::shared_ptr's instead of raw pointers for all access
 * to gr_blocks (and many other data structures).  The shared_ptr gets
 * us transparent reference counting, which greatly simplifies storage
 * management issues.  This is especially helpful in our hybrid
 * C++ / Python system.
 *
 * See http://www.boost.org/libs/smart_ptr/smart_ptr.htm
 *
 * As a convention, the _sptr suffix indicates a boost::shared_ptr
 */
typedef boost::shared_ptr<howto_detect_ff> howto_detect_ff_sptr;

/*!
 * \brief Return a shared_ptr to a new instance of howto_square_ff.
 *
 * To avoid accidental use of raw pointers, howto_detect_ff's
 * constructor is private.  howto_make_detect_ff is the public
 * interface for creating new instances.
 */
howto_detect_ff_sptr howto_make_detect_ff (float pfa, int L, int samples);

class howto_detect_ff : public gr_block
{
private:
  // The friend declaration allows howto_make_square_ff to
  // access the private constructor.

  friend howto_detect_ff_sptr howto_make_detect_ff (float pfa, int L, int samples);

  howto_detect_ff (float pfa, int L, int samples);         // private constructor
  float d_pfa; int d_L; int d_samples;

 public:
  ~howto_detect_ff ();// public destructor
```

```
  void set_pfa(float input_a) { d_pfa = input_a; }
  int get_pfa() { return d_pfa; }
  void set_L(int input_b) { d_L = input_b; }
  int get_L() { return d_L; }
  void set_samples(int input_c) { d_samples = input_c; }
  int get_samples() { return d_samples; }

  int general_work (int noutput_items,
                    gr_vector_int &ninput_items,
                    gr_vector_const_void_star &input_items,
                    gr_vector_void_star &output_items);

  // Where all the action really happens

};

#endif /* INCLUDED_HOWTO_DETECT_FF_H */
```

-howto_detect_ff.h:

```
/* -*- c++ -*- */

/*
 * config.h is generated by configure.  It contains the results
 * of probing for features, options etc.  It should be the first
 * file included in your .cc file.
 */
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <howto_detect_ff.h>
#include <stdio.h>
#include <gr_io_signature.h>
#include <math.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_linalg.h>

/*
 * Create a new instance of howto_square_ff and return
 * a boost shared_ptr.  This is effectively the public constructor.
 */
howto_detect_ff_sptr
howto_make_detect_ff (float pfa, int L, int samples)
{
  return howto_detect_ff_sptr (new howto_detect_ff (pfa, L, samples));
}

/*
 * Specify constraints on number of input and output streams.
 * This info is used to construct the input and output signatures
 * (2nd & 3rd args to gr_block's constructor).  The input and
 * output signatures are used by the runtime system to
 * check that a valid number and type of inputs and outputs
 * are connected to this block.  In this case, we accept
 * only 1 input and 1 output.
 */
static const int MIN_IN = 1;  // mininum number of input streams
static const int MAX_IN = 1;  // maximum number of input streams
```

```
static const int MIN_OUT = 1; // minimum number of output streams
static const int MAX_OUT = 1; // maximum number of output streams

float mem = 0; //Global Variable

/*
 * The private constructor
 */
howto_detect_ff::howto_detect_ff (float pfa, int L, int samples)
  : gr_block ("detect_ff",
              gr_make_io_signature (MIN_IN, MAX_IN, sizeof (float)),
              gr_make_io_signature (MIN_OUT, MAX_OUT, sizeof (float))),
    d_pfa(pfa), d_L(L), d_samples(samples)
{
  // nothing else required in this example
}

/*
 * Our virtual destructor.
 */
howto_detect_ff::~howto_detect_ff ()
{
  // nothing else required in this example
}

//------------functions---------------


/*This function gives the CDF value for the pfa in input*/

float TracyWidom (float p){

  float pd, tw;

  tw = 0.45;
  pd = 1 - p;
  if (pd >= 0.01 && pd <= 0.05){
    tw = 18*(pd - (17/75)); printf("a - %f\n", tw);
  }else if (pd >= 0.05 && pd <= 0.1){
    tw = 8*(pd - (179/400)); printf("b - %f\n", tw);
  }else if (pd >= 0.1 && pd <= 0.3){
    tw = (87/20)*(pd - (643/870)); printf("c - %f\n", tw);
  }else if (pd >= 0.3 && pd <= 0.5){
    tw = (16/5)*(pd - (287/320)); printf("d - %f\n", tw);
  }else if (pd >= 0.5 && pd <= 0.7){
    tw = (17/5)*(pd - (297/340)); printf("e - %f\n", tw);
  }else if (pd >= 0.7 && pd <= 0.9){
    tw = (5.2)*(pd - (0.813)); printf("f - %f\n", tw);
  }else if (pd >= 0.9 && pd <= 0.95){
    tw = (53/5)*(pd - (909/1060)); printf("g - %f\n", tw);
  }else if (pd >= 0.95 && pd <= 1){
    tw = 26*(pd - (593/650)); printf("h - %f\n", tw);
  }else{
    printf ("wrong pfa value: it must be between 0 and 1\n");
    printf ("the pfa value standard is 0.1\n");
    tw = (53/5)*(0.9 - (909/1060));
  }
  return tw;

}
```

```
/*this function calculates the threshold for the test
the inputs are: ns = numbers of samples;
L = length of the correlation function;
pfa = probability of false alarm*/

float gamma (int ns, int L, float pfa){
  float A, B, C, ratio1, ratio2, g;

  A = sqrt(ns)+sqrt(L);
  B = sqrt(ns)-sqrt(L);
  C = ns*L;
  ratio1 = pow(A,2)/pow(B,2);
  ratio2 = 1+( pow(A,-0.667) / pow(C,0.167) )*pfa;
  g = ratio1*ratio2;

  return g;
}

/*This function makes the detection test*/

float test (float r, float t){

  float decision;
  if(r > -1){
    if(r <= t){
      decision = 0;
    }else{
      decision = 1;
    }
  }
return decision;}


//-------------end functions-----------

int
howto_detect_ff::general_work (int noutput_items,
                               gr_vector_int &ninput_items,
                               gr_vector_const_void_star &input_items,
                               gr_vector_void_star &output_items)
{
  const float *in = (const float *) input_items[0];
  float *out = (float *) output_items[0];
  float vett [d_samples];
  int lenght = floor(d_samples / d_L) * d_L;
  int p, j, k;
  float thr, lmax, lmin, ratio, TW;
 // char c[1];
  gsl_matrix * hankel = gsl_matrix_alloc (lenght,d_L);
  gsl_matrix * V = gsl_matrix_alloc (d_L,d_L);
  gsl_vector * S = gsl_vector_alloc (d_L);
  gsl_vector * temp = gsl_vector_alloc (d_L);
  FILE *story;

  k=0; ratio = -1;
  gsl_matrix_set_zero (hankel);
  TW = TracyWidom (d_pfa);
  thr = gamma(lenght, d_L, TW);
```

```
  for (int i = 0; i < noutput_items; i++){

    vett[k]=in[i];
    k++;

    if (k >= lenght){
      k = 0;
      story = fopen("filestory.txt", "a");

      for( p=0; p<lenght; p++ ){
        for( j=0; j<d_L; j++ ){
          gsl_matrix_set (hankel, p, j, vett[p+j]);
        }
      }

     gsl_linalg_SV_decomp (hankel, V, S, temp);
     lmax = gsl_vector_get(S, 0);
     lmin = gsl_vector_get(S, (d_L -1));
     ratio = lmax/lmin;
     mem = test(ratio, thr);

     fprintf(story, "%f - ratio=%f - soglia=%f\n ", mem, ratio, thr);
     fclose(story);

    }

    out[i] =  mem;
  }

  // Tell runtime system how many input items we consumed on
  // each input stream.

  consume_each (noutput_items);

  // Tell runtime system how many output items we produced.
  return noutput_items;
}
```

-howto.i:

```
/* -*- c++ -*- */

%include "gnuradio.i"                // the common stuff

%{
#include "howto_detect_ff.h"
%}

// ----------------------------------------------------------------

/*
 * First arg is the package prefix.
 * Second arg is the name of the class minus the prefix.
 *
 * This does some behind-the-scenes magic so we can
 * access howto_square_ff from python as howto.detect_ff
 */
GR_SWIG_BLOCK_MAGIC(howto,detect_ff);

howto_detect_ff_sptr howto_make_detect_ff (float pfa, int L, int samples);
```

```
class howto_detect_ff : public gr_block
{
private:
  howto_detect_ff (float pfa, int L, int samples);

public:

  void set_pfa(float input_a) { d_pfa = input_a; }
  int get_pfa() { return d_pfa; }
  void set_L(int input_b) { d_L = input_b; }
  int get_L() { return d_L; }
  void set_samples(int input_c) { d_samples = input_c; }
  int get_samples() { return d_samples; }
};
```

Appendix B: Matlab code

Here the M-files to make the detection tests in Matlab (see Section 4.3.1) are presented.

First the functions to generate the signals.

```
function psk = Mpsk(M, SNR, shape)

%data
fsy = 1000;     %symbol frequency
fc = 1000;      %currier frequency
fs = 10*fc;     %sample frequency
ns = 1000;      %number of samples

N = ceil(fsy*ns/fs);        %number of symbols

%psk parameters
if (M ~= 2) && (M ~= 4) && (M ~= 8),
    disp('ERROR: choose M = 2, M = 4, M = 8');
    psk = 0;
    return;
end

%modulation
B = 1:M;
m = B(randint(N,1,M)+1);

%signal
ps = ceil(ns/N);                        %number of pulse sample
time = 0:1/fs:(ns-1)/fs;                %time line

modulation = 2*pi*(m-1)/M;
pskI = cos(modulation);
pskQ = sin(modulation);

if shape == 1,

    pI = repmat(pskI, ps, 1);
    pI = pI(:);
    pQ = repmat(pskQ, ps, 1);
    pQ = pQ(:);
    psk = pI'.*cos(2*pi*fc*time)-pQ'.*sin(2*pi*fc*time);

elseif shape == 2,
    R = 0.35;
    aI = upsample(pskI,ps);
    aQ = upsample(pskQ,ps);

    sx1 = rcos(aI, 1/fsy, ps, R);
```

```
    sy1 = rcos(aQ, 1/fsy, ps, R);
    psk = sx1.*cos(2*pi*fc*time)-sy1.*sin(2*pi*fc*time);

elseif shape == 3
    R = 0.35;
    aI = upsample(pskI,ps);
    aQ = upsample(pskQ,ps);

    sx2 = rrcos(aI, 1/fsy, ps, R);
    sy2 = rrcos(aQ, 1/fsy, ps, R);
    psk = sx2.*cos(2*pi*fc*time)-sy2.*sin(2*pi*fc*time);

end

%add noise
noise = sqrt( var(psk) /(10^((SNR)/10)) ) * randn(1,ns); %noise in dBm
psk = psk + noise;

%plot(time,psk);

end


function qam = Mqam(M, SNR, shape)

%data
fsy = 1000;     %symbol frequency
fc = 1000;      %currier frequency
fs = 10*fc;     %sample frequency
ns = 1000;      %number of samples

N = ceil(fsy*ns/fs);        %number of symbols

%psk parameters
if (M ~= 16) && (M ~= 64) ,
    disp('ERROR: choose M = 16, M = 64');
    psk = 0;
    return;
end

%modulation
B = 1-sqrt(M):2:-1+sqrt(M);
m = B(randint(2,N,sqrt(M))+1);

%signal
ps = ns/N;                              %number of pulse sample
time = 0:1/fs:(ns-1)/fs;                %time line

qamI = m(1,:);
qamQ = m(2,:);

if shape == 1,
    Ic = repmat(qamI, ps, 1);
    Ic = Ic(:);
    Qc = repmat(qamQ, ps, 1);
    Qc = Qc(:);
    qam = Ic'.*cos(2*pi*fc*time)-Qc'.*sin(2*pi*fc*time);

elseif shape == 2,
    R = 0.35;
```

```
    aI = upsample(qamI,ps);
    aQ = upsample(qamQ,ps);

    sx1 = rcos(aI, 1/fsy, ps, R);
    sy1 = rcos(aQ, 1/fsy, ps, R);
    qam = sx1.*cos(2*pi*fc*time)-sy1.*sin(2*pi*fc*time);

elseif shape == 3,
    R = 0.35;
    aI = upsample(qamI,ps);
    aQ = upsample(qamQ,ps);

    sx2 = rrcos(aI, 1/fsy, ps, R);
    sy2 = rrcos(aQ, 1/fsy, ps, R);
    qam = sx2.*cos(2*pi*fc*time)-sy2.*sin(2*pi*fc*time);

end
%add noise
noise = sqrt( var(qam) /(10^((SNR)/10)) ) * randn(1,ns); %noise in dBm
qam = qam + noise;
```

It's needed implement the filter function for Raised Cosine and Root Raised Cosine. The inputs are the signal, the symbol time, the filter length and the roll-off value. The output is the filtered signal.

```
function y = rcos(sign, T, n, R)

t = linspace(-T, T, n);
rc = sinc(t/T).*cos(pi*R*t/T)./(1- (2*R*t/T).^2);

y = conv(sign, rc, 'same');


function y = rrcos(sign, T, n, R)
ts = 2*T/n;
t1 = -T/4/R;
t2 = -T/4/R+ts:ts:-ts;
t3 = 0;
t4 = ts:ts:T/4/R-ts;
t5 = T/4/R;
t = [t1 t2 t3 t4 t5];

rrc1 = 0;
rrc2   =   4*R*(   sin(pi*(1-R)*t2/T)*T./t2/R/4   +   cos(pi*t2*(1+R)/T)   )./(   (1   -
(4*R*t2/T).^2) )/pi/sqrt(T);
rrc3 = 4*R/pi/sqrt(T)*(1+(1-R)*pi/4/R);
rrc4   =   4*R*(   sin(pi*(1-R)*t4/T)*T./t4/R/4   +   cos(pi*t4*(1+R)/T)   )./(   (1   -
(4*R*t4/T).^2) )/pi/sqrt(T);
rrc5 = 0;

rrc = [rrc1 rrc2 rrc3 rrc4 rrc5];

y = conv(sign, rrc, 'same');
```

The procedure below implements the test. The detection function is `detection_counter`. In this function you can found the `TracyWidom` and `threshold` function.

```
%DETECTOR TEST : matlab simulation of main signal modulations

clear all
close all
clc

%-data
nt = 100;
%shape = 2:3; %1= rectangular, 2= Raised Cosine, 3 = Root Raised Cosine
SNR = [0, -5, -10, -12];
L = 16;
pfa = 0.1;
results = SNR;
row = 2;

% %-Test PSK
% M = [2,4,8];
% for i = 1:3,                %number of levels
%     for shape = 1:3,        %shape
%         for k = 1:4,        %SNR in dB
%             ndetect = 0;
%             for n = 1:nt,   %n. tests
%                 sign = Mpsk(M(i), SNR(k), shape);
%                 ndetect = ndetect + detect_counter(sign, L, pfa);
%             end
%             results(row,k) = ndetect;
%         end
%         row = row+1;
%     end
% end

% %-Test QAM
% M = [16,64];
% for i = 1:2,                %number of levels
%     for shape = 1:3,        %shape
%         for k = 1:4,        %SNR in dB
%             ndetect = 0;
%             for n = 1:nt,   %n. tests
%                 sign = Mqam(M(i), SNR(k), shape);
%                 ndetect = ndetect + detect_counter(sign, L, pfa);
%             end
%             results(row,k) = ndetect;
%         end
%         row = row+1;
%     end
% end
```

```
function F = TracyWidom(pfa)
pd = 1-pfa;
if pd >= 0.01 && pd <= 0.05
    F = 18*(pd-17/75);
elseif pd >= 0.05 && pd <= 0.1
    F = 8*(pd-179/400);
elseif pd >= 0.1 && pd <= 0.3
    F = (87/20)*(pd-643/870);
elseif pd >= 0.3 && pd <= 0.5
    F = (16/5)*(pd-287/320);
elseif pd >= 0.5 && pd <= 0.7
    F = (17/5)*(pd-297/340);
elseif pd >= 0.7 && pd <= 0.9
    F = (26/5)*(pd-423/520);
elseif pd >= 0.9 && pd <= 0.95
    F = (53/5)*(pd-909/1060);
elseif pd >= 0.95 && pd <= 1
    F = 26*(pd-593/650);
else
    disp('Wrong pfa');
end
end
```

```
function gamma = trhld(ns,M,L,pfa)
A = sqrt(ns)+sqrt(M*L);
B = sqrt(ns)-sqrt(M*L);
C = ns*M*L;
ratio1 = (A^2)/(B^2);
ratio2 = 1 + (power(A,-2/3)/power(C,1/6))*TracyWidom(pfa);
gamma = ratio1*ratio2;
```

```
function v = detect_counter(signal, L, pfa),
v = 0;
ns = length(signal);

%Covariance matrix with Hankel approach
K = ns-L+1;
I = hankel((1:K)',K:ns);    %Hankel Matrix
H=signal(I);
S = svd(H).^2;              %SVD

%threshold
gamma = trhld(K-1,1,L,pfa);
eig_Ratio = max(S)/min(S);  %comparation valoure
if eig_Ratio > gamma
   v = 1;
end
```

References

[1]: FCC Spectrum Policy Task Force, November 2002, *ET Docket No. 02- 135*, Washington DC:

[2]: Tugba Erpek, Karl Steadman, David Jones, 2007, *Dublin IrelandSpectrum Occupancy Measurements Collected On April 16-18, 2007*, Washington DC:Shared Spectrum Company

[3]: FCC, 2003, *FACILITATING OPPORTUNITIES FOR FLEXIBLE, EFFICIENT, AND RELIABLE SPECTRUM USE EMPLOYING COGNITIVE RADIO TECHNOLOGIES (FCC-03-322)*, Washington DC:FCC

[4]: Gerald Youngblood, 2002, *A Software-Defined Radio for the Masses, Part 1*, Austin TX:QEX

[5]: Luca De Vito, Sergio Rapuano, IEEE Member, Maurizio Villanacci, May 12-15, 2008, *An Improved Method for the Automatic Digital Modulation Classification*, Vancouver Island, Canada:IEEE International Instrumentation and Measurement Technology

[6]: Luca De Vito, Sergio Rapuano, IEEE Member, Maurizio Villanacci, May 12-15, 2008, *Real Time Spectrum Analyzer Including an Automatic Digital Modulation Classifier*, Vancouver Island, Canada:IEEE International Instrumentation and Measurement Technology

[7]: M.D.Millhaem, Sept 2006, *SDR advantages for test equipment*, California:IEEE Systems Readiness Technology Conference

[8]: Eric Blossom Jun 01 2004, *GNU Radio: Tools for Exploring the Radio Frequency Spectrum*

[9]: , LESIM web site, http://lesim1.ing.unisannio.it/

[10]: Luca De Vito, Daniele Domenico Napolitano, Sergio Rapuano, Maurizio Villanacci, , May 2010, *Eigenvalue-based Signal Detector for an Automatic Modulator Classifier*, Austin TX:IEEE

[11]: Yonghong Zeng and Ying-Chang Liang, Sep2007, *Maximum-minimum eigenvalue detection for cognitive radio*, Singapore:IEEE 18th Int.Symp.Pers.Indoor Mobile Radio Commun.

[12]: Joseph Mitola and Gerald Q.Maguire, August 1999, *Cognitive Radio: Making software Radios More Personal*, :IEEE Personal communications

[13]: Joseph Mitola III, 8 May 2000, *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*, Sweden:Royal Institute of Technology

[14]: Joseph Mitola III, 2006, *Cognitive Radio Architecture: The Engineering Foundations of Radio XML*, :Wiley

[15]: Anil Shukla, 2007, *Cognitive Radio Technology – A Study forOfcom*, London:QinetiQ

[16]: Simon Haykin and Life Fellow (Feb 2005), *Cognitive Radio: Brain-Empowered Wireless communications*

[17]: Bluetooth SIG, 2004, *Specification of the Bluetooth system, Master Table of*

Contents and Compliance Requirements, :Bluetooth SIG
[18]: Zhi Quan, Shuguang Cui, H.Vincent Poor, Ali H.Sayed 2008, *Collaborative Wideband Sensing for Cognitive Radios*
[19]: Danijela Cabric, Artem Tkachenko, Robert W.Brodersen, Oct 2006, *Spectrum sensing measurements of pilot, energy, and collaborative detection*, Washington DC:IEEE Military Commun. Conference
[20]: IEEE, IEEE SCC41 web site, www.scc41.org
[21]: J. Proakis, , *Digital Communications*, :Mc Graw Hill
[22]: Danijela Cabric, Shridhar Mubaraq Mishra, and Bober W.Brodersen, nov 2004, *Implementation Issues in Spectrum Sensing for Cognitive Radios*, Berkeley :Berkeley Wireless Research Center, University of California
[23]: S.Xu Y.Shang H.Wang, Nov. 2008, *SVD Based Sensing of a Wireless Microphone Signal in Cognitive Radio Networks*, Singapore:IEEE Singapore International Conference
[24]: John G. Proakis, Dimitris G. Manolakis, , *Digital signal processing, principles, algorithms and applications*, :
[25]: Lloyd N. Trefethen, , *Numerical Linear Algebra*, :
[26]: A.M. Tulino and S. Verdu, 2004, *Random Matrix Theory and Wireless Communications*, :
[27]: I.M. Johnston, 2001, *On the Distribution of the largest eigenvalue in principle components analysis*, :The Annals of Statistic
[28]: K. Johansson, 2000, *Shape fluctuations and random matrices*, :Comm. Math Pysh
[29]: Jean-Philippe Lang, official GNU Radio web site, http://gnuradio.org/redmine/wiki/gnuradio
[30]: Ettus, ETTUS web site, http://www.ettus.com/
[31]: Kirarah Amiri, Yang Sun, Patrick Murphy, Chris Hunter, Joseph R.Cavallaro, Ashutosh Sabharwal, May 2007, *WARP, a Modular Testbed for Configurable Wireless Network Research at Rice*, :
[32]: WARP, WARP official web site, http://warp.rice.edu/
[33]: BEE2, BEE2 official web site, http://bee2.eecs.berkeley.edu/
[34]: GNU, GNU official web site, http://www.gnu.org/
[35]: , Linear Technology Corporation, www.linear.com
[36]: , Xilinx web site, www.sikinx.com
[37]: , SWIG web site, http://www.swig.org/
[38]: , Eric Blossom tutorial for making a new GNU Radio processing block, http://www.gnu.org/software/gnuradio/doc/howto-write-a-block.html
[39]: , GNU Scientific Library, http://www.gnu.org/software/gsl/