POLITECNICO DI TORINO

DOCTORAL SCHOOL

PhD program in Electronics and Communications Engineering – XXVIII Cycle

Doctoral dissertation

Spectrum sensing algorithms and software-defined radio implementation for cognitive radio systems

Daniel Gaetano Riviello

Supervisor
Prof. Roberto Garello

Coordinator of the PhD program
Prof. Ivo Montrosset

April 2016

Summary

The scarcity of spectral resources in wireless communications, due to a fixed frequency allocation policy, is a strong limitation to the increasing demand for higher data rates. However, measurements showed that a large part of frequency channels are underutilized or almost unoccupied.

The *cognitive radio* paradigm arises as a tempting solution to the spectral congestion problem. A cognitive radio must be able to identify transmission opportunities in unused channels and to avoid generating harmful interference with the licensed primary users. Its key enabling technology is the *spectrum sensing* unit, whose ultimate goal consists in providing an indication whether a primary transmission is taking place in the observed channel. Such indication is determined as the result of a binary hypothesis testing experiment wherein null hypothesis (alternate hypothesis) corresponds to the absence (presence) of the primary signal.

The first parts of this thesis describes the spectrum sensing problem and presents some of the best performing detection techniques. Energy Detection and multi-antenna Eigenvalue-Based Detection algorithms are considered. Important aspects are taken into account, like the impact of noise estimation or the effect of primary user traffic. The performance of each detector is assessed in terms of false alarm probability and detection probability.

In most experimental research, cognitive radio techniques are deployed in *software-defined radio* systems, radio transceivers that allow operating parameters (like modulation type, bandwidth, output power, etc.) to be set or altered by software.

In the second part of the thesis, we introduce the software-defined radio concept. Then, we focus on the implementation of Energy Detection and Eigenvalue-Based Detection algorithms: first, the used software platform, GNU Radio, is described, secondly, the implementation of a parallel energy detector and a multi-antenna eigenbased detector is illustrated and details on the used methodologies are given. Finally, we present the deployed experimental cognitive testbeds and the used radio peripherals.

The obtained algorithmic results along with the software-defined radio implementation may offer a set of tools able to create a realistic cognitive radio system with real-time spectrum sensing capabilities.

Acknowledgements

This thesis is the result of a 3-year research activity during which I have met and collaborated with amazing people.

I would like to express my deepest gratitude to my Ph.D. advisor, Prof. Roberto Garello, who was my Master's thesis advisor too, and guided and supported me in all possible ways ever since.

I would like to thank Dr. Alberto Perotti and Andrea Molino for hosting me at CSP - ICT Innovation for more than one year. I had the chance to work with great colleagues like Eng. Floriana Crespi and Simone Scarafia. The project described in Chap. 9 was done during that period and I especially want to thank Eng. Ferdinando Ricchiuti for his assistance.

Likewise, I am grateful to Eng. Claudio Pastrone and Dr. Hussein Khaleel for hosting me at the Pervasive Technologies Research Lab of Istituto Superiore Mario Boella (ISMB).

I am also grateful to Prof. Jean-Marie Gorce, who gave me the great opportunity to spend my last year at the Centre of Innovation in Telecommunications and Integration of service (CITI) Laboratory in Lyon. I had the chance to work with advanced radio equipment, as described in Chap. 11. A special thanks goes to Dr. Yasser Fadlallah for his support during my stay at CITI Lab.

Finally, I want to sincerely thank my friend and colleague Pawan Dhakal for our fruitful and lasting collaboration, Dr. Giuseppa Alfano for her help in the reviewing process, and, last but not least, my family and everyone who supported me during these years.

Publications

Some of the contents of this thesis have previously appeared or are going to appear in the following publications:

Book chapters

- Daniel Riviello, Sergio Benco, Floriana Loredana Crespi, Andrea Ghittino, Roberto Garello and Alberto Perotti, "Spectrum Sensing Algorithms for Cognitive TV White-Spaces System", in *Cognitive Communication and Cooperative HetNet Coexistence*, Springer International Publishing, pp. 71-90, 2014. ISBN: 978-3-319-01402-9. DOI: 10.1007/978-3-319-01402-9_4.

- A. F. Cattoni, O. Tonelli, J. L. Buthler, L. da Silva, C.L. Miranda, P. Sutton, F. Crespi, S. Benco, A. Perotti and D. Riviello, "Designing a CR Test Bed: Practical Issues", in *Cognitive Radio and Networking for Heterogeneous Wireless Networks*, Springer International Publishing, pp. 315-360, 2015. ISBN: 978-3-319-01718-1, DOI: 10.1007/978-3-319-01718-1_12.

Journal papers

- Daniel Riviello, Sergio Benco, Floriana Crespi, Alberto Perotti and Roberto Garello, "Spectrum Sensing in the TV White Spaces", *International Journal on Advances in Telecommunications*, vol. 6, n. 3-4, pp. 109-122, Dec. 2013. ISSN: 1942-2601. http://www.thinkmind.org/download.php?articleid=tele_v6_n34_2013_4

Conference papers

- Daniel Riviello, Sergio Benco, Floriana Crespi, Alberto Perotti and Roberto Garello, "A comparison between multi-sensor and CP-based spectrum sensing for TV white spaces", *Proceedings of the 3rd International workshop of COST Action IC0902*, Ohrid (MK), September 12-14, 2012.

- Daniel Riviello, Sergio Benco, Floriana Crespi, Alberto Perotti and Roberto Garello, "Sensing of DVB-T Signals for White Space Cognitive Radio System", *COCORA 2013, The Third International Conference on Advances in Cognitive Radio*, Apr. 2013, pp. 12-17. ISBN: 978-1-61208-267-7. http://www.thinkmind.orgindex.phpview=article&articleid=cocora_2013_1_30_60031

  ⋆ *Best Paper Award*: http://www.iaria.org/conferences2013/awardsCOCORA13/cocora2013_a1.pdf

- Pawan Dhakal, Roberto Garello, Federico Penna and Daniel Riviello, "Impact of noise estimation on eigenvalue based spectrum sensing in cognitive radio systems", *Proceedings of the 4th International workshop of COST Action IC0902*, Rome, October 9-11, 2013. http://newyork.ing.uniroma1.it/IC0902/4th-Workshop/Technical_contributions/October_9/Session_1/06_IC0902_4th_Workshop_contribution_Riviello.pdf

- Daniel Riviello, Pawan Dhakal, Roberto Garello and Federico Penna, "Hybrid Approach Analysis of Energy Detection and Eigenvalue Based Spectrum Sensing Algorithms with Noise Power Estimation", *COCORA 2014, The Fourth International Conference on Advances in Cognitive Radio*, Nice, 23-27 Feb. 2014, pp. 20-25. http://www.thinkmind.org/download.php?articleid=cocora_2014_1_40_80029

- Pawan Dhakal, Roberto Garello, Federico Penna and Daniel Riviello, "Impact of Noise Estimation on Energy Detection and Eigenvalue Based Spectrum Sensing Algorithms", *Communications (ICC), 2014 IEEE International Conference on*, Sydney, 10-14 Jun. 2014, pp. 1367-1372, DOI: 10.1109/ICC.2014.6883512.

- P. Dhakal, D. Riviello, R. Garello, "SNR Wall Analysis of Multi-Sensor Energy Detection with Noise Variance Estimation", *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*, Barcelona, 26-29 Aug. 2014, pp. 680-684. DOI: 10.1109/ISWCS.2014.6933440.

- Daniel Riviello, Pawan Dhakal, Roberto Garello, "On the Use of Eigenvectors in Multi-Antenna Spectrum Sensing with Noise Variance Estimation", *Signal Processing and Integrated Networks (SPIN), 2015 2nd International Conference on*, Noida, 19-20 Feb. 2015, pp. 44-49, DOI: 10.1109/SPIN.2015.7095339.

- Daniel Riviello, Pawan Dhakal, Roberto Garello, "Performance Analysis of Multi-Antenna Hybrid Detectors and Optimization with Noise Variance Estimation", *COCORA 2015, The Fifth International Conference on Advances in Cognitive Radio*, Barcelona, 19-24 Apr. 2015, pp. 14-19, ISBN: 978-1-61208-403-9. http://www.thinkmind.org/download.php?articleid=cocora_2015_1_30_80029

- Pawan Dhakal, Daniel Riviello, "Multi-Antenna Energy Detector Under Unknown Primary User Traffic", *COCORA 2016, The Sixth International Conference on Advances in Cognitive Radio*, Lisbon, 21-25 Feb. 2016, pp. 15-20, ISBN: 978-1-61208-456-5. http://www.thinkmind.org/download.php?articleid=cocora_2016_1_40_80032

  ⋆ *Best Paper Award*: http://www.iaria.org/conferences2016/awardsCOCORA16/cocora2016_a1.pdf

- Pawan Dhakal, Shree K. Sharma, Symeon Chatzinotas, Björn Ottersten, and Daniel Riviello, "Effect of Primary User Traffic on Largest Eigenvalue Based Spectrum Sensing Technique", *Cognitive Radio Oriented Wireless Networks (CROWNCOM), 2016 11th EAI International Conference on*, Grenoble, May 30-Jun. 1, 2016. [To be published].

Contents

List of Tables

List of Figures

List of Acronyms

AC Autocorrelation

A/D Analog-to-Digital

AWGN Additive White Gaussian Noise

BEVE Blind EigenVEctor Test

CDF Cumulative Distribution Function

CP Cyclic Prefix

CR Cognitive Radio

CRN Cognitive Radio Network

D/A Digital-to-Analog

DC Direct Current

DMR Digital Mobile Radio

DSP Digital Signal Processor

DVB-T Digital Video Broadcasting - Terrestrial

EBD Eigenvalue Based Detection

ED Energy Detection

ERD Eigenvalue Ratio Detector

EVE EigenVEctor Test

FFT Fast Fourier Transform

FPGA Field-Programmable Gate Array

GLRT Generalized Likelihood Ratio Test

GRC GNU Radio Companion

GUI Graphical User Interface

IFFT Inverse Fast Fourier Transform

HED see HED1

HED1 Hybrid ED approach 1 with offline noise estimation

HED2 Hybrid ED approach 2 with online noise estimation

HEVE Hybid EigenVEctor Test

HRLRT see HRLRT1

HRLRT1 Hybrid RLRT approach 1 with offline noise estimation

HRLRT2 Hybrid RLRT approach 2 with online noise estimation

IEEE Institute of Electrical and Electronics Engineers

I/Q In-phase and Quadrature

LRT Likelihood Ratio Test

LRT- Noise Independent Likelihood Ratio Test

LTE Long Term Evolution

MAC Medium Access Control

MIMO Multiple-Input Multiple-Output

ML Maximum-Likelihood

NP Neyman-Pearson

OFDM Orthogonal Frequency-Division Multiplexing

pdf Probability Density Function

pmf Probability Mass Function

PSK Phase-Shift Keying

PU Primary User

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase-Shift Keying

RF Radio Frequency

RLRT Roy's Largest Root Test

RMT Random Matrix Theory

ROC Receiver Operating Characteristic

SDR Software-defined Radio

SNR Signal-to-Noise Ratio

SS Steady State

STA Station

SU Secondary User

TCP Transmission Control Protocol

TS Transient State

TVWS TV White Spaces

TW2 Tracy-Widom distribution of order 2

UDP User Datagram Protocol

UMP Uniformly Most Powerful

USRP Universal Software Radio Peripheral

WiMAX Worldwide Interoperability for Microwave Access

WRAN Wireless Regional Area Network

XML Extensible Markup Language

Chapter 1

Introduction

The increasing demand for higher data rates in wired, and most of all, wireless technology is a consequence of the transition from voice-only communications to multimedia type applications. Wireless communication has been the fastest growing segment of the communications industry in the past few decades. Number of high speed data services (e.g., Zigbee, WiMax-Advanced, LTE, Ultra-wide Band Network), various applications (e.g., IP Television, high-speed wireless internet, cellular telephony including multimedia services) and supporting electronic devices (for example mobiles, tablets, computers) are just some of the advances in the field of wireless communications that can be named. Moreover, many new research works on Wireless Sensor Network (WSN), Next Generation Networking (NGN) services, telemedicine, smart home appliances and many more certainly demand for the new band allocations of radio spectrum.

Wireless channels are characterized by a fixed spectrum assignment policy. Electromagnetic spectrum is strictly regulated and licenced by governmental entities, for instance, the Federal Communication Commission (FCC) in United States, the Office of Communications (OFCOM) in UK and Ministero dello Sviluppo Economico - Dipartimento per le Comunicazioni in Italy. The rapid development in the field of wireless communications certainly creates a big challenge for every licensing organization to accommodate all the new applications and services noted above with the limited electromagnetic spectrum. The frequency allocation chart of UK [1] in Fig. 1.1 shows that large portion of the radio spectrum is already assigned to traditional services (Mobile, Maritime Mobile, Fixed Satellite Services, Radio Navigation), and the same applies to Italy [2] and US. Although some unlicensed bands are available, such as the industrial, scientific and medical (ISM) band in 2.4 GHz, which could be the possible solution for accommodating new services, multiple wireless technologies are already deployed in these bands such as 802.11 Wireless Local Area Network (WLAN), cordless phones, Bluetooth, Wireless Personal Area Network (WPAN), etc. Some other examples of unlicenced frequency bands include U-NII Unlicensed

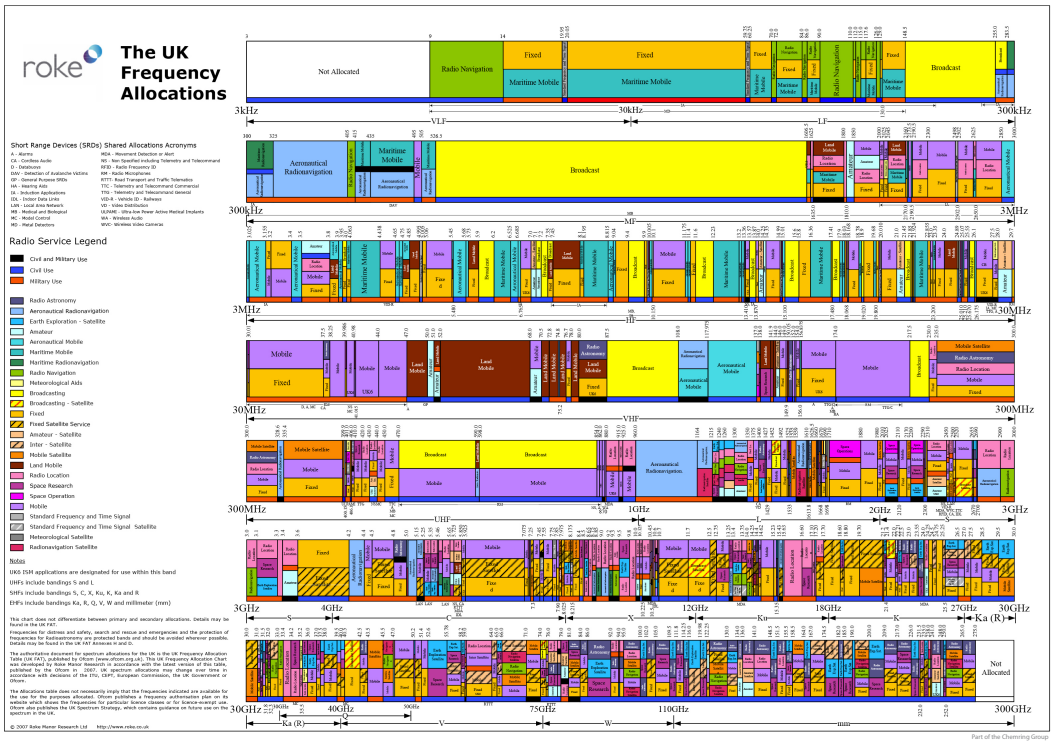


Figure 1.1: OFCOM frequency allocation chart for UK [1].

National Information Infrastructure U-NII-1 and U-NII-2 bands where systems such as IEEE802.11a WLAN and IEEE802.11n WLAN operate.

However, the measurements carried out indicate a distinct difference in the frequency channel usage where a larger part of frequency channels are partially occupied or almost unoccupied [3] (see Fig. 1.2). Some of these unoccupied frequency bands are allocated to broadcasting services and referred to as *white spaces*. In some cases, these hardly used bands are specifically assigned for a purpose, such as guard band to prevent interference while in other cases these white spaces exist naturally between used channels. That is because assigning nearby transmissions to immediate adjacent channels will create destructive interference to both channels. In addition to white space assigned for technical reasons, there is also unused radio spectrum which has either, never been used or is becoming free as a result of technical changes.

For all these reasons, the need for adopting new techniques with capability of exploiting the available spectrum arises. *Cognitive radio* (CR) is standing out to be a tempting solution to the spectral congestion problem since it introduces a proper exploitation in the utilization of the frequency bands that are not largely occupied by licensed users [5]. Below is the formal definition of Cognitive Radio adopted by

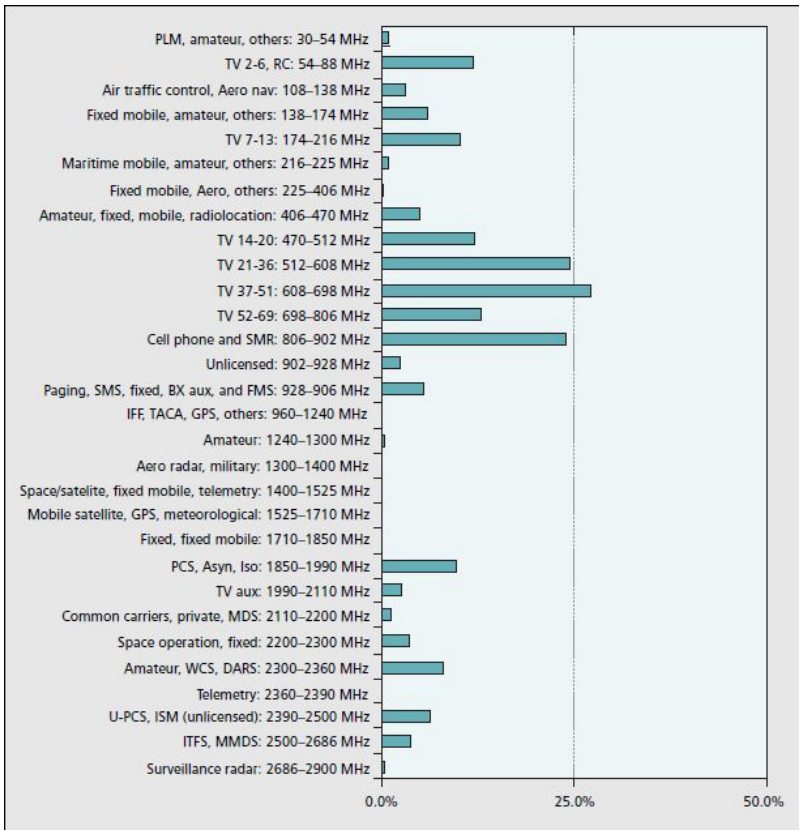


Figure 1.2: Spectrum occupancy chart [4].

the Federal Communication Commission (FCC) [6]:

> *A radio or system that senses its operational electromagnetic enviroment and can dynamically and autonomously adjust its radio operating parameters to modify system operation, such as maximize throughput, mitigate interference, facilitate interoperability, access secondary markets.*

Here's another definition given by IEEE-USA [7]:

> *A radio frequency transmitter/receiver that is designed to intelligently detect whether a particular segment of the radio spectrum is currently in use, and to jump into (and out of, as necessary) the temporarily-unused spectrum very rapidly, without interfering with the transmissions of other authorized users.*

The key feature of a CR is the capability to locally exploit in an autonomous way the unused spectrum providing new ways for spectrum access. *Spectrum sensing*

remains a key enabling technology for cognitive radios. They have to sense, measure, learn, and have a full awareness of the parameters related to the radio channel characteristics in order to identify spectrum opportunities and maybe most importantly, prevent interference with the licensed primary users (PUs).

The term *primary users* refers to the users with higher priority or legacy rights on the use of a specific part of the spectrum and *secondary users* refers to the users with lower priority. The secondary ones exploit the spectrum in a way that prevents them from causing interference to the primary users. This poses the need for secondary users to have cognitive radio capabilities that include sensing the spectrum reliably. By sensing reliably, it simply means that the secondary users should be in a position to check whether the spectrum is being used by the primary user. Additionally, cognitive radios should be able to change the parameters of the radio transceiver in order to exploit the unused part of it [8].

The concept of spectrum sensing is similar in some ways to the multiple access method used in IEEE 802.11, *Carrier sense multiple access with collision avoidance* (CSMA/CA), in which a station (STA) senses the channel in order to gain access and to send data in a basic service set (BSS). The biggest difference is that a station in IEEE 802.11 senses only a well-defined channel, 20 MHz in the ISM radio bands around 2.4 GHz, while a cognitive radio system must be able to sense and scan the whole spectrum or at least a larger portion of it.

The crucial task in spectrum sensing lies in the decision making as to whether a primary signal is present or not. Detection theory is a field of statistical signal processing where optimal or highly reliable decision making procedures are developed. The data sets obtained from the observations are assumed to be samples of continuous-time waveforms or a sequences of data points. This decision making process is formulated as a hypothesis testing problem. In common cases, the null hypothesis offers a description of the scenario in which only noise is present and when the PU is not active. The alternate hypothesis describes the scenario where primary transmission is present in the noisy observations [9].

During decision making, errors may occur and there are two types of error likely to take place; the first one is known as false alarm and the second one is misdetection. Looking at the first one, the error occurs when one decides that the primary is active based on the information (data) that comes from distribution corresponding to null hypothesis. This false alarm leads us to an unnecessary reduction of the secondary use of spectrum. Therefore, it is highly crucial to control the false alarm rate. The latter occurs when there is a failure of detecting PU activity leading us to a conclusion of availability of spectrum for secondary use and this type of error creates interference with PU signals, consequently retransmissions and reduced rate for both primary and secondary systems. Moreover, harmful interference is caused to licensed systems that have paid for the spectrum [9].

1.1 Spectrum sensing algorithms

Several spectrum sensing methods have been proposed for Cognitive Radio applications including Energy Detection (ED) [10], Matched Filtering [11], Feature Detection Algorithms [12]. Energy Detection does not make any assumption on the PU signal statistics while the matched filter detection algorithms assume the complete knowledge on the pilot waveform or the preamble to design the detectors. Feature detection lies in the middle of these two extremes and only makes certain assumptions on the statistical properties of the PU signal. Even though Matched Filtering and feature detection algorithms are known to outperform Energy Detection, the requirement of the knowledge of the PU signal characteristics and the long sensing period makes them less suitable for spectrum sensing in Cognitive Radio Networks (CRN), since the knowledge of the PU signal is usually unavailable. Thus, for Cognitive Radio applications, ED is considered to be the simplest and most popular sensing algorithm which compares the energy of the received signal to the noise variance. In recent years, sensing techniques based on the eigenvalues of the received covariance matrix evolved as a promising solution for spectrum sensing. Eigenvalue-Based detection (EBD) schemes are further categorized as "semi-blind EBD", in which the noise level is assumed to be known, and "blind EBD", in which the noise level is not known. Methods belonging to the first class provide better performance especially when the noise variance is exactly known, whereas blind methods are more robust to uncertain or varying noise level.

1.2 Software defined-radio implementation

The *software-defined radio* (SDR) concept has been introduced almost twenty years ago in [13] and it is innovative even nowadays. The main scope of SDR is to improve the flexibility of radio communication devices by implementing the digital sections entirely in software using dedicated or general-purpose processors. A software-defined radio system is a radio communication system where components that have been typically implemented in hardware (e.g., mixers, filters, amplifiers, modulators/demodulators, etc.) are instead implemented by means of software on a personal computer or embedded computing devices.

Different SDR platforms have been developed in recent and past years. Among these, an open platform called GNU Radio [14] has prevailed firstly in the academic community and it is becoming frequently adopted even in industrial projects. GNU Radio is a free and open-source software development toolkit that provides signal processing blocks to implement software-defined radios and signal processing systems. GNU Radio is not primarily a simulation tool, although it can be used for this purpose. The GNU Radio infrastructure is written entirely in C++, it is composed

of a wide collection of signal processing blocks. A set of interconnected blocks forms a flowgraph, which can be written both in C++ or in Python. GNU Radio provides also a Graphical User Interface (GUI), called GNU Radio Companion (GRC). It is possible to create new blocks in addition to the existing ones.

When paired with suitable RF front-ends, a real radio communication system can be carried out. The combination of a computer with GNU Radio installed and an RF front-end allows the creation of affordable SDR testbeds. A lot of hardware vendors provide GNU Radio support for their products, ranging from very expensive measurement-quality systems, to very cheap receiver hardware, but the most commonly used with GNU Radio are the Universal Software Radio Peripheral (USRP) devices by Ettus Research and National Instruments.

The implementation of spectrum sensing algorithms in a SDR platform is a crucial task for the construction of a cognitive system prototype. Sensing techniques have been implemented in GNU Radio for instance in [15, 16]. In this work, Energy Detection and Eigenvalue-Based Detection algorithms have been implemented in GNU Radio, more in detail a parallel Energy Detector designed for *Digital Mobile Radio* (DMR) signals and among the EBD algorithms the Roy's Largest Root Test (RLRT) and the Generalized Likelihood Ratio Test (GLRT).

1.3 Thesis organization

This thesis focuses on several aspects of spectrums sensing algorithms. Part I illustrates and assesses the performance of Energy Detection (ED) and Eigenvalue-Based Detection (EBD) test statistics, while Part II focuses on the software-defined radio (SDR) implementation in GNU Radio of the sensing algorithms. More in detail:

- Chap. 2 introduces the sensing problem and the mathematical system model that will be used throughout the thesis. Moreover, analytical results are provided for Energy Detection (ED) and two of the main EBD algorithms: Roy's Largest Root Test (RLRT) and the Generalized Likelihood Ratio Test (GLRT).

- Chap. 3 focuses on the sensing of DVB-T signals. After a description of the PU signal, a comparison between single-antenna parametric (feature-based) detectors and multi-antenna non parametric detectors is carried out. Simulations have been performed with different channel models.

- Chap. 4 introduces the idea of auxiliary noise variance estimation and presents hybrid approaches for ED and RLRT. Analytical formulations for the new tests are provided and performance analysis for both approaches is carried out in order to asses the impact of the noise power estimation.

- Chap. 5 presents the EigenVEctor Test (EVE), which exploits the eigenvector associated to the largest eigenvalue of the sample covariance matrix to estimate the channel. Results show EVE is able to outperform ED and EBD algorithms.

- Chap. 6 presents the concept of SNR Wall phenomenon for ED and extends it to the multi-antenna ED case. The analytical expression of the uncertainty bound for multi-antenna ED is derived and proved to be independent of the number of antennas.

- Chap. 7 studies the effect of primary user (PU) traffic on the performance of RLRT. A realistic and simple PU traffic model is considered, which is based only on the discrete time distribution of PU free and busy periods. Analytical expressions for the probability density functions of the decision statistic are derived and validated by simulations.

- Chap. 8 introduces the SDR and SDR-based testbed concepts, and presents the GNU Radio software platform. Since the implementation of ED and EBD algorithms required the creation of custom application in addition to the existing collection, the main focus of the chapter is on how to write a custom signal processing block in GNU Radio.

- Chap. 9 presents the first SDR implementation, a parallel Energy Detector designed to sense and collect occupancy statistics on Digital Mobile Radio (DMR) channels. The project was carried out in collaboration with CSP - ICT Innovation.

- Chap. 10 presents the GNU Radio implementation of a multi-antenna eigenvalue-based detector, whose test statistic can be selected between RLRT and GLRT. The main focus of the chapter is the description of the eigenvalue algorithm (Lanczos method plus bisection).

- Finally, Chap. 11 presents the SDR peripherals, namely the National Instruments NI USRP 2920 and the Nutaq PicoSDR 4x4, and the multi-antenna cognitive testbeds deployed in the CITI Laboratory of INSA Lyon.

Part I

Spectrum sensing theory

Chapter 2

System model and test statistics

Among the functionalities provided by Cognitive Radio, Opportunistic Spectrum Access (OSA) is devised as a dynamic method to increase the overall spectrum efficiency by allowing Secondary Users (SUs) to utilize unused licensed spectrum. For this purpose, a correct identification of available spectral resources by means of spectrum awareness techniques becomes fundamental. Spectrum sensing is defined as

> *The task of finding spectrum holes by sensing the radio spectrum in the local neighborhood of the Cognitive Radio receiver in an unsupervised manner* [17].

Specifically, the task of spectrum sensing involves the following sub-tasks [17]:

- detection of spectrum holes;
- spectral resolution of each spectrum hole;
- estimation of spatial directions of incoming interferes;
- signal classification.

To be specific, this thesis focuses on the detection techniques of spectrum holes. Spectrum hole detection is a very critical component of the Cognitive Radio concept. Tab. 2.1 shows the currently understood requirements about the SU devices sensitivity for three signal types. According to the 802.22 Working Group [18], for a receiver noise figure of 11 dB, the resulting required SNR for the secondary receiver is listed, where noise power is calculated over a bandwidth of 6 MHz for a TV signals and over a bandwidth of 200 KHz for wireless microphones. It is also evident from Tab. 2.1 that each SU is required to operate under very low SNR values. In general, such low SNR values must be expected in all deployment scenarios of CR to protect the primary users from undue interference. Thus, the goal is to design detection algorithm that meet the given constraints at very low SNR.

Parameter	Analog TV	Digital TV	Wireless microphones
Probability of detection	0.9	0.9	0.9
Probability of false alarm	0.1	0.1	0.1
Channel detection Time	$\leq$ 2s	$\leq$ 2s	$\leq$ 2s
Incumbent detection threshold	-94dBm	-116dBm	-107dBm
SNR	1dB	-21dB	-12dB

Table 2.1: Receiver Parameter for 802.22 WRAN [18].

Many spectrum hole detection algorithms have been devised with their own pros and cons. Several spectrum sensing methods have been proposed in context to Cognitive Radio applications including ED [10], Matched Filtering [11], Feature Detection Algorithms [12] and Eigenvalue-Based Detection [19] proposed using individual SU and their cooperative counterpart using multiple SUs. A survey of different spectrum sensing methodologies for Cognitive Radio [20] shows that a remarkable spectrum sensing performance can be attained with feature detection algorithms (e.g., Cyclic Prefix based and pilot based Detector), which exploit some known characteristic of the PU signal but require long sensing periods. Even more, Matched Filtering is assumed to perform best with high processing gain at the constraint of knowing the PU signal properties [21].

2.1 State of the art

In a real scenario, the information about the PU signal is generally not available and even if available, Matched Filtering and Feature Detection algorithms would require a specific implementation of the spectrum sensing unit for each PU signal to be detected. Thus, for CR application the most popular sensing algorithm is the simple Energy Detection (ED), which compares the energy of the received signal to the noise variance σ_v^2. ED requires the knowledge of σ_v^2 value. Performance of ED in AWGN and different fading channels have been studied in many works including [10, 22, 23, 24]. These works assumed a perfect knowledge of the noise power at the receiver, which allows for the perfect threshold design. In that case, ED can work with arbitrarily small values of false alarm probability P_{fa} and misdetection probability P_{md}, by using sufficiently large observation time, even in low SNR environment [25]. However, in real systems the detector does not have a prior knowledge of the noise level. In

recent years, variation and unpredictability of the precise noise level at the sensing device came as a critical issue, which is also known as *noise uncertainty*.

With the goal of reducing the impact of noise uncertainty on the signal detection performance of ED, a large amount of research has been proposed including [26, 25, 27, 28]. Hybrid Spectrum Sensing algorithms based on the combination of ED and Feature Detection techniques have been proposed for the reduction of the effect of noise variance uncertainty [29, 30]. A similar hybrid approach was discussed in [31] utilizing the positive points of ED and Covariance Absolute Value detection methods while [32] used Akaike Information Criterion (AIC), Minimum Description Length (MDL) and Rank Order Filtering (ROF) methods for the estimation of the noise power for energy based sensing. In [25] the fundamental bounds of signal detection in presence of noise uncertainty have been analyzed. This study showed that there is a threshold for the SNR in case of noise uncertainty known as *SNR Wall*, which prevents achieving the desired performance even if the detection interval is made infinitely large. It concluded that the robustness of any detector can be quantified in terms of the *SNR Wall* giving the threshold below which weak signals cannot be reliably detected no matter how many samples are taken. In [28] the asymptotic analysis of the Estimated Noise Power (ENP) of ED was performed to derive the condition of the *SNR Wall* phenomenon. It suggested that the *SNR Wall* can be avoided if the variance of the noise power estimator can be reduced while the observation time increases. [33] proposed an uniform noise power distribution model for the noise uncertainty study of ED in low SNR regime. Similarly, [34] proposed a discrete-continuous model of the noise power uncertainty for the performance analysis of the ED in presence of noise uncertainty. Performance of ED using Bartlett's estimate is being studied in [35].

In recent years various new algorithms able to outperform ED have been applied to CR, mostly based on Random Matrix Theory (RMT) and information theoretic criteria. Two thorough reviews have been presented in [36] and [19]. Different diversity enhancing techniques such as multiple antenna, cooperative and oversampling techniques have been introduced in the literature to enhance the spectrum sensing efficiency in the wireless fading channels [37, 38, 39, 40, 41]. Most of these methods use the properties of the eigenvalues of covariance matrix of the received signal and use the results from advances in RMT. In particular, sensing techniques based on the eigenvalues of the covariance matrix have recently emerged as a promising solution, as they also do not require any prior assumption on the signal to be detected, and typically outperform the popular ED method when multiple sensors are available.

The various EBD algorithms can be divided in two classes:

A. Those that require a prior knowledge of the noise variance σ_v^2 (called "semi-blind" in [19]). This class includes the classical ED, channel independent tests [36], and RLRT [42], which shows the best performance in this class.

B. Those that do not require knowledge of σ_v^2 (referred to as "blind" in [19]). This class includes the Eigenvalue Ratio Detector (ERD) [43, 44], channel and noise-independent tests [36], information theoretic criteria detectors like the Akaike Information Criterion (AIC) and the Minimum Description Length (MDL) [41], and the Generalized Likelihood Ratio Test (GLRT) [45], which shows the best performance in this class.

Methods belonging to the first class provide better performance when the noise variance is exactly known, whereas blind methods are more robust to uncertain or varying noise level. Recently, some research works have focused on the noise variance uncertainty and their effect in semi-blind EBD including [46, 47]. In [46] author showed the importance of accurate noise estimation for better performance of the EBD algorithms.

2.2 Problem formulation

In this thesis the following scenario is usually considered:

- Single, unknown, primary signal. Its samples are modelled as Gaussian and independent.
- Flat-fading channel, constant over all the sample window.
- Additive Gaussian white noise.

The detection problem is formulated as a simple binary test between the mutually exclusive hypotheses:

$\mathcal{H}_0$ (single primary signal absent) and $\mathcal{H}_1$ (single primary signal present).

This model is mostly used in detection theory because it allows a clear analytical approach and represents a benchmark for the case of non-parametric analysis of a single unknown primary signal. In particular relating to CR applications, where it is very popular and it was adopted by many relevant papers (including [45], and many others). The reason is that, despite of its simplicity, it is well matched to many practical situations:

1. *Single primary user.* In many CR scenarios, the primary signal of interest for a secondary opportunistic CR network is unique. This is the case, for example, in reuse of TV signal bands [18] or the coexistence between a Wi-Fi access point and sensor networks or Bluetooth in the 2.4 GHz band, two typical CR applications that have already been put into practice [48, 49]. Scenarios with

multiple signals at the same time have been analyzed [50], however, the single-signal case is the most important because, even if more signals are present, it turns out that the detection performance is determined essentially by the one with highest received power.

2. *Gaussian primary signal.* Most CR sensing algorithms working on the time-axis use non-parametric detection, that does not exploit the (complete or limited) knowledge of the signal shape. This is a realistic assumption for several CR wireless applications: even if we know that the primary signal has a PSK/QAM/OFDM format, the secondary network is not synchronized, neither in carrier or in time (this would require a great amount in complexity, not available for most of current applications). Then, the I/Q samples do not correspond to the constellation signals and do not possess special properties (e.g., constant envelope for PSK signals). Under these conditions, the Gaussian approximation for the signal amplitude turns out to be appropriate for many practical situations.

3. *Uncorrelated signal samples.* In practical CR sensing, some correlation between adjacent samples may be present, but (a) it strongly depends on the shape of the transmitting and the receiver filters and (b) the sampling frequency is completely asynchronous with respect to the received signal. For this reason, it is difficult to be modelled. Furthermore, including it into the framework is expected to have a negligible impact on the detector performance.

4. *Channel and noise.* The flat fading channel assumption is rather realistic when the sampling window time is relatively short and the system mobility is limited, which is the typical scenario for current CR applications. Finally, the Gaussian model for the noise is appropriate in general. (Impulsive noise is usually of secondary importance for CR wireless applications.)

2.2.1 Mathematical framework

We consider a cooperative detection framework in which K receivers or antennas collaborate to identify the presence of a signal.

Let us denote with y_k the discrete baseband complex (I/Q) sample at receiver k and let us define

$$\boldsymbol{y} = [y_1, \ldots, y_k]^T \tag{2.1}$$

the received $K \times 1$ vector containing the K received signal samples.

Under $\mathcal{H}_0$, the received vector contains only noise and consists of K independent complex Gaussian noise samples with zero mean and variance σ_v^2:

$$\boldsymbol{y}|_{\mathcal{H}_0} = \boldsymbol{v} \tag{2.2}$$

where $\boldsymbol{v} \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}_{K\times 1}, \sigma_v^2 \boldsymbol{I}_{K\times K})$ is a vector of circularly symmetric complex Gaussian (CSCG) noise samples.

Under $\mathcal{H}_1$, the received vector contains signal plus noise:

$$\begin{aligned} \boldsymbol{y}|_{\mathcal{H}_1} &= \boldsymbol{x} + \boldsymbol{v} \\ &= \boldsymbol{h}s + \boldsymbol{v} \end{aligned} \tag{2.3}$$

where,

- s is the transmitted signal sample, modelled as Gaussian with zero mean and variance $\sigma_s^2 : s \sim \mathcal{N}_{\mathbb{C}}(0, \sigma_s^2)$
- $\boldsymbol{h}$ is the channel complex vector $\boldsymbol{h} = [h_1, \ldots, h_K]^T$; assumed to be constant and memoryless during the sampling window.

Under $\mathcal{H}_1$, the **average SNR** at the receiver is defined as

$$\rho \triangleq \frac{\mathrm{E}\|\boldsymbol{x}\|^2}{\mathrm{E}\|\boldsymbol{v}\|^2} = \frac{\sigma_s^2 \|\boldsymbol{h}\|^2}{K\sigma_v^2} \tag{2.4}$$

where $\|.\|$ denotes the Euclidean norm and $\mathrm{E}[\cdot]$ is the mean operator.

The statistical covariance matrix of the received signal is defined as:

$$\boldsymbol{\Sigma} \triangleq \mathrm{E}[\boldsymbol{y}\boldsymbol{y}^H]. \tag{2.5}$$

In practice, the receiver constructs a sample covariance matrix to approximate $\boldsymbol{\Sigma}$. Let us now define with N the number of samples collected by each receiver or antenna during the sensing period. It is assumed that consecutive samples are uncorrelated and that all the random processes involved (signals and noise) remain stationary for the sensing duration. Hence, we define $s(n)$, $\boldsymbol{v}(n)$ and $\boldsymbol{y}(n)$, respectively, as the transmitted signal sample, the noise vector and the received signal vector at time n. We define the $1 \times N$ signal vector

$$\boldsymbol{s} \triangleq [s(1), \ldots, s(N)], \tag{2.6}$$

the $K \times N$ noise matrix

$$\boldsymbol{V} \triangleq [\boldsymbol{v}(1), \ldots, \boldsymbol{v}(n), \ldots, \boldsymbol{v}(N)], \tag{2.7}$$

and the $K \times N$ **received matrix**:

$$\boldsymbol{Y} \triangleq [\boldsymbol{y}(1), \ldots, \boldsymbol{y}(n), \ldots, \boldsymbol{y}(N)] \tag{2.8}$$

The hypothesis testing experiment becomes:

$$\boldsymbol{Y}|_{\mathcal{H}_0} = \boldsymbol{V} \tag{2.9}$$

$$\boldsymbol{Y}|_{\mathcal{H}_1} = \boldsymbol{h}\boldsymbol{s} + \boldsymbol{V} \tag{2.10}$$

Then, the **sample covariance matrix** is defined as:

$$\boldsymbol{R} \triangleq \frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^H \tag{2.11}$$

and we will denote by $\lambda_1 \geq \ldots \geq \lambda_K$ the **eigenvalues** of $\boldsymbol{R}$ sorted in decreasing order.

False alarm and detection probabilities

Let T be the **test statistic** employed by a detector to distinguish between $\mathcal{H}_0$ and $\mathcal{H}_1$.The detector computes the test statistic T and compares it against a pre-defined threshold t, if $T > t$ it decides for $\mathcal{H}_1$, otherwise $\mathcal{H}_0$. Usually, the decision threshold t is determined as a function of the target false alarm probability. False alarm and detection probability are defined as follows:

$$\begin{aligned} P_d &= \mathcal{P}(T > t \mid \mathcal{H}_1) \\ P_{fa} &= \mathcal{P}(T > t \mid \mathcal{H}_0). \end{aligned} \tag{2.12}$$

Both P_{fa} and P_d are key quantities for practical CRN: P_{fa} must be low to maximize the spectrum exploitation by the secondary user and P_d must be high to minimize the interference caused by the opportunistic user to the primary one. As an example, the WRAN standard [18] imposes stringent requirements on both of them: $P_{fa} < 0.1$ and $P_d > 0.9$. In practical applications, the decision threshold t is typically computed as a function of the target P_{fa}: this ensures the so-called Constant False Alarm Rate (CFAR) detection. The Receiving Operating Characteristic (ROC) curve is obtained by plotting the probability of correct detection versus the probability of false alarm. In order to compare the performance for different threshold values, ROC curves can be used. ROC curves allow us to explore the relationship between the sensitivity (probability of detection) and specificity (probability of false alram) of a sensing method for a variety of different threshold, thus allowing the determination of an optimal threshold.

2.2.2 The Neyman-Pearson test

The usual criterion for comparing two tests is to fix the false alarm rate P_{fa} and look for the test achieving the higher P_d. The NP lemma [51] is known to provide the Uniformly Most Powerful (UMP) test, achieving the maximum possible P_d for any given value of P_{fa}. The NP criterion is applicable only when both both $\mathcal{H}_0$ and $\mathcal{H}_1$ are simple hypotheses. In our setting this is the case when both the noise level σ_v^2 and the channel vector $\boldsymbol{h}$ are a priori known. The NP test is given by the following likelihood ratio:

$$T_{NP} = \frac{p_1(\boldsymbol{Y}; \boldsymbol{h}, \sigma_s^2, \sigma_v^2)}{p_0(\boldsymbol{Y}; \sigma_v^2)} \tag{2.13}$$

The NP test provides the best possible performance, but requires exact knowledge of both $\boldsymbol{h}$ and σ_v^2. For most practical applications, the knowledge of $\boldsymbol{h}$ is questionable. The noise variance is somewhat easier to know: since we only consider thermal noise, if the temperature is constant some applications may possess an accurate estimation of it.

2.3 Test statistics

In this section, we describe in detail Energy Detection (ED) and Eigenvalue-Based Detection (EBD) sensing algorithms. ED is the simplest and least computationally complex algorithm, hence very easy to be implemented. As far as EBD algorithms are concerned, we will focus on two particular test statistics:

- Roy's Largest Root Test (RLRT)
- Generalized Likelihood Ratio Test (GLRT)

RLRT is the algorithm that shows the best performance in the class of semi-blind algorithms, while GLRT is the best performing algorithm in the blind class, which does not require the knowledge of the noise variance (see Sec. 3.5). For these reasons, these 3 detection algorithms have been implemented in SDR in Chap. 9 and 10.

For each test, theoretical performance results are provided in terms of distribution probabilities for P_{fa} and P_d.

2.3.1 Energy Detection

Energy detection is the spectrum sensing technique that evaluates the signal energy over a certain time interval and compares it against a threshold to decide whether the spectrum is in use or not. The presence of noise in the signal may affect the decision of energy detector thus causing false alarm or even misdetection.

Formulation of the Decision Statistic

Using the information of the received signal matrix $\boldsymbol{Y}$, the test statistic T_{ED} computes the average energy of the received signal over a sensing interval N. The detector compares T_{ED} against a predefined threshold t; if $T_{ED} < t$ then it decides in favor of null hypothesis $\mathcal{H}_0$ otherwise in favor of alternate hypothesis $\mathcal{H}_1$. The average energy of the received signal normalized by the noise variance σ_v^2 can be represented as,

$$T_{ED} = \frac{1}{KN\sigma_v^2} \sum_{k=1}^{K} \sum_{n=1}^{N} |y_k(n)|^2 \tag{2.14}$$

or

$$T_{ED} = \frac{\|\boldsymbol{Y}\|_F^2}{KN\sigma_v^2} \tag{2.15}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Note that it is possible to express T_{ED} in terms of the eigenvalues λ_i of $\boldsymbol{R}$ by exploiting the equivalence $\|\boldsymbol{Y}\|_F^2 = \mathrm{tr}(\boldsymbol{Y}\boldsymbol{Y}^H)$, thus obtaining

$$T_{ED} = \frac{1}{K\sigma_v^2}\mathrm{tr}(\boldsymbol{R}) = \frac{1}{KN\sigma_v^2}\sum_{i=1}^{K}\lambda_i. \tag{2.16}$$

Case 1: Null hypothesis

For null hypothesis, rearranging (2.14) using $y_k(n) = v_k(n)$,

$$T_{ED}|_{\mathcal{H}_0} = \frac{1}{KN\sigma_v^2}\sum_{k=1}^{K}\sum_{n=1}^{N}|v_k(n)|^2 \tag{2.17}$$

$$= \frac{1}{2KN}\sum_{k=1}^{K}\sum_{n=1}^{N}\left|\frac{v_k^R(n)}{\frac{\sigma_v}{\sqrt{2}}} + j\frac{v_k^C(n)}{\frac{\sigma_v}{\sqrt{2}}}\right|^2 \tag{2.18}$$

$$= \frac{1}{2KN}\sum_{k=1}^{K}\sum_{n=1}^{N}\left|\beta^R + j\beta^C\right|^2 \tag{2.19}$$

$$= \frac{1}{2KN}\sum_{k=1}^{K}\sum_{n=1}^{N}\beta_R^2 + \beta_C^2 \tag{2.20}$$

where $v_k^R(n)$ and $v_k^C(n)$ are real and imaginary part of the noise signal $v_k(n)$ respectively. $\beta_R = \sqrt{2}v_k^R(n)/\sigma_v$ and $\beta_C = \sqrt{2}v_k^C(n)/\sigma_v$. As $v_k(n)$ is a zero mean and σ_v^2 variance complex valued Gaussian random variable, β_R and β_C are standard normal random variables with mean zero and unity variance. The numerator of T_{ED} in (2.20) is sum of square of $2KN$ standard normal random variable with mean zero and variance 1, thus, the decision statistic $T_{ED}|_{\mathcal{H}_0}$ follows the chi-squared distribution (also χ^2-distribution) with $2KN$ degrees of freedom scaled by the factor $(1/2KN)$.

Thus, (2.20) can be written as,

$$T_{ED}|_{\mathcal{H}_0} = \frac{1}{2KN}\chi_{2KN}^2 \tag{2.21}$$

Case 2: Alternate hypothesis

For alternate hypothesis, rearranging (2.14) using $y_k(n) = h_k s_k(n) + v_k(n)$,

$$T_{ED}|_{\mathcal{H}_1} = \frac{1}{KN\sigma_v^2} \sum_{k=1}^{K} \sum_{n=1}^{N} |h_k s_k(n) + v_k(n)|^2 \tag{2.22}$$

$$= \sum_{k=1}^{K} \frac{\sigma_{t_k}^2}{KN\sigma_v^2} \sum_{n=1}^{N} \left| \frac{h_k s_k(n) + v_k(n)}{\frac{\sigma_{t_k}}{\sqrt{2}}} \right|^2 \tag{2.23}$$

$$= \sum_{k=1}^{K} \frac{\sigma_{t_k}^2}{2KN\sigma_v^2} \sum_{n=1}^{N} |\alpha|^2 \tag{2.24}$$

where $\alpha = \frac{h_k s_k(n) + v_k(n)}{\sigma_{t_k}/\sqrt{2}}$. As h_k is assumed to be constant for the sensing interval and both the signal and noise are complex valued Gaussian signals with variances σ_v^2 and σ_s^2 respectively and both are independent signals, $h_k s_k(n) + v_k(n)$ is also complex valued Gaussian signal with mean zero and variance $\sigma_{t_k}^2$. It is clear that α is also a complex valued standard normal random variable with mean zero and unity variance. So the sum $\sum_{n=1}^{N} |\alpha|^2$ in an expression of (2.24) follows the chi-squared distribution with $2N$ degrees of freedom. (2.24) can be re-written as,

$$T_{ED}|_{\mathcal{H}_1} = \sum_{k=1}^{K} \left(\frac{|h_k|^2 \sigma_s^2 + \sigma_v^2}{2KN\sigma_v^2} \right) \chi_{2N}^2 \tag{2.25}$$

$$= \sum_{k=1}^{K} \left(\frac{|h_{t_k}|^2 \sigma_s^2}{2KN\sigma_v^2} \right) \chi_{2N}^2 + \sum_{k=1}^{2} \frac{1}{2KN} \chi_{2N}^2 \tag{2.26}$$

$$T_{ED}|_{\mathcal{H}_1} = \frac{K\rho\chi_{2N}^2}{2KN} + \frac{\chi_{2KN}^2}{2KN} \tag{2.27}$$

Normal Approximation of ED Decision Statistic

According to the Central Limit Theorem, when N is made sufficiently large, the chi-squared distributed random variable in (2.27) converges to a Gaussian distribution.

For good approximation, different models have been developed such as Edell's model [52], Torrieri's model [53] and Berkeley model [54, 55] which analyzed the accuracy of different Energy Detection models in approximating the exact solution of the T_{ED} and concluded that these models almost have the same performance for such scenario. Thus, for the result in (2.27) and (2.21), using Berkeley model [54], the chi-squared distribution function can be approximated to a normal distribution function as,

$$T_{ED} = \begin{cases} \mathcal{N}\left(1, \frac{1}{KN}\right) & \mathcal{H}_0, \\ \mathcal{N}\left(\rho + 1, \frac{K\rho^2 + 2\rho + 1}{KN}\right) & \mathcal{H}_1 \end{cases} \tag{2.28}$$

Formulation of Detection and False Alarm Probabilities

Hypothesis test is a procedure which divides the space of observations into 2 regions, Rejection Region (R) and Acceptance Region (A). The two important characteristics of a test are called significance and power referring to errors of type I and II in hypothesis testing which relates to probability of false alarm and probability of detection respectively. The probabilities of false alarm P_{fa} and probability of detection P_d for a given decision statistic referring to ED test is given by,

$$P_{fa} = \text{Prob}\{T_{ED} > t|_{\mathcal{H}_0}\} \tag{2.29}$$
$$P_d = \text{Prob}\{T_{ED} > t|_{\mathcal{H}_1}\} \tag{2.30}$$

Based on the statistics of T_{ED} shown in (2.28), P_{fa} can be evaluated as,

$$P_{fa} = \int_t^\infty T_{ED}|_{\mathcal{H}_0} dt \tag{2.31}$$
$$= 1 - \phi(t) \equiv 1 - \frac{1}{2}\left[1 + erf\left[\frac{t-\mu}{\sqrt{2\sigma^2}}\right]\right] \tag{2.32}$$
$$= \frac{1}{2}\left[1 - erf\left[\frac{t-\mu}{\sqrt{2\sigma^2}}\right]\right] \tag{2.33}$$
$$= \frac{1}{2} erfc\left[\frac{t-\mu}{\sqrt{2\sigma^2}}\right] \tag{2.34}$$
$$P_{fa} = Q\left(\frac{t-\mu}{\sqrt{2\sigma^2}}\right) \tag{2.35}$$

where $\phi(t)$ is the Cumulative Distribution Function (CDF) of the normal distribution, $erf()$ is the error function and $erfc()$ is the complementary error function and $Q()$ is the complementary CDF of a normal random variable. Now putting the value of mean and variance for $\mathcal{H}_0$ from (2.28),

$$P_{fa} = Q\left[(t-1)\sqrt{KN}\right] \tag{2.36}$$

Similarly, for the same threshold level, the expression of the probability of detection is given by,

$$P_d = Q\left[\frac{(t-1-\rho)\sqrt{KN}}{\sqrt{K\rho^2 + 2\rho + 1}}\right] \tag{2.37}$$

2.3.2 Eigevalue-Based Detection algorithms

The spectral properties of the covariance matrix are the main reason why the largest eigenvalue of this matrix plays a crucial role in both RLRT and GLRT.

Under $\mathcal{H}_0$ and $\mathcal{H}_1$, respectively, the statistical covariance matrix $\boldsymbol{\Sigma}$, defined in Sec. 2.2.1, can be written as:

$$\boldsymbol{\Sigma} = \begin{cases} \sigma_v^2 \boldsymbol{I}_K & \mathcal{H}_0 \\ \boldsymbol{\Sigma}_{\boldsymbol{x}} + \sigma_v^2 \boldsymbol{I}_K & \mathcal{H}_1 \end{cases} \tag{2.38}$$

with $\boldsymbol{\Sigma}_{\boldsymbol{x}} = \sigma_s^2 \boldsymbol{h}\boldsymbol{h}^2$. Let $\zeta_1 \geq \ldots \geq \zeta_K$ be the eigenvalues of $\boldsymbol{\Sigma}$ sorted in decreasing order and let $\boldsymbol{\zeta} \triangleq [\zeta_1, \ldots, \zeta_K]$.

Under $\mathcal{H}_0$, it is straightforward to verify

$$\boldsymbol{\zeta}|_{\mathcal{H}_0} = \sigma_v^2 \mathbf{1}_{1,K} \tag{2.39}$$

where $\mathbf{1}_{1,K}$ denotes a $1 \times K$ vector in each every element is equal to 1.

Under $\mathcal{H}_1$, since the rank of $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ is 1, all but one eigenvalue of $\boldsymbol{\Sigma}$ are still equal to σ_v^2. The signal eigenvalue can be written as written

$$\zeta_1 = \zeta_x + \sigma_v^2 \tag{2.40}$$

where ζ_x, the only non-zero eigenvalue of $\boldsymbol{\Sigma}_{\boldsymbol{x}}$, can be easily computed by exploiting the property that the trace of a matrix equals the sum of its eigenvalues:

$$\zeta_x = \mathrm{tr}(\boldsymbol{\Sigma}_{\boldsymbol{x}}) = \sigma_s^2 \mathrm{tr}(\boldsymbol{h}\boldsymbol{h}^H) = \sigma_s^2 \|\boldsymbol{h}\|^2 \tag{2.41}$$

therefore,

$$\boldsymbol{\zeta}|_{\mathcal{H}_1} = \left[\sigma_s^2 \|\boldsymbol{h}\|^2 + \sigma_v^2,\ \sigma_v^2 \mathbf{1}_{1,K-1}\right] = \sigma_v^2 \left[K\rho + 1,\ \mathbf{1}_{1,K-1}\right] \tag{2.42}$$

RLRT uses as test statistics the ratio between the largest eigenvalue and the noise variance (semi-blind case), while GLRT the ratio between the largest eigenvalue and the average of all eigenvalues (blind case). It is evident that, when there is no signal, both ratios are equal to 1, whereas for any signal, even with very low SNR, both ratios are greater than 1.

This discrimination criterion becomes no longer exact when applied to the sample covariance matrix $\boldsymbol{R}$, because the eigenvalues λ_i of $\boldsymbol{R}$ have a probabilistic behaviour and their fluctuations get larger as N decreases. Hence, in order to assess the analytical performance of RLRT and GLRT, we need to use the recent RMT results on the asymptotic eigenvalue distribution for $K, N \to \infty$ of the sample covariance matrix $\boldsymbol{R}$ in both $\mathcal{H}_0$ and $\mathcal{H}_1$ hypothesis scenarios.

Asymptotic eigenvalue distribution under $\mathcal{H}_0$

Under $\mathcal{H}_0$, the columns of $\boldsymbol{Y}$ are zero-mean independent complex Gaussian vectors, hence the sample covariance matrix $\boldsymbol{R}$ is a complex Wishart matrix [56]. Very important results have been provided on Wishart matrices, like the limiting joint

eigenvalue distribution by Marchenko and Pastur [57], and more recently the marginal distribution of single ordered eigenvalues. In particular, the asymptotical value and the limiting distribution of the largest eigenvalue of $\boldsymbol{R}$ are of importance for our problem.

If we define

$$c \triangleq \frac{K}{N} \tag{2.43}$$

for $K, N \to \infty$ and $K \ll N$, and we define

$$\mu_+(c) \triangleq \left(c^{1/2}+1\right)^2 \tag{2.44}$$

$$\xi_+(c) \triangleq \left(c^{1/2}+1\right)\left(c^{-1/2}+1\right)^{1/3} \tag{2.45}$$

the following holds:

1. Amost sure convergence of the largest eigenvalue:

$$\lambda_1 \xrightarrow{a.s.} \sigma_v^2 \mu_+(c) \tag{2.46}$$

   derived from [57] and proved in [58].

2. Convergence in distribution of the largest eigenvalue:

$$N^{2/3}\frac{\lambda_1 - \sigma_v^2\mu_+(c)}{\sigma_v^2\xi_+(c)} \to \text{TW2} \tag{2.47}$$

   where TW2 denotes the second order Tracy-Widom distributionp. This convergence was proved under the assumption of Gaussian entries in [59, 60, 61] and generalized to the non-Gaussian case in [62].

The Tracy-Widom distributions were first introduced in [63] and the second order Tracy-Widom Cumulative Distribution Function for Gaussian Unitary Ensembles (GUE) is explicitly given by:

$$F_{TW2}(x) = \exp\left(-\int_x^\infty (s-x)q^2(s)\,ds\right) \tag{2.48}$$

where $q(s)$ is the unique solution to the Painlevé II differential equation

$$q''(s) = sq(s) + 2q^3(s) \tag{2.49}$$

satisfying the boundary condition

$$q(s) \sim \text{Ai}(s), \quad s \to +\infty \tag{2.50}$$

where $\text{Ai}(s)$ denotes the Airy special function, one of the two linearly independent solutions to the following differential equation:

$$z''(w) - wz(w) = 0 \tag{2.51}$$

Asymptotic eigenvalue distribution under $\mathcal{H}_1$

Under $\mathcal{H}_1$, the statistical covariance matrix $\boldsymbol{\Sigma}$ is a rank-1 perturbation of the identity matrix, therefore it belongs to the class of spiked population model. These models were first introduced in [60]. In order to reduce the sample covariance matrix $\boldsymbol{R}$ to the standard spiked model [64, 65, 66] we need to rewrite the received signal matrix $\boldsymbol{Y}$ as

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{Z} \tag{2.52}$$

where $\boldsymbol{T}$ is a $K \times (1+K)$ block matrix defined as

$$\boldsymbol{T} = \left[\begin{array}{c|c} \frac{\sigma_s}{\sigma_v}\boldsymbol{h} & \boldsymbol{I}_K \end{array} \right] \tag{2.53}$$

and $\boldsymbol{Z}$ a $(1+k) \times N$ block matrix defined as

$$\boldsymbol{Z} = \left[\begin{array}{c} \frac{\sigma_v}{\sigma_s}\boldsymbol{s} \\ \hline \boldsymbol{V} \end{array} \right] \tag{2.54}$$

the covariance matrix becomes

$$\boldsymbol{R} = \frac{1}{N}\boldsymbol{T}\boldsymbol{Z}\boldsymbol{Z}^H\boldsymbol{T}^H \tag{2.55}$$

Let $\tau_1 \geq \ldots \geq \tau_K$ be the eigenvalues of $\boldsymbol{T}\boldsymbol{T}^H$. The largest eigenvalue τ_1 is equal to:

$$\tau_1 = K\rho + 1 \tag{2.56}$$

and that the eigenvalues of $\boldsymbol{T}\boldsymbol{T}^H$ are:

$$\boldsymbol{\tau} = \frac{1}{\sigma_v^2}\boldsymbol{\zeta} = [K\rho + 1, \mathbf{1}_{1,K-1}] \tag{2.57}$$

as proved in [44].

We can now focus on the asymptotical value and limiting distribution of the largest eigenvalue of the sample covariance matrix $\boldsymbol{R}$ for $\mathcal{H}_1$ case. By recalling $c \triangleq \frac{K}{N}$ and $\mu_+(c) \triangleq \left(c^{1/2}+1\right)^2$, we define

$$\mu_s(\tau_1, c) \triangleq \tau_1 \left(1 + \frac{c}{\tau_1 - 1}\right) \tag{2.58}$$

and

$$\xi_s(\tau_1, c) \triangleq \tau_1 \sqrt{1 - \frac{c}{(\tau_1 - 1)^2}} \tag{2.59}$$

then the following holds:

1. As $N, K \to \infty$, the largest eigenvalue λ_1 of $\boldsymbol{R}$ almost surely converges to:

$$\begin{cases} \lambda_1 \xrightarrow{a.s.} \sigma_v^2 \mu_+(c), & \text{if } \tau_1 \leq 1 + c^{1/2} \\ \lambda_1 \xrightarrow{a.s.} \sigma_v^2 \mu_s(\tau_1, c), & \text{if } \tau_1 > 1 + c^{1/2} \end{cases} \tag{2.60}$$

   called *phase transition phenomenon* and proved in [64]. Hence a signal is identifiable in RLRT or GRLT only if

$$\tau_1 > 1 + c^{1/2} \tag{2.61}$$

2. Limiting distribution of λ_1 for identifiable signals.

$$N^{1/2} \frac{\lambda_1 - \sigma_v^2 \mu_s(\tau_1, c)}{\sigma_v^2 \xi_s(\tau_1, c)} \to \mathcal{N}(0,1), \qquad \text{if } \tau_1 > 1 + c^{1/2} \tag{2.62}$$

   where $\mathcal{N}(0,1)$ denotes a zero-mean unit-variance Gaussian distribution. This last convergence was proved in [65] for Gaussian signals and was generalized into this form in [66].

2.3.3 Roy's Largest Root Test (RLRT)

RLRT tests the largest eigenvalue of the sample covariance matrix against the noise variance. The test statistic is

$$T_{RLRT} = \frac{\lambda_1}{\sigma_v^2}. \tag{2.63}$$

Roy's Largest Root Test was originally derived by the union intersection principle in [42], and applied to CR in [67, 68]. For Gaussian signals and not too low signal-to-noise ratio, the RLRT is the best test statistics in this class.

Formulation of Detection and False alarm Probabilities

1. *False alarm probability*: Concerning the null hypothesis where $\boldsymbol{Y} = \boldsymbol{V}$, and the sample covariance matrix $\boldsymbol{R}$ follows a Wishart distribution of degree N. From (2.63), it is clear that, the false alarm rate depends upon the distribution of the largest eigenvalue λ_1 of the sample covariance matrix $\boldsymbol{R}$. According to the recent results involving RMT, the detection statistic T_{RLRT} under null hypothesis for sufficiently large N and K follows a Tracy-Widom distribution of order 2 (TW2) [60]. Thus,

$$\text{Prob}\left[\frac{T_{RLRT}|_{\mathcal{H}_0} - \mu}{\xi} < t\right] \to F_{TW2}(t) \tag{2.64}$$

where $F_{TW2}(t)$ is the CDF of the TW2 with suitably chosen centering and scaling parameters shown below,

$$\mu = \left[\left(\frac{K}{N}\right)^{\frac{1}{2}} + 1\right]^2 \tag{2.65}$$

$$\xi = N^{-2/3}\left[\left(\frac{K}{N}\right)^{\frac{1}{2}} + 1\right]\left[\left(\frac{K}{N}\right)^{-\frac{1}{2}} + 1\right]^{1/3} \tag{2.66}$$

hence, the probability of false alarm can be written as,

$$P_{fa} = \text{Prob}\left(P_{RLRT} > t|_{\mathcal{H}_0}\right) \tag{2.67}$$

$$= \text{Prob}\left(\frac{T_{RLRT}|_{\mathcal{H}_0} - \mu}{\xi} > \frac{t-\mu}{\xi}\right) \tag{2.68}$$

$$P_{fa} = 1 - F_{TW2}\left(\frac{t-\mu}{\xi}\right) \tag{2.69}$$

Hence an approximate expression for the threshold of RLRT is

$$t_{RLRT}(\alpha) \approx \mu + F_{TW2}^{-1}(1-\alpha)\xi \tag{2.70}$$

where F_{TW2}^{-1} is the inverse of the TW2 CDF.

2. *Detection probability*: Under alternate hypothesis, the asymptotic distribution of λ_1 in the joint limit $N, K \to \infty$ is characterized by a phase transition phenomenon for smaller SNR [64]. For single signal detection, the critical detection threshold can be expressed in terms of SNR as ,

$$\rho_{Cric} = \frac{1}{\sqrt{KN}} \tag{2.71}$$

which is of immediate proof by combining (2.56) and (2.61)[44]. In fact, this suggests us that when SNR is lower than the critical value, the limiting distribution of the detection static T_{RLRT} is same as that of the largest noise eigenvalue, thus nullifying the statistical power of a largest eigenvalue test. Thus, for $\rho > \rho_{Cric}$ the distribution of T_{RLRT} was found to be asymptotically Gaussian [68, 64] as shown below,

$$\frac{\lambda_1}{\sigma_v^2} \sim \mathcal{N}(\mu_1, \sigma_1^2) \tag{2.72}$$

where,

$$\mu_1 = (1 + K\rho)\left(1 + \frac{K-1}{NK\rho}\right) \tag{2.73}$$

$$\sigma_1^2 = \frac{1}{N}(K\rho + 1)^2\left(1 - \frac{K-1}{NK^2\rho^2}\right) \tag{2.74}$$

are refined expressions of (2.58) and (2.59) respectively, in which correction terms for finite N, K have been added [68]. Thus, the probability of detection of RLRT can be as shown below,

$$P_d = \text{Prob}\left[T_{RLRT}|_{\mathcal{H}_1} < t\right] \tag{2.75}$$

$$P_d = Q\left(\frac{t-\mu_1}{\sigma_1}\right) \approx Q\left[\sqrt{N}\left(\frac{t(\alpha)}{K\rho+1} - \frac{K-1}{NK\rho} - 1\right)\right] \tag{2.76}$$

2.3.4 Generalized Likelihood Ratio Test (GLRT)

GLRT uses as test statistic the ratio

$$T_{GLRT} = \frac{\lambda_1}{\frac{1}{K}\text{tr}(\boldsymbol{R})} = \frac{\lambda_1}{\frac{1}{K}\sum_{i=1}^{K}\lambda_i}. \tag{2.77}$$

The distribution of this ratio was derived in [69], performance analysis of GLRT can be found for example in [70].

It is interesting to note that the GLRT is equivalent (up to a nonlinear monotonic transformation) to [46]:

$$T_{GLRT'} = \frac{\lambda_1}{\frac{1}{K-1}\sum_{i=2}^{K}\lambda_i}. \tag{2.78}$$

The denominator of $T_{GLRT'}$ is the maximum-likelihood (ML) estimate of the noise variance assuming the presence of a signal, hence the GLRT can be interpreted as a largest root test with an estimated $\hat{\sigma}_v^2$ instead of the true σ_v^2.

Formulation of Detection and False Alarm Probabilities

1. *False alarm probability*: Asymptotically, as both $N, K \to \infty$, the random variable T_{GLRT} also follows a second-order TW distribution [45], hence in first approximation $t_{GLRT}(\alpha) \approx t_{RLRT}(\alpha)$. However, as described in [69], this approximation is not very accurate for tail probabilities of T_{GLRT} for small values of K. In [69] the following improved expression was derived:

$$\Pr\left[\frac{T_{GLRT}-\mu}{\xi} < s\right] \approx F_{TW2}(s) - \frac{1}{2NK}\left(\frac{\mu}{\xi}\right)^2 F''_{TW2}(s) \tag{2.79}$$

   The above equation can be numerically inverted to find the required threshold $t_{GLRT}(\alpha)$.

2. *Detection probability*: To derive an explicit approximate expression for the detection performance of the GLRT under $\mathcal{H}_1$, we note that

$$\frac{1}{K}\sum_{i=1}^{K}\lambda_i = \frac{1}{K}\left[\lambda_1 + \sum_{j=2}^{K-1}\lambda_i\right] \tag{2.80}$$

and we rewrite the GLRT (2.77) as

$$\lambda_1 > \tilde{t}(\alpha)\frac{\sum_{j=2}^{K}\lambda_i}{K-1} \tag{2.81}$$

with

$$\tilde{t}(\alpha) = \frac{K-1}{K - t_{GLRT}(\alpha)} t_{GLRT}(\alpha) \tag{2.82}$$

Assuming the presence of a sufficiently strong signal ($\rho > \rho_{crit}$), the largest sample eigenvalue is (with high probability) due to a signal whereas the remaining eigenvalues, $\lambda_2, \ldots, \lambda_K$, are due to noise. Let

$$Z \triangleq \frac{1}{K-1}\sum_{i=2}^{K}\lambda_i \tag{2.83}$$

denote their mean. Asymptotically in N, the random variable Z is Gaussian distributed with variance $O\left(\frac{1}{N(K-1)}\right)$, and with a mean value that is slightly biased downwards:

$$\mathrm{E}\left[\frac{Z}{\sigma_v^2}\right] = 1 - \frac{1}{N}\frac{K\rho+1}{K\rho} + O\left(\frac{1}{N^2}\right) \tag{2.84}$$

We then recall that λ_1/σ_v^2 is asymptotically Gaussian distributed with mean and variance given by (2.73) and (2.74). For a large number of sensors ($K \gg 1$), the fluctuations of Z are relatively much smaller than those of λ_1, hence we can approximate (2.81) as

$$(1+K\rho)\left(1+\frac{K-1}{NK\rho}\right) + \frac{1+K\rho}{\sqrt{N}}\eta_1 > \tilde{t}(\alpha)\cdot\mathrm{E}\left[\frac{Z}{\sigma_v^2}\right] \tag{2.85}$$

where $\eta_1 \sim \mathcal{N}(0,1)$ is a standard Gaussian random variable. Hence, we conclude that

$$P_d = \mathrm{Prob}\left[T_{GLRT}|_{\mathcal{H}_1} < t\right] \tag{2.86}$$

$$P_d \approx Q\left[\sqrt{N}\left(\tilde{t}(\alpha)\left(\frac{1}{K\rho+1} - \frac{1}{NK\rho}\right) - \frac{K-1}{NK\rho}1\right)\right] \tag{2.87}$$

Chapter 3

Sensing techniques for cognitive TV White-Spaces systems

This chapter describes and compares the performance of a set of spectrum sensing algorithms to be employed for the detection of OFDM-based (Ortogonal Frequency Division Multiplexing) transmissions in the TV bands (470-790 MHz) i.e., DVB-T channels. Spectrum sensing techniques take a crucial role to support geo-referenced TV White-Spaces (TVWS) databases and to maintain them up-to-date over time. When considering a single-antenna spectrum sensing unit, very effective methods for detecting OFDM signals are based on DVB-T Cyclic Prefix and pilot pattern feature detection. Starting from these, further improvements can be made using multi-antenna techniques. This chapter shows performance analysis of feature-based single-antenna and multi-antenna ED/EBD techniques in order to derive trade-offs and conclusions.

The huge interest in the TV White-Spaces (TVWS) availability has recently determined the development of novel Machine-to-Machine (M2M) standards such as the Weightless standard, secondary user rural broadband internet access (i.e., the IEEE 802.22 standard), and other proprietary protocols. In UK and USA the regulatory process is almost terminated, so that a huge number of large companies have demonstrated interest in the management of large TVWS databases. These databases are required to inform secondary users about the presence of primary transmitters and thus, must be maintained up-to-date over time and assessed by field test trials. In 2013 large-scale TVWS networks were deployed for end-user testing purposes in the USA and South Africa. The crucial importance of precise and up-to-date information provided by such TVWS databases represents one of the weak points of the future TVWS broadband networks. In order to guarantee the highest primary detection sensitivity, future TVWS secondary networks must rely on master units able to perform both TVWS database access and advanced spectrum sensing.

3.1 Primary signal

Currently, in the TVWS domain, the most relevant primary signal, is represented by DVB-T (ETSI Digital Video Broadcasting - Terrestrial) broadcast transmissions [71]. DVB-T transmissions consist of OFDM channels that are continuously occupied in time during the channel activity. From a spectrum sensing point of view, important DVB-T parameters (see also Tab. 3.1) are represented by: channel bandwidth (that ranges from 5 to 8 MHz), the OFDM Cyclic Prefix (CP) length (that ranges from 1/32 to 1/4 of the OFDM symbol length), and the presence of OFDM pilots (continual and scattered). The presence of pre-determined patterns (and their periodic repetition) in the transmitted DVB-T signal determines the cyclo-stationary property shown by OFDM signals.

	2k mode	8k mode
Symbol duration (T_U)	224μs	896μs
Guard interval duration (Δ)	$7-56\mu$s	$28-224\mu$s
Number of active subcarriers	1705	6817
Subcarrier spacing (approx.)	4464Hz	1116Hz
CP duration ratio (Δ/T_U)	1/4, 1/8, 1/16, 1/32	
Constellations	QPSK, 16-QAM, 64-QAM	
Code rate	1/2, 2/3, 3/4, 5/6, 7/8	

Table 3.1: Main parameters of DVB-T.

In order to accurately assess the performance of the studied sensing algorithms in a realistic scenario consisting of real DVB-T primary signals, a DVB-T SDR transmitter has been implemented by CSP - ICT Innovation in GNU Radio [14]. Two GNU Radio custom blocks have been developed: a DVB-T encoder and a DVB-T modulator. These feature a subset of the DVB-T physical layer parameters. The OFDM mode with 8k subcarriers and cyclic prefix 1/4 has been used in the spectrum sensing tests, which corresponds to the most used configuration for DVB-T in Europe. The DVB-T encoder block receives the input Transport Stream (TS) containing a set of multiplexed TV and radio channels and, after encoding, sends the stream to the modulator. The output of the OFDM modulator is the sampled complex envelope (I/Q stream) ready to be up-converted by a radio front-end device. The resulting bitrate is approximately 24.88 Mbits/s. At the sensing unit, the DVB-T signal was sampled at the nominal rate of 64/7 Msamples/s.The processing performed by the encoder and modulator blocks are resumed in Fig. 3.1 and 3.2. The whole project has been described in [72].

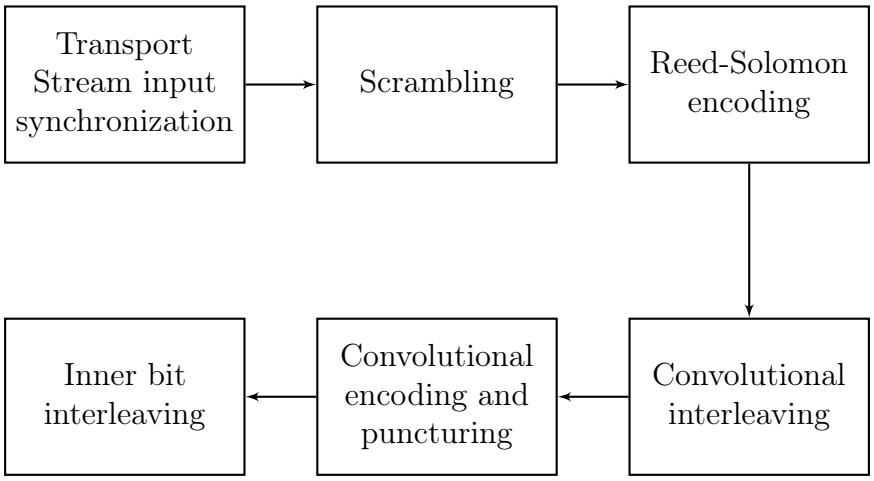


Figure 3.1: Encoder block.

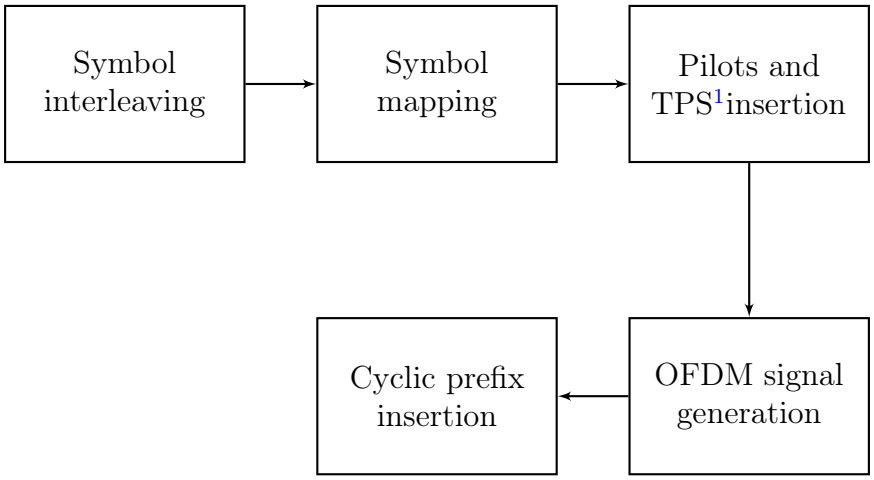


Figure 3.2: Modulator block.

As a common assumption in the literature on spectrum sensing, the primary signal is modelled as a Gaussian process. Fig. 3.3, 3.43.5 and 3.6 show that, in the case of DVB-T signals, this assumption is well motivated. In fact, Fig. 3.3 and 3.4 show the *pdf* of the real and imaginary parts of the DVB-T signal's complex envelope. Clearly, the Gaussian distribution is very well approximated. A more accurate evaluation is provided in Fig. 3.5 and 3.6, where the *quantile-quantile* plot

[1]Transmission Parameter Signalling

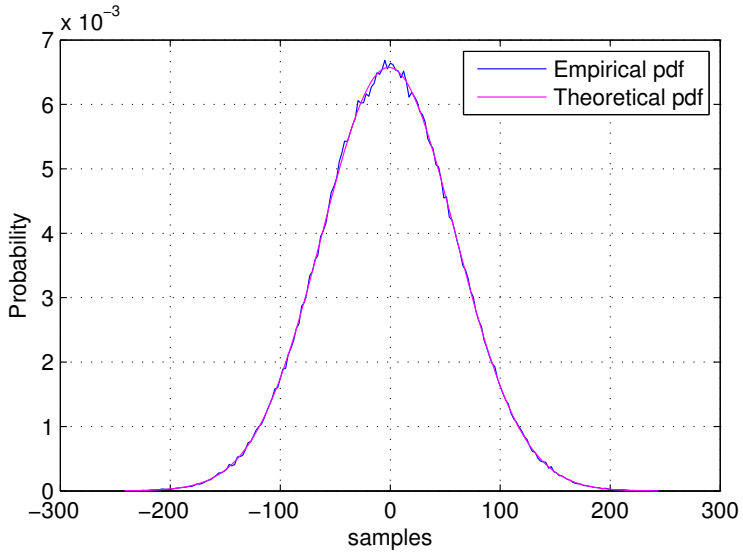


Figure 3.3: Estimated vs. theoretical *pdf* of the DVB-T signal, real part.

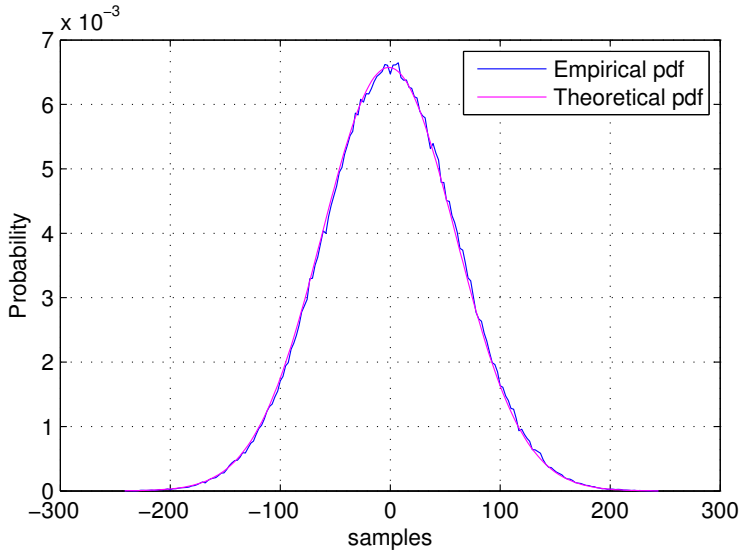


Figure 3.4: Estimated vs. theoretical *pdf* of the DVB-T signal, imaginary part.

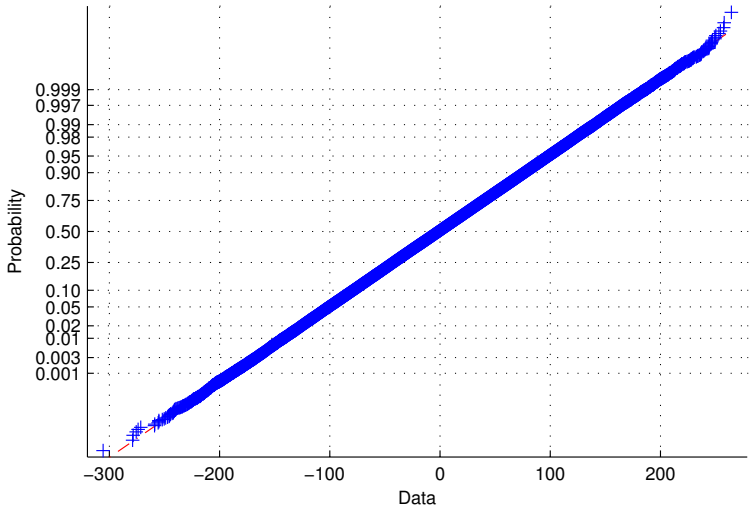


Figure 3.5: Quantile-quantile plot of the DVB-T signal, real part.

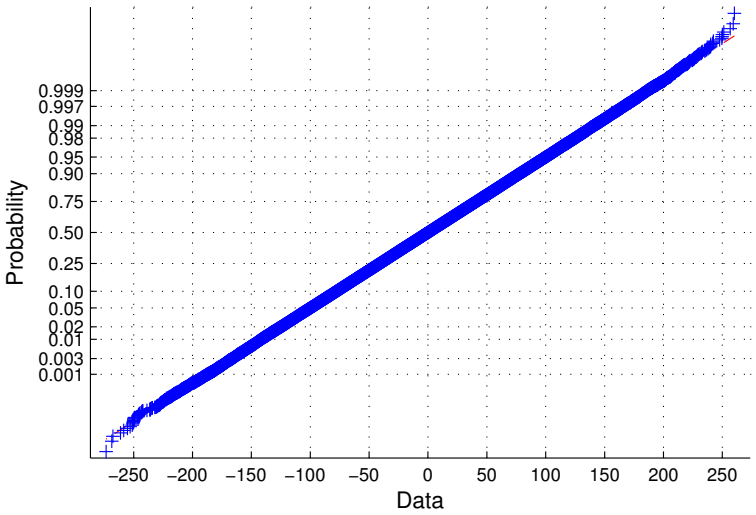


Figure 3.6: Quantile-quantile plot of the DVB-T signal, imaginary part.

of the DVB-T distribution vs. a zero-mean Gaussian distribution with same variance is shown.

3.2 Single antenna spectrum sensing algorithms

An effective approach to performing spectrum sensing in the TVWSs consists in the exploitation of the intrinsic structure of OFDM signals. Spectrum sensing techniques able to detect a number of "features" in the received signal can be devised. Such features must uniquely characterize the DVB-T transmission in order to allow efficient signal detection.

Single-antenna feature-based detectors exhibit much better performance in case of unknown noise variance (unlike energy detection-based techniques), but require much longer sensing time for synchronization in order to achieve such performance. In the following subsection, these single-antenna CP-based feature detectors will be analyzed and their performance assessed.

3.2.1 Cyclic prefix based detector

The cyclostationary properties of OFDM signals in DVB-T transmissions are due to the presence of cyclic repetitions of signal segments in the time domain.

We briefly recall the OFDM symbol structure in Fig. 3.7, where N_s is total number of samples per symbol, N_u is the number of useful samples and N_c the number of cyclic prefix samples.

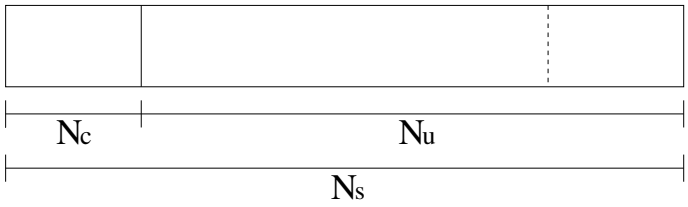


Figure 3.7: OFDM symbol structure with cyclic prefix.

The number of samples N_s, N_u and N_c can be obtained through the following relationships:

$$\begin{aligned} N_s &= N_{\text{mode}} \cdot (1 + \text{CP}) \cdot \frac{7}{8B} \cdot f_s \\ N_c &= N_{\text{mode}} \cdot \text{CP} \cdot \frac{7}{8B} \cdot f_s \\ N_u &= N_s - N_c \end{aligned} \tag{3.1}$$

where $N_{\text{mode}} = 8192$ for 8k mode and $N_{\text{mode}} = 2048$ for 2k mode. Moreover, $CP = N_c/N_u$ is the ratio of cyclic prefix duration and the OFDM symbol duration, B is the signal bandwidth in MHz, and f_c is the sampling frequency, in Msamples/s.

Fig. 3.8 shows the amplitude of the symbol autocorrelation after cyclic prefix insertion for 8k mode. Besides the peak at lag 0, the autocorrelation (AC) function shows two other peaks at $\pm N_d$ due to the cyclic prefix.

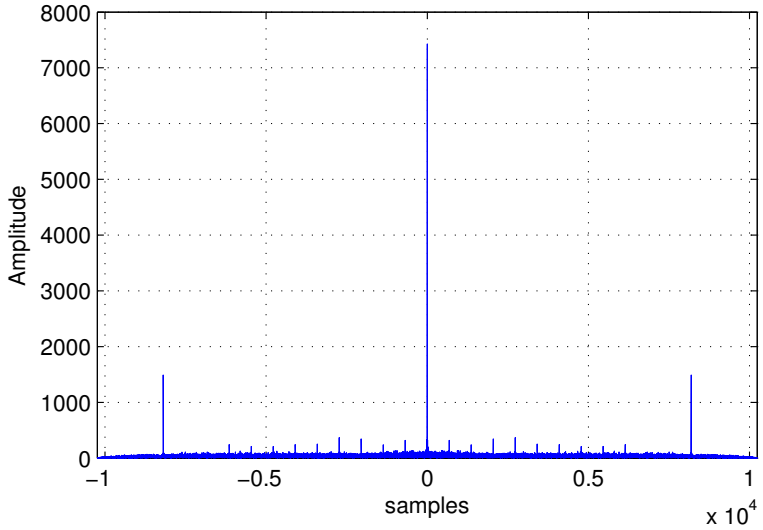


Figure 3.8: Symbol autocorrelation after CP insertion - Amplitude - 8k mode.

As already mentioned, the detector operates asynchronously with the primary signal. The first detection technique we present is based on the computation of the autocorrelation (AC) of the received signal in order to detect its cyclostationary features. The autocorrelation function must be computed in the range $[0, N_s]$ in order to include at least one cyclic repetition. The expression for the AC function used for detection is the following:

$$R_{xx}^{\text{CP}}[n] = \frac{1}{N_c} \left| \sum_{k=0}^{N_c-1} x^*[n-k]x[n-k-N_u] \right|. \tag{3.2}$$

where $x[n]$ is the received primary signal, modelled as

$$x[n] = p[n] + w[n]. \tag{3.3}$$

Here, $p[n]$ is the transmitted primary signal and $w[n]$ is additive white Gaussian noise and the SNR $\triangleq E[|x[n]|^2]/E[|w[n]|^2]$. If detection is performed in very low SNR conditions, it may be necessary to improve the sensitivity by extending the

observation window to K consecutive OFDM symbols (KN_s samples), thus the new AC function is

$$\widetilde{R}_{xx}^{\mathrm{CP}}[n] = \frac{1}{KN_c}\left|\sum_{j=0}^{K-1}\sum_{k=0}^{N_c-1} x^*[n-k-jN_s]x[n-k-jN_s-N_u]\right|. \tag{3.4}$$

which reduces to (3.2) for $K = 1$. The AC functions (3.2) and (3.4) exhibit a peak shape with the maximum value achieved synchronously with the end of the OFDM symbol. Based on this observation, the proposed test statistic for the CP-based detector based on (3.2) and (3.4) is:

$$T_{CP} = \frac{\max_i\{\widetilde{R}_{xx}^{\mathrm{CP}}[i]\}}{\mathbb{E}_j\{\widetilde{R}_{xx}^{\mathrm{CP}}[j]\}} \gtrless t_{CP}. \tag{3.5}$$

Here, the numerator denotes the peak of the whole autocorrelation function while the denominator denotes the temporal mean of the AC computed only at samples j, with $j \in J$, while t_{CP} denotes the threshold. If we denote with $\varphi = \mathrm{argmax}_n(\widetilde{R}_{xx}^{\mathrm{CP}}[n])$ the peak of the AC function, the set J is defined as

$$\begin{aligned} &j \in J = N \backslash Q \\ &N = \{n \in \mathbb{N} \ : \ 0 \leq n \leq KN_s\} \\ &Q = \{q \in \mathbb{N} \ : \ \varphi - N_c \leq q \leq \varphi + N_c\}. \end{aligned} \tag{3.6}$$

Basically the mean of the autocorrelation is computed for all those samples whose "distance" from the instant with the peak (maximum of the AC) is larger than N_c samples. This way, neither part of the cyclic prefix is comprised in the summation of (3.4) and hence the estimation of the correlation noise is improved. The aforementioned mean is used to perform the calibration of the threshold t_{CP} [73].

Fig. 3.9 shows the amplitude of the CP-based autocorrelation function with an observation window of respectively $K = 1$ and $K = 10$ symbols with an infinite SNR. Results with realistic SNR values will be given later in the following sections. We observe that, for $K = 10$, the ratio of the peak value to the maximum value observed outside the cyclic prefix window is improved with respect to $K = 1$.

3.2.2 Pilot-based detector

The DVB-T signal contains pilot subcarriers that can be used for channel estimation and signal synchronization at the receiver, since their values are known. We can define an expression of the OFDM signal in terms of complex envelope where only the pilot carriers are active, while data and Transmission Parameter Signalling (TPS) carriers are set to zero:

$$s_p(t) = e^{-j2\pi f_c t} \sum_{p,l,k:(l,k)\in A} c_{p,l,k} \cdot \Psi_{p,l,k}(t) \tag{3.7}$$

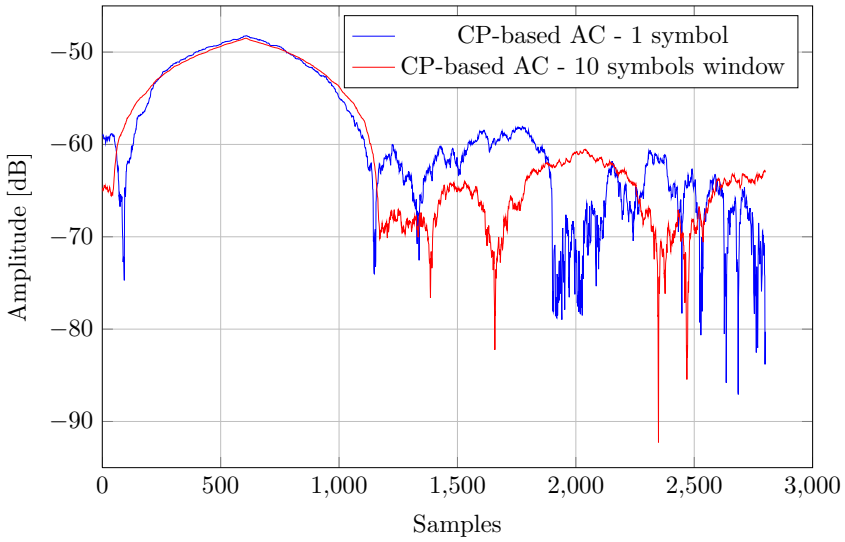


Figure 3.9: Amplitude of the CP-based auto-correlation, 1 symbol vs. 10 symbols observation.

where the set A defines the position of the continual and scattered pilot, while indices p, l and k correspond, respectively, to the DVB-T frame, to the OFDM symbol and to the subcarrier. $c_{p,l,k}$ is the pilot symbol and $\Psi_{p,l,k}(t)$ is the corresponding IFFT basis function. If such deterministic signal is sampled at the rate $1/T_s$, we obtain the sequence:

$$s_p[n] = s_p(nT_s), \qquad n = 0,1,\ldots \tag{3.8}$$

We know that continual pilots have the same carrier positions for all symbols, while scattered pilots take the same position every four symbols. Thus, since all symbols contain $N_u + N_c$ samples, the sequence $s_p[n]$ is cyclic with period $4(N_u + N_c)$, which means that it is completely determined by the vector: $(s_p[0], s_p[1], s_p[2], \ldots, s_p[4(N_u + N_c) - 1])$. We can therefore apply a *matched filtering* approach to detecting the DVB-T incumbent by defining the following test statistic:

$$T_P = \max_{\tau \in \{0,1,\ldots,4(N_u+N_c)-1\}} \left| \sum_{k=0}^{M-\tau-1} s_p[k] x^*[k+\tau] \right| \tag{3.9}$$

where M is total number of samples observed by the detector. As in the previous case, the determination of the threshold value must be performed empirically. In order to perform such computation, the detector is usually fed with pure white noise, whose variance is first estimated and then the test statistic is derived. Once the desired false alarm probability is chosen, the threshold is set accordingly [74] [11].

Fig. 3.10 shows the amplitude of the cross-correlation function shown in (3.9). The peak appears when our DVB-T signal and the only-pilot signal are synchronized.

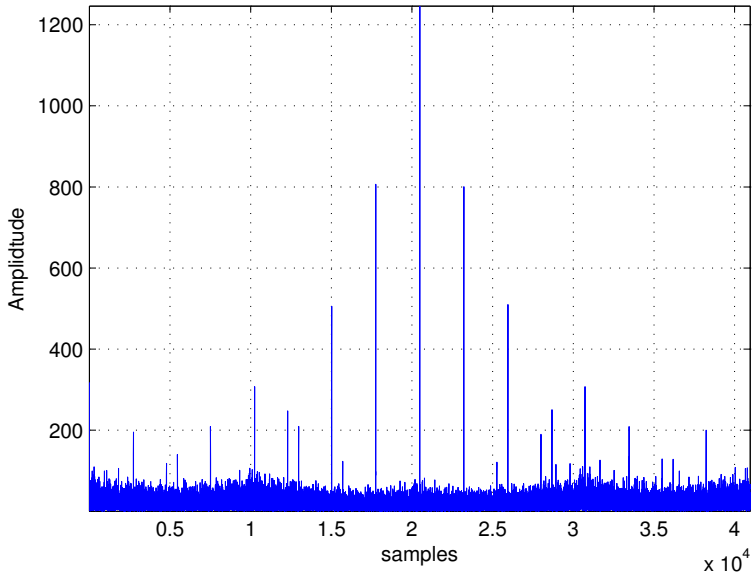


Figure 3.10: Cross-correlation between DVB-T signal and only-pilot signal.

3.3 Multi-antenna spectrum sensing algorithms

In this section we focus on a special class of detectors based on a multi-sensor/multi-antenna approach which will try to overcome the limits of single sensor detectors. As far as the feature detectors are concerned, such approaches are quite sensitive to synchronization errors. In case of very low SNR the synchronization loops might not be able to provide the required accuracy for the carrier frequency and clock rate estimates. All these reasons motivate the search for asynchronous multi-sensor detectors robust to noise uncertainty.

Multiple-input multiple-output (MIMO) technology has reached considerable maturity, since it is already part of many wireless standards (LTE, IEEE 802.11, IEEE 802.16, etc.) it is very likely for future CR terminals to incorporate it. The basic idea is to exploit the fact that, if the channel is being used by the PU, then some spatial correlation should be present in signals at different antennas. On the other hand, when the signal is absent, all contributions will correspond to thermal noise, thus spatial correlation should be absent [75].

All the presented detectors are non parametric, i.e., they don't assume any prior knowledge of the signal. All their test statistics will be expressed as functions of the eigenvalues of the *sample covariance matrix* of the received signals defined by a new matricial system model. All the eigenvalue-based algorithms are based on results from Random Matrix Theory (RMT) [76] [77]. Among them, we will present the energy detector for the multi-antenna eigenvalue-based case and methods based on likelihood ratio tests (LRT) or generalized LRT.

The algorithms will be divided in two classes:

1. those that assume a known noise level;

2. those that estimate it from the received signal.

3.3.1 System model

The system model used for ED/EBD algorithms is described in Sec. 2.2.1. We only recall that the detector computes its test statistic from K sensors or antennas and N time samples. The received samples are stored in the matrix

$$\boldsymbol{Y} = \boldsymbol{h}\boldsymbol{s} + \boldsymbol{V} \tag{3.10}$$

where *boldysmbols* represents the $1 \times N$ signal vector, which will contain 8k DVB-T samples in our simulation; $\boldsymbol{h}$ is a $K \times 1$ unknown complex channel vector. In this model a flat Rayleigh fading channel is considered, but for our simulation we considered also another channel model (the 6-path Typical Urban [78]); finally $\boldsymbol{V}$ is the noise matrix.

At this point we can define the sample covariance matrix $\boldsymbol{R}$ as follows:

$$\boldsymbol{R} \triangleq \frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^H. \tag{3.11}$$

Let $\lambda_1 \geq \ldots \geq \lambda_K$ be the eigenvalues of $\boldsymbol{R}$ sorted in decreasing order.

All the described test statistics are non-parametric, i.e., they do not assume any prior knowledge about the signal to be detected. In general all test have only two parameters besides the received samples used to construct the covariance matrix and to compute its eigenvalues: the number of samples N and the number of sensors K.

The methods are divided into two groups: methods for known or for unknown noise level. In the first group, the noise variance σ_v^2 is assumed to be known and appears explicitly in the test statistic. Methods belonging to the second group, on the contrary, do not require such information, i.e., the noise level is estimated in the test statistic.

3.3.2 Known noise variance algorithms

1. **Energy Detection (ED)**: see Sec. 2.3.1.

2. **Roy's Largest Root Test (RLRT)**: see Sec. 2.3.3.

3. **Likelihood Ratio Tests (LRT)**: different LRT-based detectors were given in [36]. The complete noise-dependent log-likelihood ratio test statistic is given by

$$T_{LRT} = 2(N-1)\left[\log\left(\frac{\sigma_v^{2K}}{\det \boldsymbol{R}}\right) + \left(\frac{\mathrm{tr}\boldsymbol{R}}{\sigma_v^2} - K\right)\right]. \tag{3.12}$$

   For this statistic, expressions of the false-alarm probability have been derived by means of numerical integration techniques. Performance analysis for this test can be found, for example, in [36].

3.3.3 Unknown noise variance algorithms

1. **Generalized Likelihood Ratio Test (GLRT)**: see Sec. 2.3.4.

2. **Eigenvalue Ratio Detector (ERD)**: the test statistic (also called maximum-minimum eigenvalue, or condition number test) is the ratio between the largest eigenvalue and the smallest eigenvalue of $\boldsymbol{R}$

$$T_{ERD} = \frac{\lambda_1}{\lambda_K}. \tag{3.13}$$

   A complete performance analysis can be found in [43, 39].

3. **Noise-independent LRT (LRT-)**: an alternative log-likelihood ratio was derived in [36], under the assumption of unknown noise variance:

$$T_{LRT-} = 2(N-1)\left[\frac{\frac{1}{K}\sum_{i=1}^{K}\lambda_i}{\left(\prod_{i=1}^{K}\lambda_i\right)^{1/K}}\right]^K. \tag{3.14}$$

   In statistics, this method has been known for many years as the *sphericity test* [79]. Performance analysis for cognitive radio applications plus an expression for the false alarm probability of this detector can be found in [36].

3.4 Channel models

We consider three different channel models. For each of them we describe how the $K \times N$ matrix $\boldsymbol{Y}$ of received samples has been calculated.

3.4.1 Additive White Gaussian Noise (AWGN) channel

In this case we generated a matrix $\boldsymbol{S}$ of size $K \times N$ where each row of $\boldsymbol{S}$ is equal to the $1 \times N$ signal vector $\boldsymbol{s}$. Hence:

$$\boldsymbol{Y} = \boldsymbol{S} + \boldsymbol{V} \tag{3.15}$$

This channel model will be used to evaluate the performance of the CP-based detector and to compare them against the eigenvalue-based detectors with unknown noise variance.

3.4.2 Flat Rayleigh fading channel

With this model we assume that the coherence bandwidth, defined as the inverse of the channel time dispersion, is much larger than the signal bandwidth. Under this assumption, our $1 \times N$ signal vector $\boldsymbol{s}$ is simply multiplied by a complex constant modelled as a Rayleigh random variable. We will have K independent random variables, one for each sensor, represented by the $K \times 1$ channel vector. Hence, as already presented in Sec. 2.2.1:

$$\boldsymbol{Y} = \boldsymbol{h}\boldsymbol{s} + \boldsymbol{V}. \tag{3.16}$$

In addition, the following normalization has been performed:

$$\sum_{i=1}^{K} h_i h_i^* = K \tag{3.17}$$

hence the energy of the channel vector is normalized to the number K of antennas.

3.4.3 Typical Urban 6-path (TU6) channel

This channel models the terrestrial propagation in an urban area. It has been defined by the COST (European Cooperation in Science and Technology) Action 207 as a typical urban (TU6) profile and is made of six paths having wide dispersion in delay and relatively strong power [78].

This model is a frequency- and time-selective Rayleigh multipath fading channel model. Given the input signal $x(t)$, the output signal $y(t)$ can be expressed as follows:

$$y(t) = \sum_{i=1}^{M} \gamma_i e^{-j\theta_i} x(t - \tau_i) \tag{3.18}$$

where:

- M is the number of paths equal to 6;
- γ_i is the average path gain of the ith path (listed in Tab. 3.2);
- θ_i is the phase shift from scattering of the i'th path, modelled as a uniformly distributed random variable in $[-\pi, \pi]$;
- τ_i is the relative delay of the ith path (listed in Tab. 3.2);

Tap number	Delay τ_i (µs)	Average gain γ_i (dB)	Doppler spectrum
1	0.0	-3	Classical
2	0.2	0	Classical
3	0.5	-2	Classical
4	1.6	-6	Classical
5	2.3	-8	Classical
6	5.0	-10	Classical

Table 3.2: Typical Urban profile (TU6).

where the classical doppler spectrum is defined as:

$$G(f; f_D) = \frac{1}{\sqrt{1 - (f/f_D)^2}} \tag{3.19}$$

In our simulations the Doppler spread f_D has been set to 10 Hz, corresponding to a pedestrian mobile profile.

This channel has been implemented in our simulation environment as a *Finite Impulse Response* (FIR) filter. We generated K realizations of this channel and performed the convolution (filtering) with the input signal for each realization. We generated a $K \times N$ matrix X where each row corresponds to N samples of our K filtered signals. Hence the model yields:

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{V} \tag{3.20}$$

By storing all K channel realization in a $K \times 6$ matrix $\boldsymbol{H}$, the following normalization has been performed:

$$\|\boldsymbol{H}\|_F^2 = \sum_{i=1}^{K} \sum_{j=1}^{6} h_{ij} h_{ij}^* = K \tag{3.21}$$

i.e., we performed the same normalization as for the flat fading case.

3.5 Performance assessment and trade-offs

The performance of the different sensing methods on the various channels have been evaluated by simulation (performed within a Matlab environment). For each simulation we computed:

- the ROC (Receiver Operating Characteristic) curve obtained by plotting the detection probability versus the false alarm probability;
- the detection probability as a function of the signal-to-noise ratio, with fixed false alarm probability $P_{fa} = 0.01$.

For our simulations we used the Monte Carlo method. In order to estimate the values of P_{fa} and P_d we performed $N_T = 10000$ trials for each SNR value.

For each trial we generated two instances of the matrix $\boldsymbol{Y}$: the first one is computed as in (3.15), (3.16) or (3.20) (signal plus noise case); the second one instead as $\boldsymbol{V} = \boldsymbol{N}$ (only noise case). This way, we computed two instances of the covariance matrix $\boldsymbol{R}$ and two test statistics for each algorithm: T_1 and T_0 respectively. Once all the trials have been performed, we generated a vector of threshold values from the smallest T_0 to the largest T_1 statistic. At this point we simply computed each ith element of the P_{fa} vector by counting how many T_0 values are greater than the ith threshold value. Similarly, each ith element of the P_d vector is computed by counting how many T_1 values are greater than the ith threshold value. Each value of both P_{fa} and P_d vectors are finally divided by the number of trials. Algorithm 1 shows a short description of the simulation algorithm in pseudocode.

3.5.1 Results

All the ROC performance curves have been evaluated at SNR = -10 dB. Such a challenging scenario corresponds to the so-called "hidden node problem" in the Wireless Regional Access Network (WRAN) cognitive radio scenario and has been chosen to emphasize the differences among the methods.

First of all, the performance of eigenvalue-based algorithms has been evaluated and compared with different sets of parameters: by default we assumed $N = 50$ stored samples for each antenna and $K = 10$ sensors.

Eigenvalue-based algorithms

Fig. 3.11 and Fig. 3.12 refer to a DVB-T 8k signal with $N = 50$ samples and $K = 10$ sensors. In both cases we observe that the best algorithm under known noise variance is the RLRT, while GLRT is the best under unknown variance. These results are in agreement with the results provided in the literature for Gaussian primary signals.

Algorithm 1 Simulation algorithm in pseudocode.

$N_B = 1000$ ▷ N_B = number of threshold values
for all *SNR* values **do**
  for $i = 1 \rightarrow N_T$ **do**
    compute σ_s^2
    compute σ_v^2
    generate $K \times N$ random Gaussian noise matrix $\boldsymbol{V}$ as a function of *SNR*
    if *chan* = AWGN **then**
      $\boldsymbol{Y}_1 \leftarrow \boldsymbol{S} + \boldsymbol{V}$ ▷ $\boldsymbol{Y}_1$ = signal+noise $\boldsymbol{Y}$
    else if *chan* = flat-fading **then**
      $\boldsymbol{Y}_1 \leftarrow \boldsymbol{hs} + \boldsymbol{V}$
    else ▷ *chan* = TU6
      $\boldsymbol{Y}_1 \leftarrow \boldsymbol{X} + \boldsymbol{V}$
    end if
    $\boldsymbol{Y}_0 \leftarrow \boldsymbol{V}$ ▷ $\boldsymbol{Y}_0$ = only-noise $\boldsymbol{Y}$
    compute $\boldsymbol{R}_1 \leftarrow (1/K)\boldsymbol{Y}_1\boldsymbol{Y}_1^H$ ▷ signal+noise
    compute $\boldsymbol{R}_0 \leftarrow (1/K)\boldsymbol{Y}_0\boldsymbol{Y}_0^H$ ▷ noise-only
    for all detectors **do**
      compute signal+noise test statistic $T_1(i)$
      compute only-noise test statistic $T_0(i)$
    end for
  end for
  for all detectors **do**
    create vector of thresholds $\boldsymbol{t}$ of N_B equally spaced values from $\min(\boldsymbol{T}_0)$ to $\max(\boldsymbol{T}_1)$
    create vector of N_B elements $\boldsymbol{P_{fa}}$
    create vector of N_B elements $\boldsymbol{P_d}$
    for $i = 1 \rightarrow N_B$ **do**
      for $j = 1 \rightarrow N_T$ **do**
        if $T_0(j) > t(i)$ **then**
          $P_{fa}(i) \leftarrow P_{fa}(i) + 1$
        end if
        if $T_1(j) > t(i)$ **then**
          $P_d(i) \leftarrow P_d(i) + 1$
        end if
      end for
      $P_{fa}(i) \leftarrow P_{fa}(i)/N_T$
      $P_d(i) \leftarrow P_d(i)/N_T$
    end for
  end for
end for

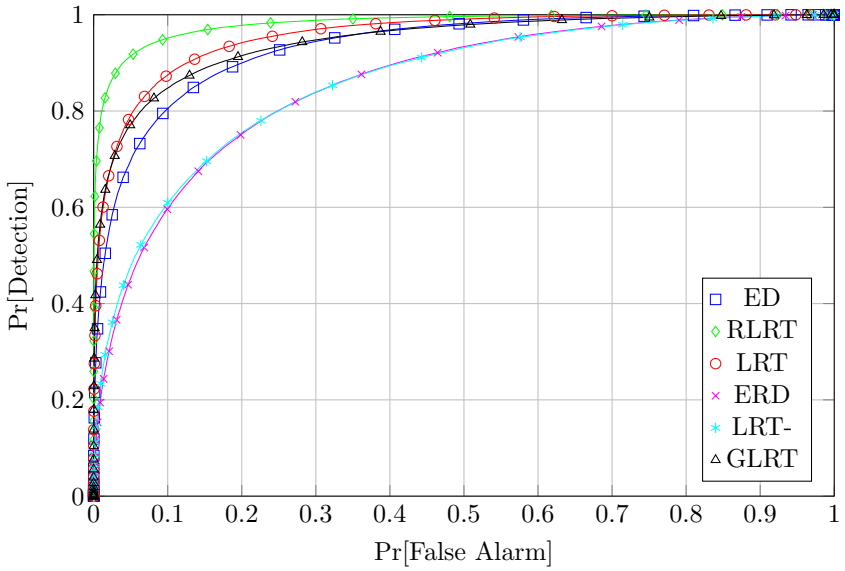


Figure 3.11: Eigenvalue-based detectors, DVB-T 8k PU signal, flat fading channel, $N = 50$, $K = 10$, ROC curves (SNR = -10dB).

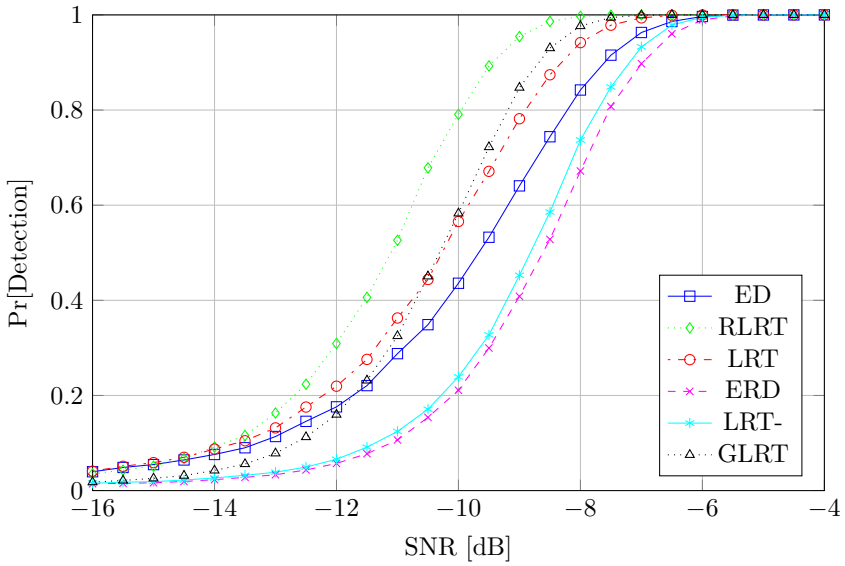


Figure 3.12: Eigenvalue-based detectors, DVB-T 8k PU signal, flat fading channel, $N = 50$, $K = 10$, P_d vs. SNR ($P_{fa} = 0.01$).

Secondly, in Fig. 3.13 and 3.14 the observation interval has been increased to $N = 200$, whereas the number of sensors reduced to $K = 4$.

Finally, in Fig. 3.15 and 3.16 we plot the detection probability of GLRT as a function of the observation interval (expressed both in time units and number of received samples per sensor) and the number of sensors respectively for SNR value of -10dB and -15dB, while the false alarm probability remains fixed to 10^{-2}. The channel is Rayleigh flat-fading. Under a more realistic model, the TU6 channel, the performance of the algorithms are different, as it can be observed in Fig. 3.17 and 3.18. We can see how both GLRT and RLRT lose their predominant position when the received model is different from the linear mixture one: simple energy detection becomes highly competitive in this case. The difference between algorithms with known and unknown noise variance is larger as well.

CP-based vs. eigenvalue-based (unknown σ_v^2) - AWGN channel

In this last section we compare the eigenvalue-based algorithms for unknown noise variance against the technique exploiting the cyclic prefix autocorrelation of the received signal. Here, the AWGN channel model is adopted. In Fig. 3.19 and 3.20 we can observe that the performance of this algorithm is similar to that of the GLRT. This single-antenna algorithm does not require the computation of the sample covariance matrix eigenvalues, but resorts to a precise knowledge of the signal characteristics.

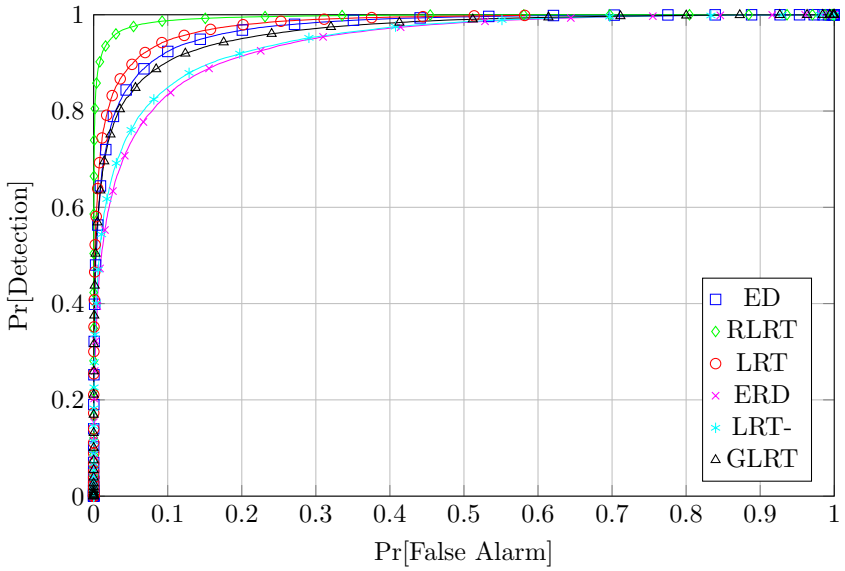


Figure 3.13: Eigenvalue-based detectors, DVB-T 8k PU signal, flat fading channel, $N = 200$, $K = 4$, ROC curves (SNR = -10dB).

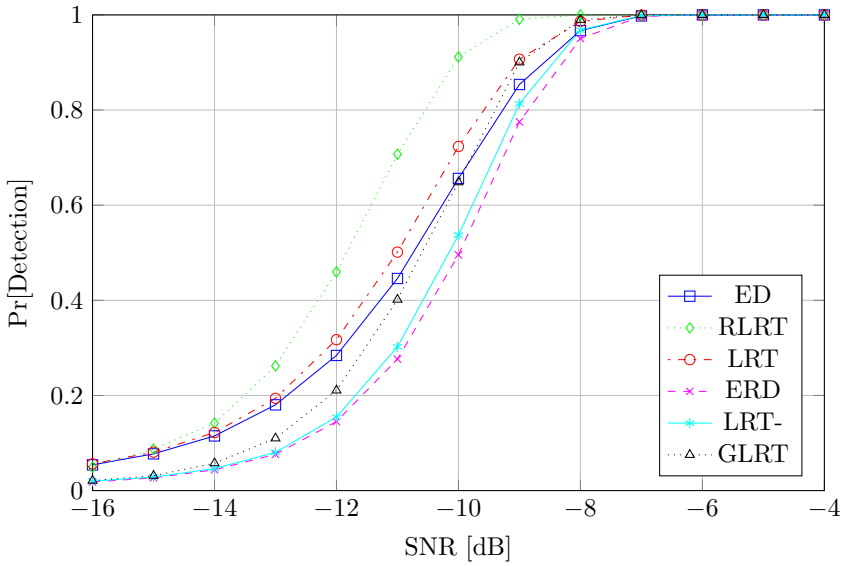


Figure 3.14: Eigenvalue-based detectors, DVB-T 8k PU signal, flat fading channel, $N = 200$, $K = 4$, P_d vs. SNR ($P_{fa} = 0.01$).

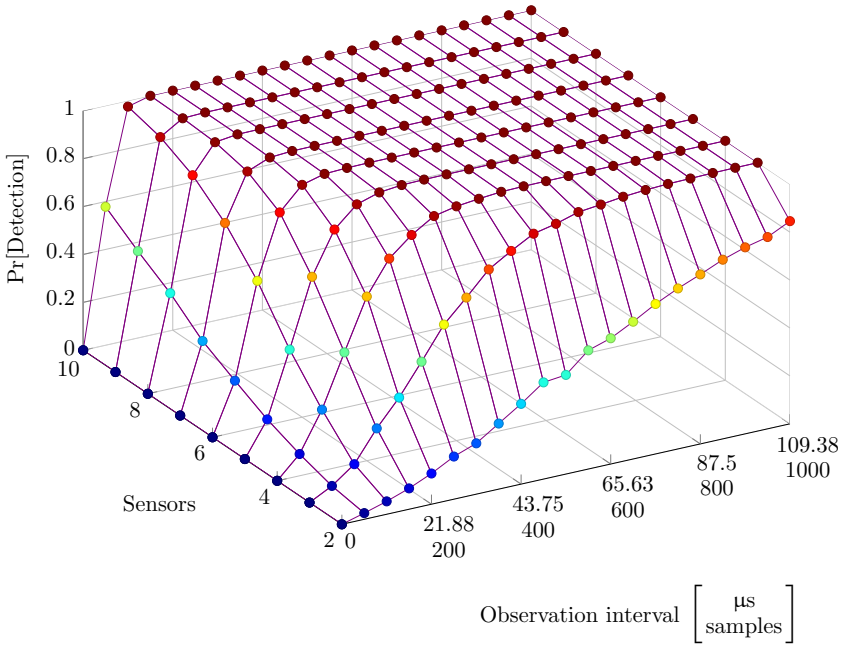


Figure 3.15: GLRT detection probability as a function of time (samples) and sensors through flat-fading channel, SNR = -10dB, $P_{fa} = 0.01$.

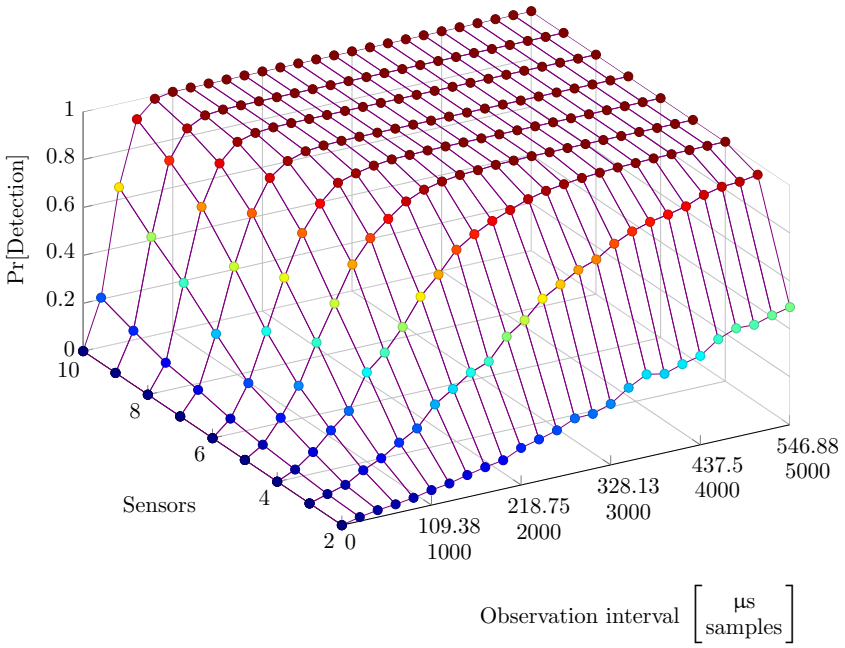


Figure 3.16: GLRT detection probability as a function of time (samples) and sensors through flat-fading channel, SNR = -15dB, $P_{fa} = 0.01$.

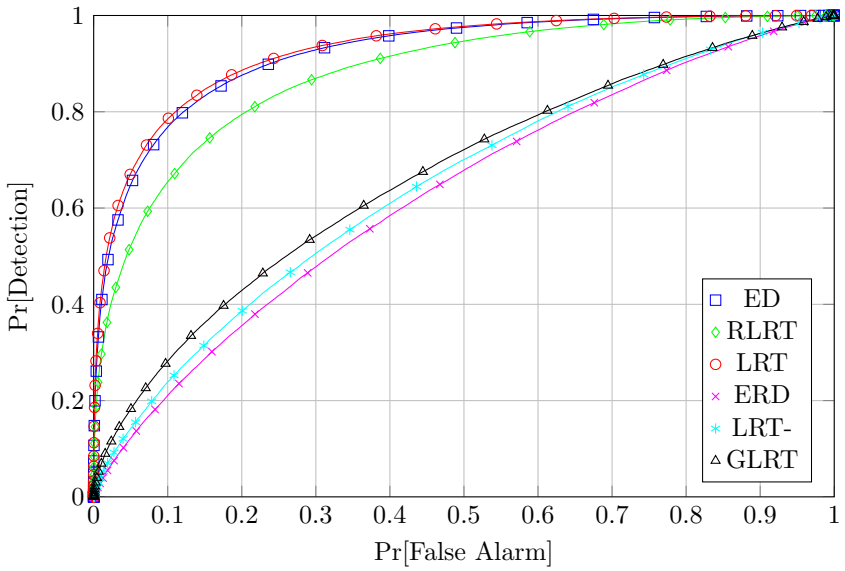


Figure 3.17: Eigenvalue-based detectors, DVB-T 8k PU signal, TU6 channel model, $N = 50$, $K = 10$, ROC curves (SNR = -10dB).

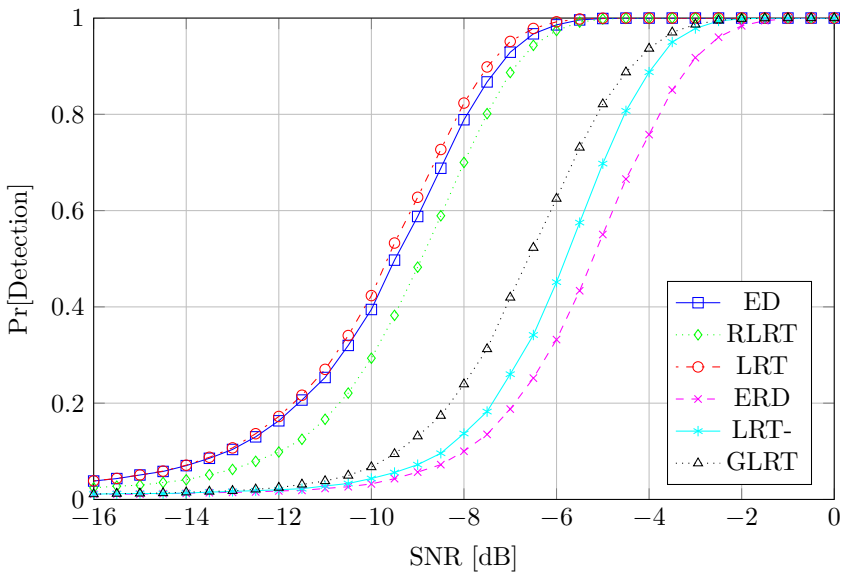


Figure 3.18: Eigenvalue-based detectors, DVB-T 8k PU signal, TU6 channel model, $N = 50$, $K = 10$, P_d vs. SNR ($P_{fa} = 0.01$).

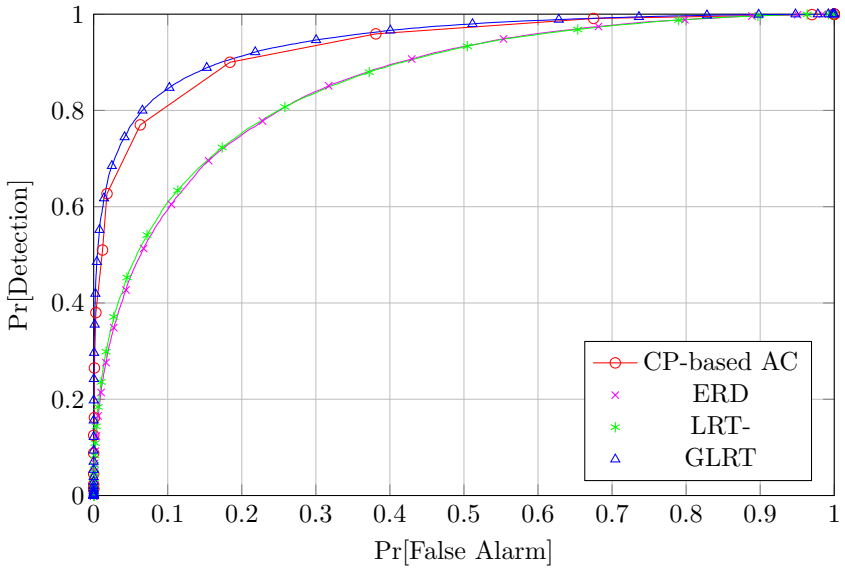


Figure 3.19: CP-based vs. eigenvalue-based (unknown σ_v^2) detectors, DVB-T 8k PU signal, flat fading channel, $N = 50$, $K = 10$, ROC curves (SNR = -10dB).

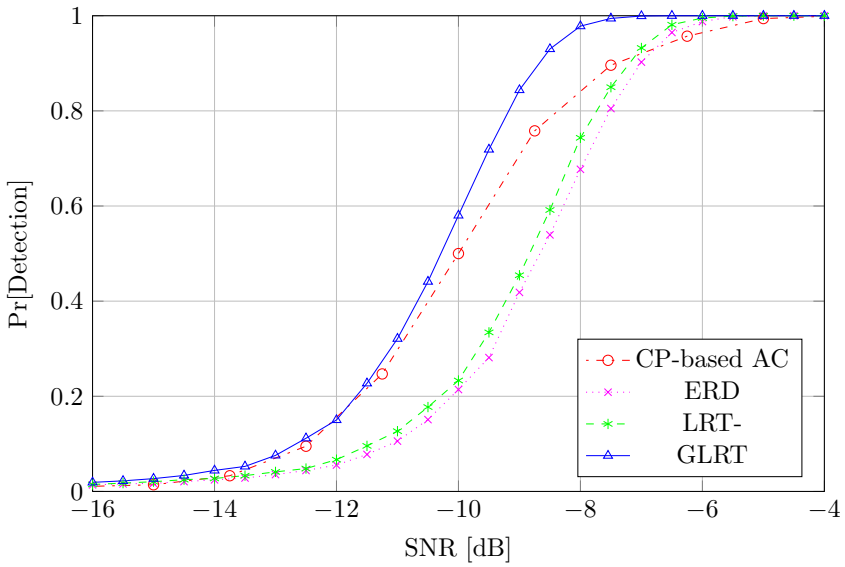


Figure 3.20: CP-based vs. eigenvalue-based (unknown σ_v^2) detectors, DVB-T 8k PU signal, flat fading channel, $N = 200$, $K = 4$, P_d vs. SNR ($P_{fa} = 0.01$).

Chapter 4

Hybrid Energy and Eigenvalue-based Detection algorithms with noise variance estimation

Semi-blind spectrum sensing algorithms, i.e., ED and RLRT, are the optimum spectrum sensing techniques in a known noise power level scenario. However, in real systems the detector does not have a prior knowledge of the noise level. In recent years, variation and unpredictability of the precise noise level at the sensing device came as a critical issue, which is also known as noise uncertainty.

This chapter presents an idea of auxiliary noise variance estimation and focuses on the performance evaluation of Hybrid Approach of semi-blind detection algorithms, namely ED and RLRT, using the same estimated noise variance.

4.1 Noise estimation

It is evident that the knowledge of the noise power is imperative for the optimum performance of both ED and RLRT. Unfortunately, the variation and the unpredictability of noise power is unavoidable. Thus, the knowledge of the noise power is one of the critical limitations especially of semi-blind detection algorithms for their operation in low SNR.

4.1.1 Offline noise estimation: hybrid approach 1

In the first type of hybrid approaches (HED1 and HRLRT1), noise variance is estimated from S auxiliary noise-only slots in which we are sure that the primary signal is absent.

Consider a sampling window of length M prior and adjacent to the detection window containing noise-only samples for sure. Then, the estimated noise variance from the noise-only samples using a Maximum Likelihood noise power estimation can be written as,

$$\hat{\sigma}_{v1}^2 = \frac{1}{KM} \sum_{k=1}^{K} \sum_{m=1}^{M} |v_k(m)|^2 \tag{4.1}$$

If the noise variance is constant, the estimation can be averaged over S successive noise-only slots and (4.1) can be modified by averaging over S successive noise-only slots as,

$$\hat{\sigma}_{v_1}^2(S) = \frac{1}{KSM} \sum_{s=1}^{S} \sum_{k=1}^{K} \sum_{m=1}^{M} |v_k(m)|^2 \tag{4.2}$$

A possible scheme of RLRT/ED detection algorithm using offline noise estimation approach is shown in Fig. 4.1: t_{tot} represents a periodic time interval divided into a training phase (noise estimation) and a runtime phase (detection). The runtime interval can be much longer than the training one, however the noise estimation needs to be updated after t_{tot}.

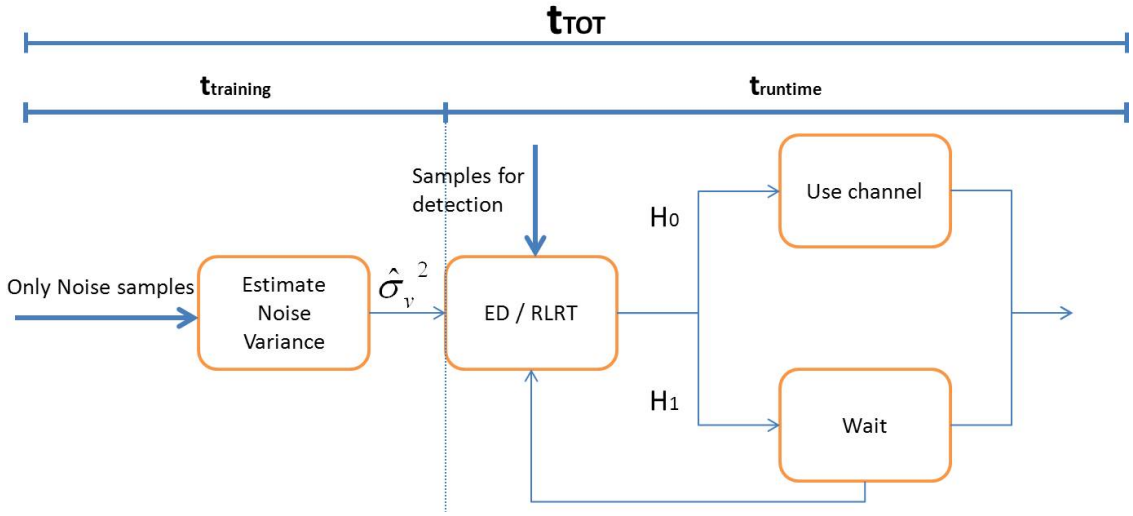


Figure 4.1: HED1/HRLRT1 with offline noise estimation approach.

4.1.2 Online noise estimation: hybrid approach 2

In a real time scenario, it is difficult to guarantee the availability of signal free samples so as to estimate the noise variance. Some literature analyzed the performance of ED using estimated noise variance setting aside a separate frequency channel for the measurement of the noise power [80]. However, it is not always suitable to assume uniformly distributed noise in all the frequency bands of concern.

The second hybrid approach does not resort to the existence of auxiliary noise-only slots, but estimates the noise variance information from the previous slots declared as $\mathcal{H}_0$ by the algorithm. Now, the noise variance estimated from those S auxiliary noise-only slots (previously declared $\mathcal{H}_0$) is used in the following detection interval to get the decision about the presence or absence of the primary signal.

Given P_S the probability of alternate hypothesis ($\mathcal{H}_1$), P_d is probability of detection, and S is the number of slots, the Maximum Likelihood noise variance estimate $\hat{\sigma}^2_{v_2}(S)$ using M received signal samples declared noise samples by the detector from K receivers is given by,

$$\frac{\left[\sum_{s=1}^{S_s}\sum_{k=1}^{K}\sum_{m=1}^{M}|h_k s(m)+v(m)|^2+\sum_{s=1}^{S_N}\sum_{k=1}^{K}\sum_{m=1}^{M}|v_k(m)|^2\right]}{KMS} \tag{4.3}$$

where, $S_S = SP_S\,(1-P_d)$ is the number of primary signal slots missed by the detector and $S_N = S - S_S$ is the number of noise samples successfully detected.

Fig. 4.2 shows a possible scheme of RLRT/ED detection algorithm using online noise estimation approach; after a transient stage (offline noise estimation), the detector automatically updates the noise estimation after S slots declared $\mathcal{H}_0$ (sliding window). Unlike the first approach, no further training offline phases are required.

4.2 Hybrid Energy Detection

Incorporating the *offline noise estimation* and *online noise estimation* described in Sec. 4.1 in ED, hybrid approaches of ED are developed and their performance parameters are derived in the following subsections.

4.2.1 Hybrid ED approach 1 (HED1)

The Energy Detection Test Statistic in (2.14) can be modified to HED1 test statistic using (4.2) as,

$$T_{HED1} = \frac{1}{KN\hat{\sigma}^2_{v_1}(S)}\sum_{k=1}^{K}\sum_{n=1}^{N}|y_k(n)|^2 \tag{4.4}$$

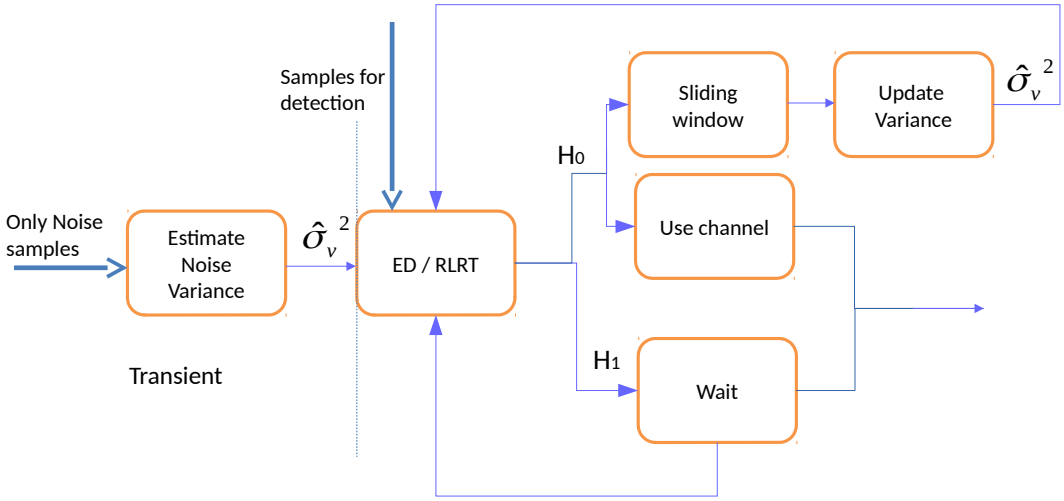


Figure 4.2: HED2/HRLRT2 with online noise estimation approach.

Moreover, (4.4) can be considered as the parametric likelihood ratio test when the signal to be detected is assumed to be Gaussian with zero mean and variance σ_s^2.

Proposition 1. Under null hypothesis, after rigorous simplification, the test statistic in (4.4) could be approximated with a normal random variable whose probability of false alarm P_{fa}^{HED1} for number of sensors K, number of samples N, number of auxiliary slots S and threshold t_{hed1} is given by,

$$P_{fa}^{HED1} = Q\left[\frac{t_{hed1} - 1}{\sqrt{\frac{MS+Nt_{hed1}^2}{KMNS}}}\right] \tag{4.5}$$

Similarly, under alternate hypothesis, the test statistic in (4.4) also approximates to a normal random variable with different mean and variance parameters whose probability of detection P_d^{HED1} could be written as,

$$P_d^{HED1} = Q\left[\frac{(t_{hed1} - 1 - \rho)}{\sqrt{\frac{t_{hed1}^2}{KMS} + \frac{K\rho^2+2\rho+1}{KN}}}\right] \tag{4.6}$$

Proof. Under null hypothesis, after normalizing the Gaussian noise samples at the numerator and denominator by σ_v^2, (4.4) can be approximated with a chi-squared random variable as shown below,

$$T_{HED1}|_{\mathcal{H}_0} = \frac{2KMS}{2KN}\left(\frac{\chi^2_{2KN}}{\chi^2_{2KMS}}\right) \tag{4.7}$$

For larger values of $2KN$ and $2KMS$, normally > 50, using Lyapunov Central Limit Theorem, the chi-squared random variable in (4.7) can be approximated with a normal random variable as,

$$T_{HED1}|_{\mathcal{H}_0} = \mathcal{N}\left(1, \frac{1}{KN} + \frac{t^2_{hed1}}{KMS}\right) \tag{4.8}$$

Similarly, under alternate hypothesis, after normalizing the Gaussian signal plus noise samples at the numerator and Gaussian noise samples at the denominator by the respective variances, the expression in (4.4) can be approximated with a chi-squared random variable as,

$$T_{HED}|_{H_1} = \frac{KMS}{KN}\left[\frac{K\rho\chi^2_{2N} + \chi^2_{2KN}}{\chi^2_{2KMS}}\right] \tag{4.9}$$

Again, with Normal approximation under sufficiently large degree of freedom of chi-squared random variables, detection statistic of HED1 under alternate hypothesis can be approximated with,

$$T_{HED1}|_{\mathcal{H}_1} = \mathcal{N}\left((\rho+1), \frac{t^2_{hed1}}{KMS} + \frac{K\rho^2 + 2\rho + 1}{KN}\right) \tag{4.10}$$

Finally, for Normal approximations of T_{HED1} in (4.8) and (4.10), using the property of normal random variable

$$\text{Prob}\left\{\mathcal{N}\left(\mu, \sigma^2\right) > t\right\} \cong Q\left(\frac{t-\mu}{\sigma}\right) \tag{4.11}$$

we obtain (4.5) and (4.6) as the expressions of false alarm and detection probabilities. □

4.2.2 Hybrid ED approach 2 (HED2)

Using (4.3), decision statistic of HED2 can be written as,

$$T_{HED2} = \frac{1}{KN\hat{\sigma}^2_{v_2}(S)}\sum_{k=1}^{K}\sum_{n=1}^{N}|y_k(n)|^2 \tag{4.12}$$

Proposition 2. Under null hypothesis, after rigorous simplification, the test statistic in (4.12) could be approximated with a normal random variable whose false alarm probability P_{fa}^{HED2} for number of sensors K, number of samples N, number of auxiliary slots S for noise estimation using (4.3) and threshold t_{hed2} is given by,

$$P_{fa}^{HED2} = Q\left[\frac{t_{hed2} - \frac{S}{S+\rho S_S}}{\sqrt{\frac{t_{hed2}^2 NC + MS^2}{KMN(S+\rho S_S)^2}}}\right] \tag{4.13}$$

where, $C = (S_S K\rho^2 + \rho S_S + S)$.

Similarly, under alternate hypothesis, the test statistic in (4.12) also approximates to normal random variable with different mean and variance parameters whose probability of detection P_d^{HED2} in a similar scenario could be written as,

$$P_d^{HED2} = Q\left[\frac{t_{hed2} - \frac{S(\rho+1)}{S+\rho S_S}}{\sqrt{\frac{t_{hed2}^2 NC + MS^2(K\rho^2+2\rho+1)}{KMN(S+\rho S_S)^2}}}\right] \tag{4.14}$$

Proof. Under null hypothesis, after normalizing the samples in the summation of (4.12) by their respective variances, the detection statistic $T_{HED2}|_{\mathcal{H}_0}$ can be simplified using a chi-squared random variable as,

$$T_{HED2}|_{H_0} = \frac{2KN}{2MKS}\frac{\left[\rho\chi^2_{2MS_S} + \chi^2_{2MKS_S} + \chi^2_{2MKS_N}\right]}{\chi^2_{2KN}} \tag{4.15}$$

Under a sufficiently large degree of freedom, we can easily note that the chi-square random variable can be approximated with a normal random variable $\chi_N^2 \sim \mathcal{N}(N, 2N)$, thus, approximating (4.15) as,

$$T_{HED2}|_{\mathcal{H}_0} = \frac{\mathcal{N}\left(1, \frac{1}{KN}\right)}{\mathcal{N}\left(\frac{S+\rho S_S}{S}, \frac{C}{KMS^2}\right)} \tag{4.16}$$

For alternate hypothesis, after similar normalization and normal approximation the test statistic $T_{HED2}|_{\mathcal{H}_1}$ can be written as,

$$T_{HED2}|_{\mathcal{H}_1} = \frac{\mathcal{N}\left(\rho+1, \frac{K\rho^2+2\rho+1}{KN}\right)}{\mathcal{N}\left(\frac{S+\rho S_S}{S}, \frac{C}{KMS^2}\right)} \tag{4.17}$$

Noting the result,

$$T_{HED2} = \begin{cases} \mathcal{N}\left(\frac{S}{S+\rho S_S}, \frac{t_{hed2}^2 NC + MS^2}{KMN(S+\rho S_S)^2}\right) & \mathcal{H}_0, \\ \mathcal{N}\left(\frac{S(\rho+1)}{S+\rho S_S}, \frac{t_{hed2}^2 NC + MS^2(K\rho^2+2\rho+1)}{KMN(S+\rho S_S)^2}\right) & \mathcal{H}_1 \end{cases} \tag{4.18}$$

Based on the statistics of T_{HED2} given by (4.18), we obtain the false alarm and detection probability in terms of Q-function as (4.13) and (4.14) respectively. □

4.3 Hybrid Roy's Largest Root Test

In a similar way as for ED, if we incorporate *offline noise estimation* and *online noise estimation* in RLRT, hybrid approaches of RLRT are developed and their performance parameters are derived in the following subsections.

4.3.1 Hybrid RLRT approach 1 (HRLRT1)

HRLRT1 is a similar approach as HED1, which deals with the study of detection performance of the RLRT algorithm using estimated noise variance. Noise variance is estimated from S auxiliary noise-only slots where we are sure that the primary signal is absent. Using the ML estimate of the noise variance (4.2), the decision statistic of HRLRT1 can be expressed as,

$$T_{HRLRT1} = \frac{\lambda_1}{\hat{\sigma}_{v_1}^2(S)} \tag{4.19}$$

Proposition 3. Under null hypothesis, after rigorous simplification, the test statistic in (4.19) could be approximated to the ratio of a Tracy-Widom random variable of order 2 (TW2) and a normal random variable. Hence, the false alarm probability P_{fa}^{HRLRT1} for number of sensors K, number of samples N, number of auxiliary slots S for noise estimation using (4.2) and threshold t_1 is given by,

$$P_{fa}^{HRLRT1} = 1 - F_0^{HRLRT1}(t_{hrlrt1}) \tag{4.20}$$

where $F_0^{HRLRT1}(t_{hrlrt1})$ is the CDF of the Probability Density Function shown below,

$$f_0^{HRLRT1}(t_{hrlrt1}) = C_1 \int_{-\infty}^{+\infty} |x| f_{TW2}\left(\frac{x t_{hrlrt1} - \mu}{\xi}\right) e^{\frac{-D(x-1)^2}{4}} \, \mathrm{d}x \tag{4.21}$$

with $f_{TW2}(\cdot)$ being the *pdf* of the TW2 and $C_1 = \frac{1}{2\xi}\sqrt{\frac{D}{\pi}}$.

Similarly, under alternate hypothesis, the test statistic in (4.19) approximates to normal random variable whose probability of detection P_d^{HRLRT1} under a similar scenario is given by,

$$P_d^{HRLRT1} = Q\left(\frac{t_{hrlrt1} - \mu_x}{\sqrt{\frac{2t_1^2}{D} + \sigma_x^2}}\right) \tag{4.22}$$

where, μ_x and σ_x^2 are mean and variance of a normal random variable, defined as

$$\mu_x = (1 + K\rho)\left(1 + \frac{K-1}{NK\rho}\right) \tag{4.23}$$

$$\sigma_x^2 = \frac{1}{N}(K\rho + 1)^2\left(1 - \frac{K-1}{NK^2\rho^2}\right) \tag{4.24}$$

Proof. Rearranging the test statistic of HRLRT1 for the sake of simplification as shown below,

$$T_{HRLRT1} = \frac{\lambda_1}{\sigma_v^2} \times \frac{1}{\hat{\sigma}_{v_1}^2(S)/\sigma_v^2} \tag{4.25}$$

Concerning null hypothesis where, $\boldsymbol{Y} = \boldsymbol{V}$ the sample covariance matrix $\boldsymbol{R}$ follows a Wishart distribution of degree N. Thus, for sufficiently large N and K, $\frac{\lambda_1|_{\mathcal{H}_0}}{\sigma_v^2}$ follows a Tracy Widom distribution of order 2 with suitably chosen centering and scaling parameters, recalling μ (2.65) and ξ (2.66) respectively.

Similarly, for sufficiently large values of $D = 2KSM$, $\hat{\sigma}_{v_1}^2(S)/\sigma_v^2$ follows a normal distribution $\mathcal{N}\left(1, \frac{2}{D}\right)$. Thus, the test statistic for HRLRT1 can be written as,

$$T_{HRLRT1}|_{\mathcal{H}_0} = \frac{1}{\mathcal{N}\left(1, \frac{2}{D}\right)} F_{TW2}\left(\frac{\frac{\lambda_1|_{\mathcal{H}_0}}{\sigma_v^2} - \mu}{\xi}\right) \tag{4.26}$$

The two random variables at the numerator and denominator of (4.26) are statistically independent, thus, the false alarm probability can be directly formulated as in (4.20).

Under alternate hypothesis, for $\rho > \rho_{Cric}$, where ρ_{Cric} is given by $\rho_{Cric} = \frac{1}{\sqrt{KN}}$, the distribution of $\frac{\lambda_1|_{\mathcal{H}_1}}{\sigma_v^2}$ was found to be asymptotically Gaussian [44, 68, 64] as shown below,

$$\frac{\lambda_1|_{\mathcal{H}_1}}{\sigma_v^2} \sim \mathcal{N}(\mu_x, \sigma_x^2) \tag{4.27}$$

where, μ_x and σ_x^2 are mean and variance of a normal random variable given by expressions (4.23) and (4.24) respectively.

Thus, the $T_{HRLRT1}|_{\mathcal{H}_1}$ in (4.25) can be written as,

$$T_{HRLRT1}|_{\mathcal{H}_1} = \frac{\mathcal{N}(\mu_x, \sigma_x^2)}{\mathcal{N}\left(1, \frac{2}{D}\right)} \cong \mathcal{N}\left(\mu_x, \frac{2t_{hrlrt1}^2}{D} + \sigma_x^2\right) \tag{4.28}$$

With above Normal Approximation, the Probability of Detection for HRLRT1 could be easily noted as (4.22). □

4.3.2 Hybrid RLRT approach 2 (HRLRT2)

HRLRT2 is an alternate hybrid approach of RLRT where noise variance given by (4.3) is estimated from the previously received signal slots declared as $\mathcal{H}_0$ by the algorithm. The decision statistic of HRLRT2 can be written as,

$$T_{HRLRT2} = \frac{\lambda_1}{\hat{\sigma}_{v_2}^2(S)} \tag{4.29}$$

Proposition 4. Under null hypothesis, after rigorous simplification, the test statistic in (4.29) could be approximated to the ratio of a TW2 random variable and a normal random variable. Hence, the false alarm probability P_{fa}^{HRLRT2} for number of sensors K, number of samples N, number of auxiliary slots S for noise estimation using (4.3) and threshold t_2 is given by,

$$P_{fa}^{HRLRT2} = 1 - F_0^{HRLRT2}(t_{hrlrt2}) \tag{4.30}$$

where, $F_0^{HRLRT2}(t_{hrlrt2})$ is the CDF of the Probability Density Function shown below,

$$f_0^{HRLRT2}(t_{hrlrt2}) = C_2 \int_{-\infty}^{+\infty} |x| f_{TW2}\left(\frac{x t_{hrlrt2} - \mu}{\xi}\right) e^{\frac{-(x-\mu_1)^2}{2\sigma_1^2}} \mathrm{d}x \tag{4.31}$$

with $C_2 = \frac{1}{\xi\sigma_1^2\sqrt{2\pi}}$.

Similarly, under alternate hypothesis, the test statistic in (4.29) approximates to normal random variable whose probability of detection P_d^{HRLRT2} under a similar scenario is given by,

$$P_d^{HRLRT2} = Q\left(\frac{t_2 - \mu_x/\mu_1}{\sqrt{\frac{t_{hrlrt2}^2\sigma_1^2+\sigma_x^2}{\mu_1^2}}}\right) \tag{4.32}$$

where, μ_x (4.23), μ_1 (4.33) and σ_x^2 (4.24), σ_1^2 (4.34) are mean and variance parameters with,

$$\mu_1 = \frac{S + S_S}{S} \tag{4.33}$$

$$\sigma_1^2 = \frac{S + 2\rho S_S + \rho^2 K S_S}{KMS^2} \tag{4.34}$$

Proof. Rearranging the test static of HRLRT2 (4.29) for the sake of simplification as shown below,

$$T_{HRLRT2} = \frac{\lambda_1}{\sigma_v^2} \times \frac{1}{\hat{\sigma}_{v_2}^2(S)/\sigma_v^2} \tag{4.35}$$

Concerning null hypothesis, $\frac{\lambda_1|\mathcal{H}_0}{\sigma_v^2}$ for sufficiently large N and K follows a TW2 distribution with centering and scaling parameters given by (2.65) and (2.66) respectively.

Similarly, for sufficiently large values of $2KS_SM$, $\hat{\sigma}_{v_2}^2(S)/\sigma_v^2$ follows a Normal Distribution $\mathcal{N}(\mu_1, \sigma_1^2)$ with μ_1 and σ_1^2 given by (4.33) and (4.34) respectively. Thus, the test statistics for HRLRT2 under null hypothesis could be written as,

$$T_{HRLRT2}|_{\mathcal{H}_0} = \frac{1}{\mathcal{N}(\mu_1, \sigma_1^2)} F_{TW2}\left(\frac{\frac{\lambda_1|\mathcal{H}_0}{\sigma_v^2} - \mu}{\xi}\right) \tag{4.36}$$

The two random variables at the numerator and denominator of (4.36) are statistically independent. Thus, the false alarm probability for HRLRT2 can directly be formulated as (4.30).

Similarly, under alternate hypothesis, for $\rho > \rho_{Cric}$ the distribution of $\frac{\lambda_1|\mathcal{H}_1}{\sigma_v^2}$ is asymptotically Gaussian as shown below,

$$\frac{\lambda_1|\mathcal{H}_1}{\sigma_v^2} \sim \mathcal{N}(\mu_x, \sigma_x^2) \tag{4.37}$$

with parameters μ_x (4.23) and σ_x^2 (4.24). Thus, $T_{HRLRT2}|_{\mathcal{H}_1}$ in (4.35) can be written as,

$$T_{HRLRT2}|_{\mathcal{H}_1} = \frac{\mathcal{N}(\mu_x, \sigma_x^2)}{\mathcal{N}(\mu_1, \sigma_1^2)} \cong \mathcal{N}\left(\frac{\mu_x}{\mu_1}, \frac{t_{hrlrt2}^2 \sigma_1^2 + \sigma_x^2}{\mu_1^2}\right) \tag{4.38}$$

With the above normal approximation, the detection probability for HRLRT2 can be noted as (4.32). □

4.4 Results

This section shows the simulation of the ROC curves and performance curves of hybrid approaches of ED and RLRT spectrum sensing algorithms. The accuracy of the closed-form expressions is confirmed by the results presented in Fig. 4.3 and 4.4, respectively, where the theoretical formulas are compared against the simulated detection performance over S auxiliary noise-only slots (S ranges from 1 to 8). Perfect match of the theoretical and the numerical curve validates the considered model. As it can be noticed, with the increase in the number of auxiliary slots used for the estimation of the noise variance, the probability of detection increases for both hybrid approaches.

Fig. 4.5 illustrates the comparison of ED, HED1 and HED2 performance as a function of the SNR. Performance of HED1 and HED2 varies typically around 0 dB SNR but no visible difference can be noted in extreme high or low SNR values. Since there is a chance of misinterpretation of noise plus primary signal as only-noise samples (used to estimate the noise variance) by ED in case of HED2, performance of HED2 is slightly lower than HED1 near 0 dB of SNR. By increasing the number of slots used for the estimation of the noise variance, the gap between HED1 and HED2 decreases and both approaches approximate the known-variance ED curve.

The convergence of the hybrid approach of RLRT to an ideal RLRT (known variance) is illustrated in Fig. 4.6. By increasing the number of auxiliary slots used for the estimation of noise variance, the performance of HRLRT1 and HRLRT2 converge at the ideal RLRT performance.

The performance of HED1 and HRLRT1 is compared in Fig. 4.7. The noise variance is estimated using (4.2) from S auxiliary sure noise-only slots. The curves approach the ideal ED and RLRT curves by increasing the number of auxiliary slots S, but the rate of convergence of HED1 is slower.

The effect of the noise variance estimation uncertainty on ED and RLRT algorithms is considered in Fig. 4.8. Assuming the Gaussian distribution of the noise variance estimate with mean equal to nominal value, the ROC for ED and RLRT is plotted, setting $\mathrm{Var}(\hat{\sigma}_v^2) = 0.0032$ (-25 dB). The result shows that, for the same

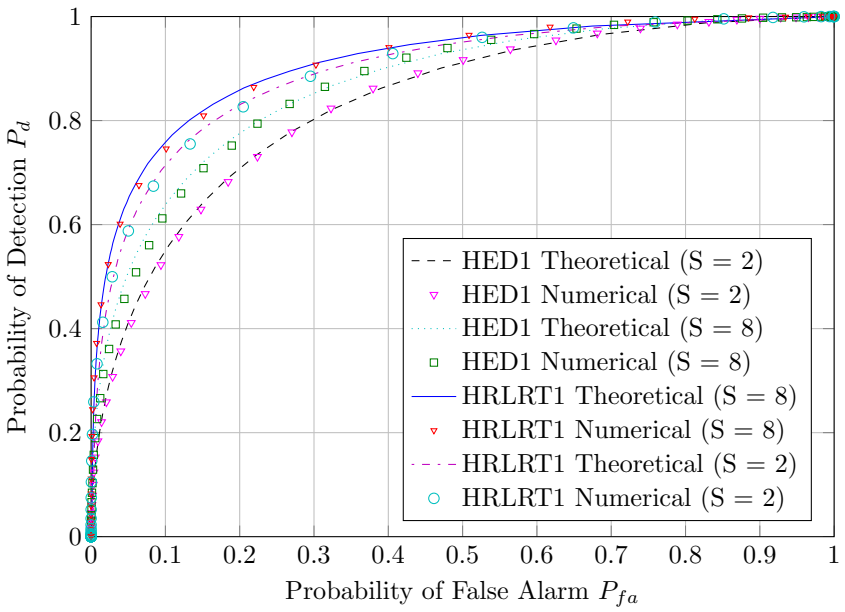


Figure 4.3: Theoretical and numerical ROC plot of HED1/HRLRT1, $N = 80$, $M = 80$, $K = 4$, SNR $= -10$ dB.

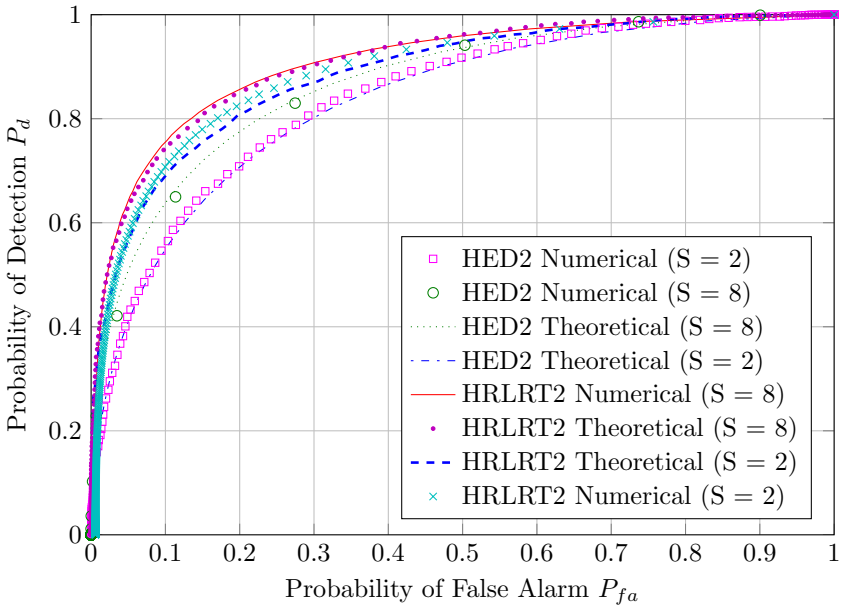


Figure 4.4: Theoretical and numerical ROC plot of HED2/HRLRT2, $N = 80$, $M = 80$, $K = 4$, SNR $= -10$ dB.

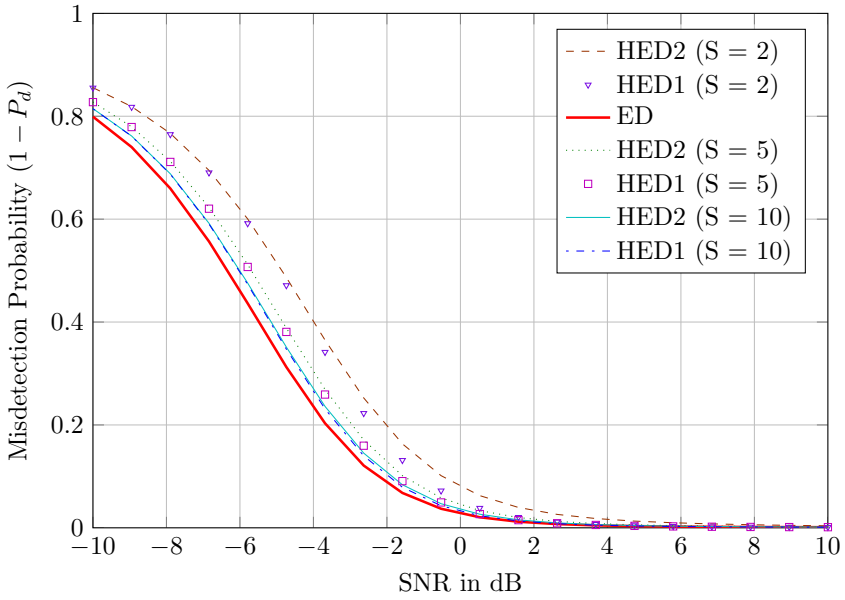


Figure 4.5: Performance curves of ED and its hybrid approaches, $N = 10$, $M = 10$, $K = 5$, $P_{fa} = 0.05$.

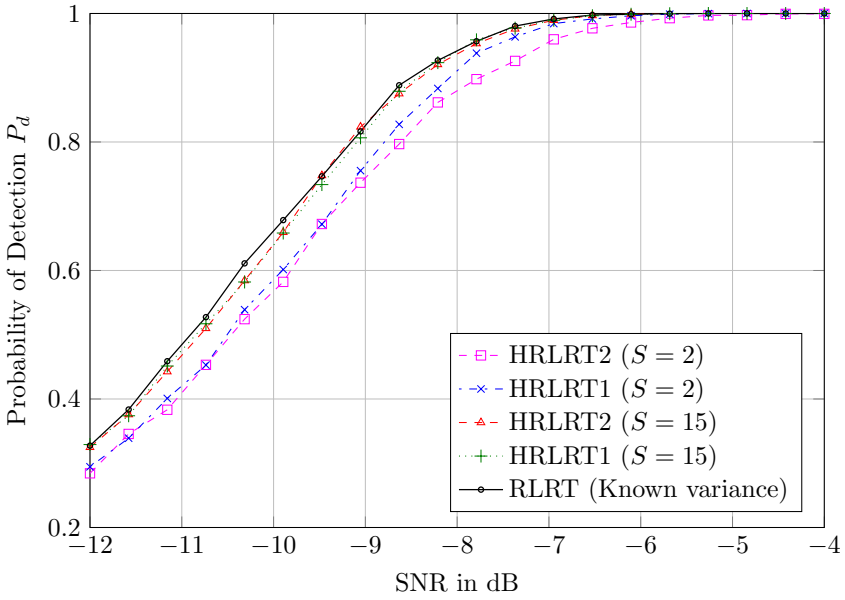


Figure 4.6: Performance curves of RLRT and its hybrid approaches, $N = 80$, $M = 80$, $K = 4$, $P_{fa} = 0.05$.

uncertainty of the noise variance estimate, the performance gap between the ideal curve and the curve with wrong variance is larger for ED as compared to RLRT.

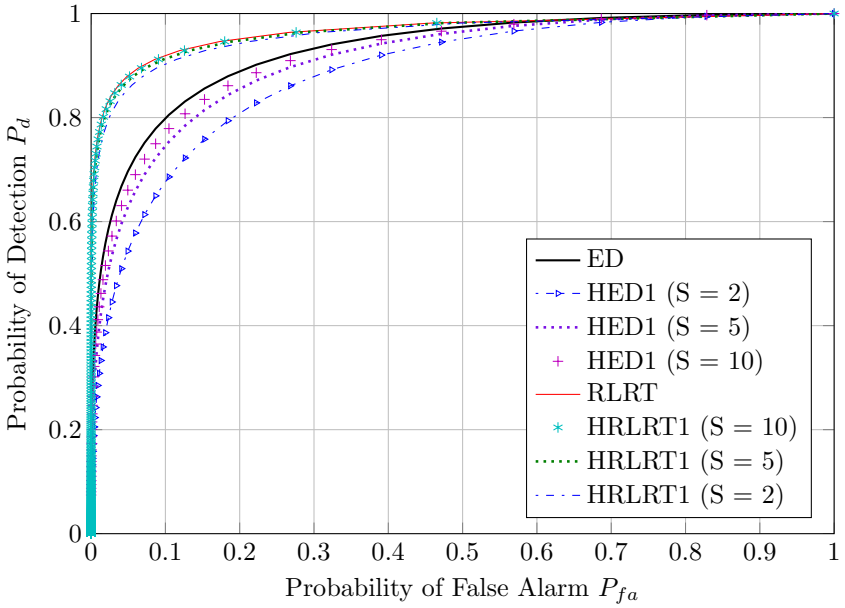


Figure 4.7: ROC curves of ED/RLRT and their hybrid approach 1 (HED1/HRLRT1), $N = 80$, $M = 80$, $K = 4$, SNR $= -10$ dB.

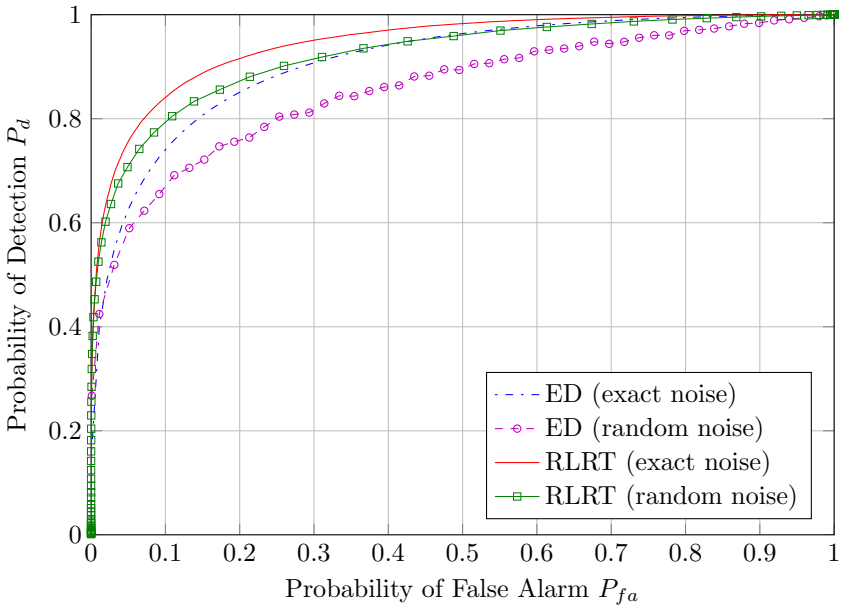


Figure 4.8: Effect of noise variance fluctuation on ED and RLRT, $N = 100$, $K = 4$, $\mathrm{Var}(\hat{\sigma}_v^2) = 0.0032$ (-25 dB) given nominal noise variance $\sigma_v^2 = 1$.

Thus, it can be easily noticed that RLRT is more robust to noise variance uncertainty as compared to ED algorithm.

4.5 Optimization of Hybrid Energy Detection

In this section, we show some preliminary results on the optimization of Hybrid Energy Detection. Let us assume that the secondary user has a fixed time window for both noise estimation and detection, i.e., the number of samples that the SU can use for both noise estimation and signal detection is constant. For the sake of simplicity, the Maximum Likelihood expression of (4.1) will be considered for the optimization of HED. Considering K antennas, M samples are used for estimation and N samples for detection. Our fixed time constraint implies $M + N = c$ where c is a constant, hence our goal is to find the optimal M (and consequently optimal N) that gives the maximum detection probability.

The false alarm and detection probability expressions are the starting point of our optimization task. The false alarm probability $P_{fa}^{(HED)}$ for number of sensors K, number of samples N, number of noise estimation samples M and threshold t is given by,

$$P_{fa}^{(HED)} = Q\left[\frac{t-1}{\sqrt{\frac{M+Nt^2}{KMN}}}\right] \tag{4.39}$$

Similarly, the probability of detection $P_d^{(HED)}$ in similar scenario is given by,

$$P_d^{(HED)} = Q\left[\frac{(t-1-\rho)}{\sqrt{\frac{t^2}{KM}+\frac{K\rho^2+2\rho+1}{KN}}}\right] \tag{4.40}$$

where ρ is the signal-to-noise ratio.

First of all, from (4.39) we find the threshold t expression as a function of the P_{fa},

$$t = \frac{M\left(K + \epsilon\sqrt{\frac{KM+KN-\epsilon^2}{MN}}\right)}{KM-\epsilon^2} \tag{4.41}$$

where $\epsilon = Q^{-1}[P_{fa}]$. This is the only acceptable solution ($t > 1$) of a second degree equation. Unless KM is smaller than ϵ^2 (which is of no interest), this is always true.

By substituting (4.41) in (4.40) we obtain the following expression:

$$P_d = Q\left[\frac{\frac{M\left(K+\epsilon\sqrt{\frac{KM+KN-\epsilon^2}{MN}}\right)}{KM-\epsilon^2} - 1 - \rho}{\sqrt{\frac{M\left(K+\epsilon\sqrt{\frac{KM+KN-\epsilon^2}{MN}}\right)^2}{K(KM-\epsilon^2)^2} + \frac{K\rho^2+2\rho+1}{KN}}}\right] \tag{4.42}$$

Let us now use the following substitutions:

$$M = x \tag{4.43}$$
$$N = c - x \tag{4.44}$$

where $x \in \mathbb{N}$ and $c = M + N$.

We first rewrite the threshold expression in (4.41):

$$t = \frac{x(K + \epsilon\sqrt{\frac{Kc-\epsilon^2}{xc-x^2}})}{Kx - \epsilon^2} \tag{4.45}$$

Then, we rewrite the argument of the Q-function of (4.42):

$$f(x) = \frac{\frac{x\left(K+\epsilon\sqrt{\frac{Kc-\epsilon^2}{cx-x^2}}\right)}{Kx-\epsilon^2} - 1 - \rho}{\sqrt{\frac{x\left[K^2+\frac{\epsilon^2(Kc-\epsilon^2)}{cx-x^2}+2K\epsilon\sqrt{\frac{Kc-\epsilon^2}{cx-x^2}}\right]}{K(K^2x^2+\epsilon^4-2\epsilon^2Kx)} + \frac{K\rho^2+2\rho+1}{Kc-Kx}}} \tag{4.46}$$

Fig. 4.9 shows the probability of detection of HED as a function of M for different values of $M + N$, with 4 antennas, SNR equal to -10dB and P_{fa} equal to 10^{-2}. It is clear to see that, when c samples are available for both estimation and detection, the highest probability of detection occurs for:

$$M \approx N \approx \frac{M+N}{2} \tag{4.47}$$

Hence, given a time slot for spectrum sensing, the best performance occurs when estimation and detection slots are equally split.

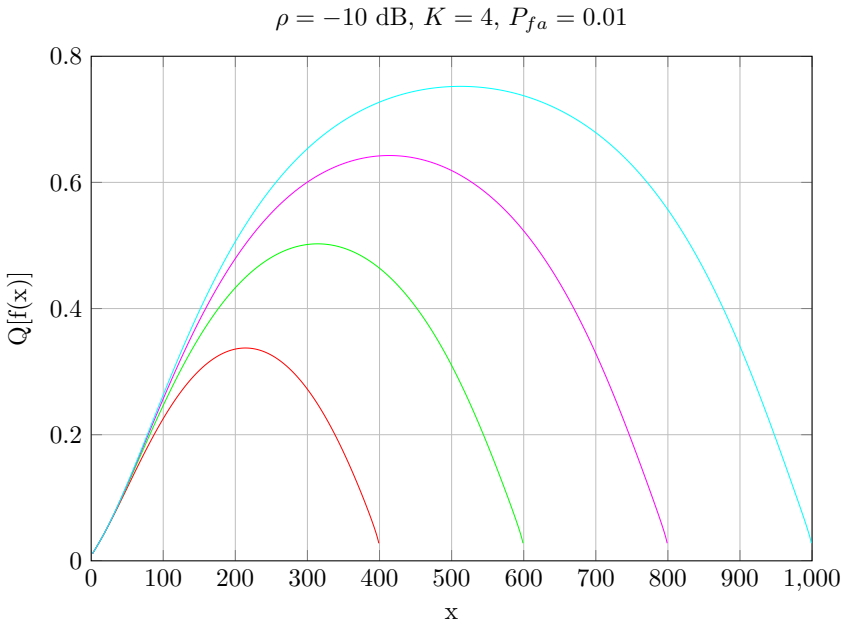


Figure 4.9: Probability of detection as a function of M given $M + N = \text{const}$.

Chapter 5

EigenVEctor (EVE) Test and hybrid variant

We introduce in this chapter a test statistic belonging to the class of EBD algorithms which was first proposed in [81]. This algorithm, named EigenVEctor Test (EVE), uses the eigenvector associated to the largest eigenvalue of the covariance matrix to improve the performance of the existing EBD algorithms. A hybrid approach of the EVE test is also introduced and a final perfomance comparison among most of test statistics presented so far is assessed.

5.1 EigenVEctor (EVE) Test

The starting idea of the new test is that given a $\mathcal{H}_1$ slot, the eigenvector $\boldsymbol{e}_1$ associated to largest eigenvalue λ_1 provides an estimation of the channel vector $\boldsymbol{h}$.

Given S_{aux} signal slots available before the current sensing slot, we can construct a matrix of size $K \times (S_{aux} \cdot N)$ from all the samples and evaluate the eigenvector $\boldsymbol{e}_{aux}$ corresponding to largest eigenvalue. The proposed statistical test known as EVE test [81], which exploits the channel estimation parameter $\boldsymbol{e}_{aux}$ in its test statistic, is defined as,

$$T_{EVE} = \frac{S_{aux}\left[\boldsymbol{e}_{aux}^H \boldsymbol{R}\, \boldsymbol{e}_{aux}\right] + \left[\boldsymbol{e}^H \boldsymbol{R}\, \boldsymbol{e}\right]}{\sigma_v^2 (S_{aux}+1)} \tag{5.1}$$

Note that if $S_{aux} = 0$, the test reduces to

$$T_{EVE} = \frac{\boldsymbol{e}^H \boldsymbol{R}\, \boldsymbol{e}}{\sigma_v^2} = \frac{\|\boldsymbol{e}\|^2 \lambda_1}{\sigma_v^2} = \frac{\lambda_1}{\sigma_v^2} \tag{5.2}$$

and has the same statistical power of the RLRT.

5.2 Hybrid and blind variant of EVE

As we did in the previous chapter for Energy Detection (ED) and Roy's Largest Root Test (RLRT), noise variance can be estimated from S auxiliary noise-only slots in which we are sure that the primary signal is absent.

Let us consider in this case for the sake of simplicity the offline noise estimation method, in which the noise variance is estimated in the following way:

$$\hat{\sigma}_v^2(S) = \frac{1}{KSM} \sum_{s=1}^{S} \sum_{k=1}^{K} \sum_{m=1}^{M} |v_{km}|^2 \tag{5.3}$$

This corresponds to the first hybrid approach (see Sec. 4.1.1), and we apply it to the new EVE test, by defining a new Hybrid EigenVEctor (HEVE) test:

$$T_{HEVE} = \frac{S_{aux} \left[\boldsymbol{e}_{aux}^H \boldsymbol{R} \boldsymbol{e}_{aux} \right] + \left[\boldsymbol{e}^H \boldsymbol{R} \boldsymbol{e} \right]}{\hat{\sigma}_{HEVE}^2(S) \cdot (S_{aux} + 1)} \tag{5.4}$$

where $\hat{\sigma}_{HEVE}^2(S)$ is computed as in (5.3). Note that in HEVE we use S_{aux} slots declared as $\mathcal{H}_1$ (signal-only) to compute the eigenvector $\boldsymbol{e}_{aux}$ for channel estimation and S slots declared as $\mathcal{H}_0$ (noise-only) to estimate the noise variance $\hat{\sigma}_{HEVE}^2(S)$. Similarly to (5.2) if $S_{aux} = 0$, the test reduces to $\lambda_1 / \hat{\sigma}_{HEVE}^2(S)$, which has the same statistical power of Hybrid RLRT.

Moreover, the EVE test can be further modified to cover the blind case scenario. So, we can the define the Blind EigenVEctor (BEVE) test as:

$$T_{BEVE} = \frac{S_{aux} \left[\boldsymbol{e}_{aux}^H \boldsymbol{R} \boldsymbol{e}_{aux} \right] + \left[\boldsymbol{e}^H \boldsymbol{R} \boldsymbol{e} \right]}{\left(\frac{1}{K} \sum_{i=1}^{K} \lambda_i \right) (S_{aux} + 1)}. \tag{5.5}$$

By following the same procedure for EVE and HEVE, this test is equivalent to GLRT for $S_{aux} = 0$.

5.2.1 Neyman-Pearson Test

The Neyman Pearson (NP) lemma is known to provide the Uniformly Most Powerful (UMP) test, achieving the maximum possible P_d for any given value of P_{fa}. In our setting this is the case when both the noise level σ_v^2 and the channel vector $\boldsymbol{h}$ are a priori known. The NP test is given by the following likelihood ratio:

$$T_{NP} = \frac{p_1(\boldsymbol{Y}; \boldsymbol{h}, \sigma_s^2, \sigma_v^2)}{p_1(\boldsymbol{Y}; \sigma_v^2)} \tag{5.6}$$

and is known to be optimal, i.e., to achieve the maximum possible P_d for any given value of P_{fa}.

As an example, under the considered model, if the signal samples are independent Gaussian samples, the NP test is obtained by using:

$$\boldsymbol{p}_0(\boldsymbol{Y};\sigma_v^2) = \frac{1}{(\pi\sigma_v^2)^{NK}} \exp\left(-\frac{N\boldsymbol{tr}\boldsymbol{R}}{\sigma_v^2}\right) \tag{5.7}$$

and

$$p_1(\boldsymbol{Y};\boldsymbol{h},\sigma_s^2,\sigma_v^2) = \frac{1}{(\pi^K det\boldsymbol{\Sigma})^N} \exp\left(-\boldsymbol{R}\boldsymbol{\Sigma}^{-1}\right) \tag{5.8}$$

where, $\boldsymbol{\Sigma} = \sigma_v^2\boldsymbol{I}_k + \sigma_s^2\boldsymbol{h}\boldsymbol{h}^H$

The NP test provides the best possible performance, but requires exact knowledge of both $\boldsymbol{h}$ and σ_v^2. Hence, the NP test can be used as a benchmark to evaluate

5.3 Simulation results

First, let us list all the test statistics that have been compared in this scenario. We have already mentioned:

- Neyman Pearson (NP) Test
- EVE Test
- Hybrid EVE (HEVE)
- Blind EVE (BEVE)

and we have also taken into account:

- Energy Detection (ED)
- Hybrid ED (HED)
- Roy's Largest Root Test (RLRT)
- Hybrid RLRT (HRLRT)
- Generalized Likelihood Ratio Test (GLRT).

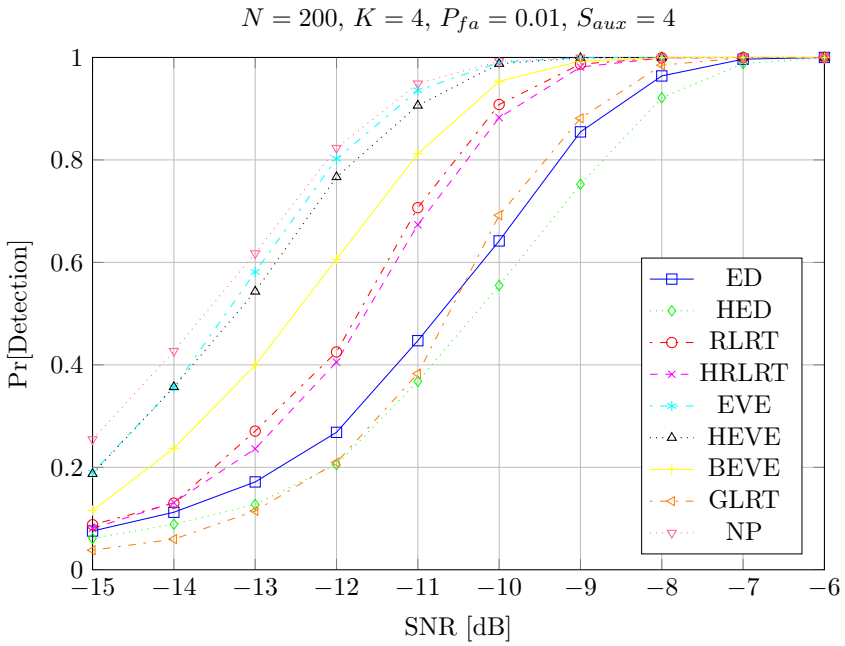


Figure 5.1: Performance curve, Pd vs. SNR, $K = 4$, $N = 200$.

Please note that all considered hybrid approaches with noise estimation (HEVE, HED, HRLRT) use (5.3) to estimate the noise power, hence HED and HRLRT are in fact HED1 and HRLRT1 of the previous chapter. Results are shown in terms of Receiver Operating Characteristic (ROC) curves (P_d vs. P_{fa}) and performance curves, in which P_d is plotted against SNR, by fixing P_{fa}. The primary signal has a Gaussian distribution and the typical Rayleigh flat fading channel scenario has been considered. In performance curves, P_{fa} is fixed to 10^{-2} while in ROC curves, SNR = -12 dB.

Fig. 5.1 and 5.2 show respectively the performance and ROC curves of all the test statistics with 4 antennas, 200 samples per slot and 4 auxiliary slots for both channel and noise estimation. It can be noticed that EVE and HEVE are clearly capable to significantly reduce the gap with NP wrt RLRT. The gap between EVE and RLRT is 1 dB at $P_d = 0.9$. In general, the hybrid approaches HEVE, HRLRT and HED are very close in performance with their respective known-noise tests, while the blind tests BEVE and GLRT show a clear gap with EVE and RLRT.

In Fig. 5.3 and 5.4 $N = 200$ and $S_{aux} = 4$ as the previous case, but 8 antennas have been used. It can be noticed that both BEVE and GLRT improve their performance (GLRT outperforms ED) and the gap between EVE and RLRT increases (2 dB at $P_d = 0.9$).

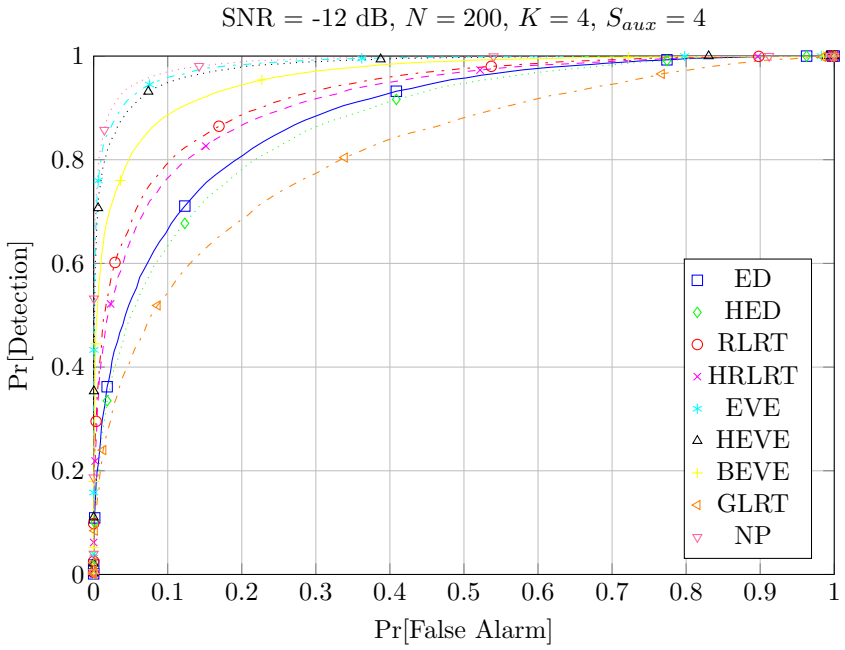


Figure 5.2: ROC curve, $K = 4$, $N = 200$.

By reducing the number of antennas and samples (but same number of auxiliary slots), we obtain the results shown in Fig. 5.5 and 5.6 with 4 antennas and 100 samples. The performance of all tests is lower but especially blind tests suffer from this reduction and GLRT becomes by far the worst performing test.

We focus now our attention on the hybrid tests HEVE, HRLRT and HED. Fig. 5.7 and 5.8 show how the number of slots affects the performance of these tests. The number of antennas is equal to 4, while we used 200 samples per slot. It is evident that there is an important gap between 2 and 4 auxiliary slots (especially for HEVE), while the curves with 4 and 6 auxiliary slots are almost overlapped.

Finally, we show 2 other performance curves. In Fig. 5.9 the detection probability is plotted against the number of antennas, with 100 samples per slot and 6 auxiliary slots, while in Fig. 5.10 P_d is plotted against the number of samples, with 4 antennas and 6 auxiliary slots. NP and the EVE group tests require a much smaller number of antennas or sensors to reach $P_d \simeq 1$ wrt to all other tests. It is again very clear that BEVE and GLRT have the best improvement in performance as the number of antennas increases.

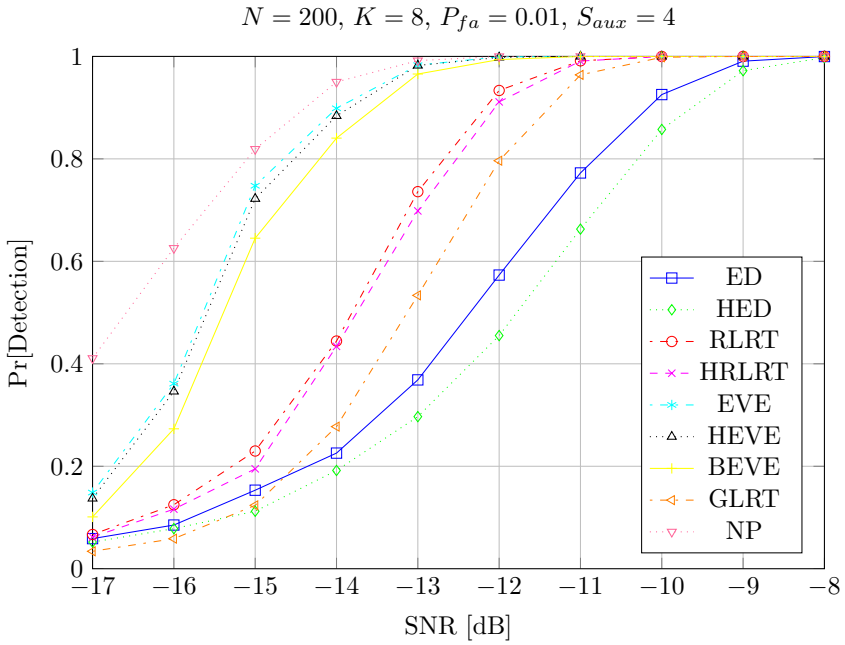


Figure 5.3: Performance curve, Pd vs. SNR, $K = 8$, $N = 200$.

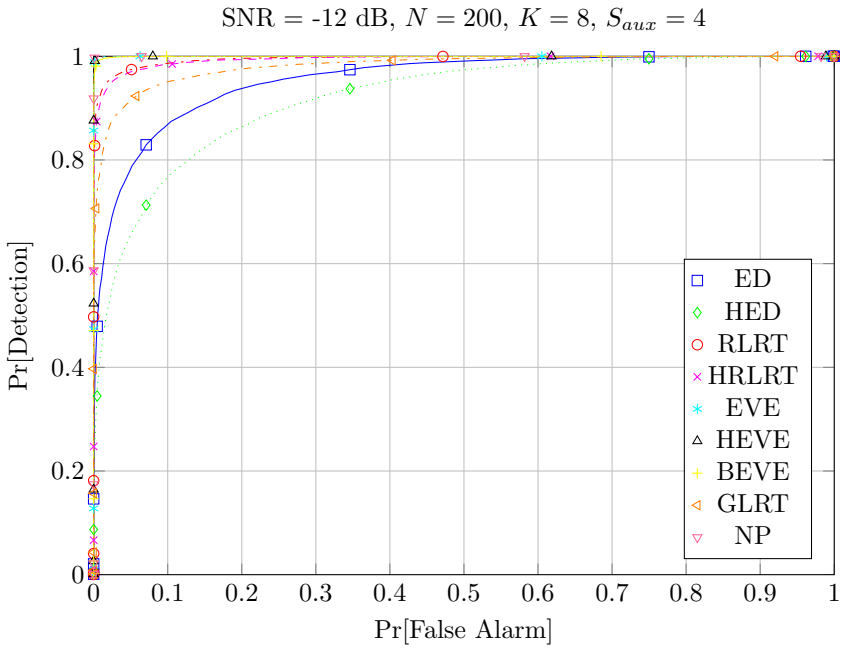


Figure 5.4: ROC curve, $K = 8$, $N = 200$.

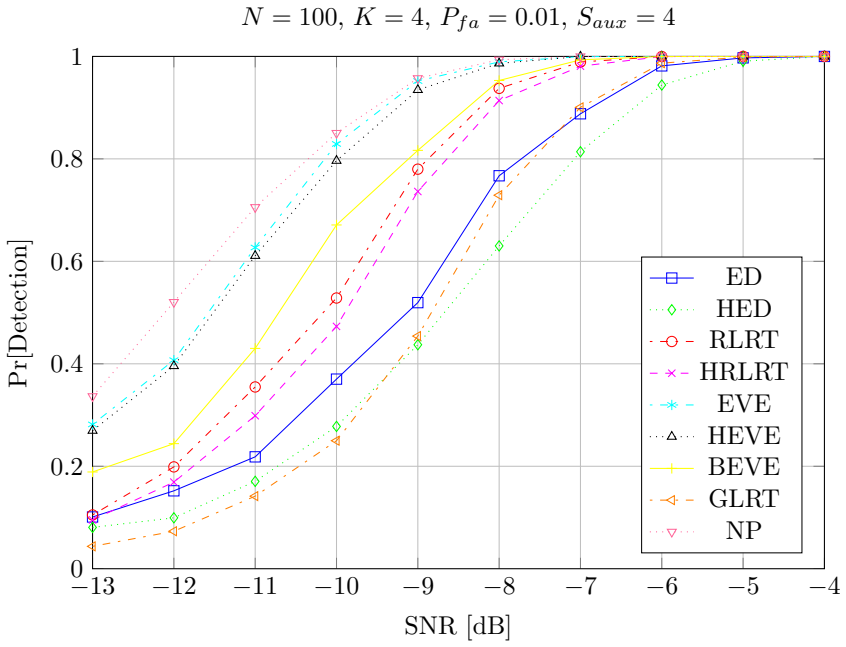


Figure 5.5: Performance curve, Pd vs. SNR, $K = 4$, $N = 100$.

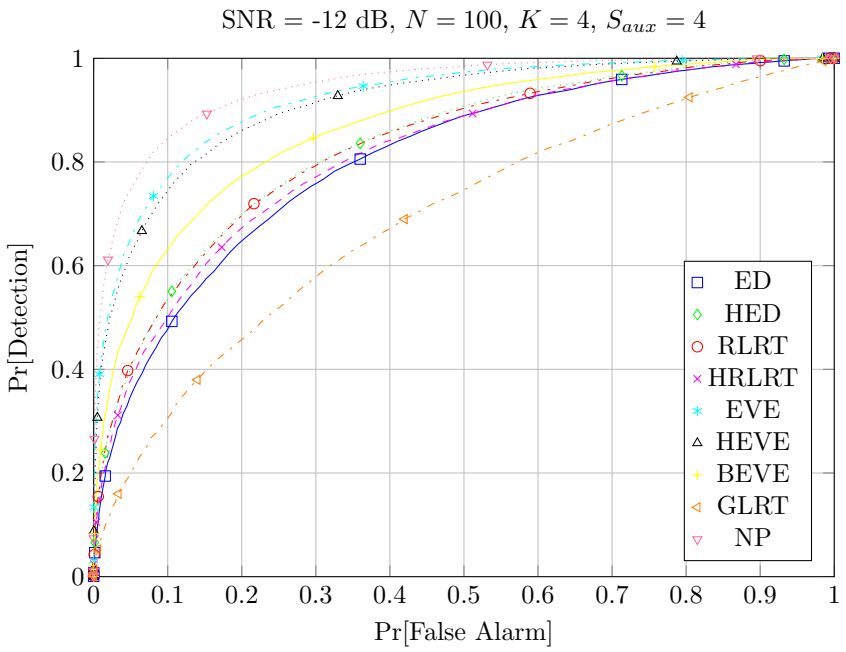


Figure 5.6: ROC curve, $K = 4$, $N = 100$.

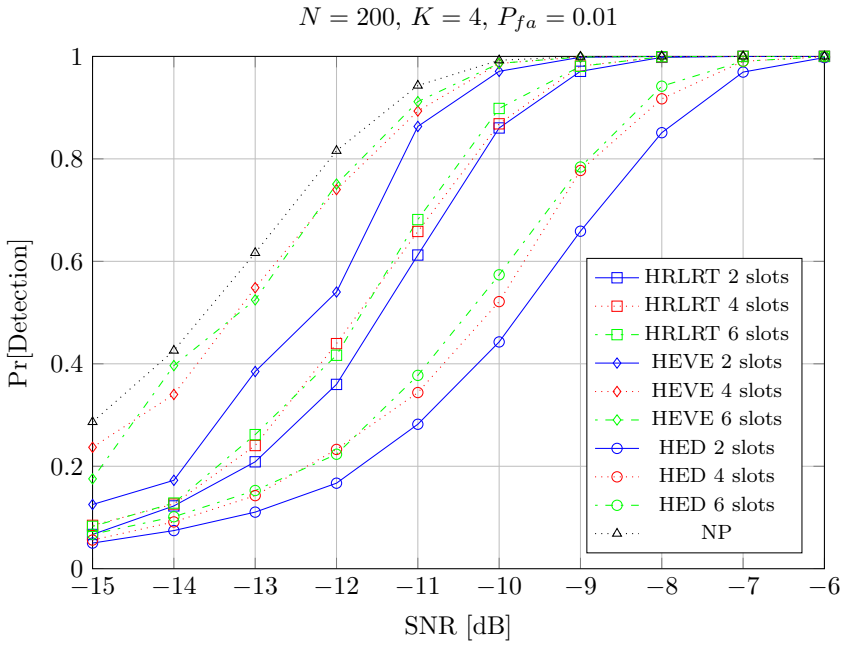


Figure 5.7: Performance curve, Pd vs. SNR, with 2, 4, 6 auxiliary slots.

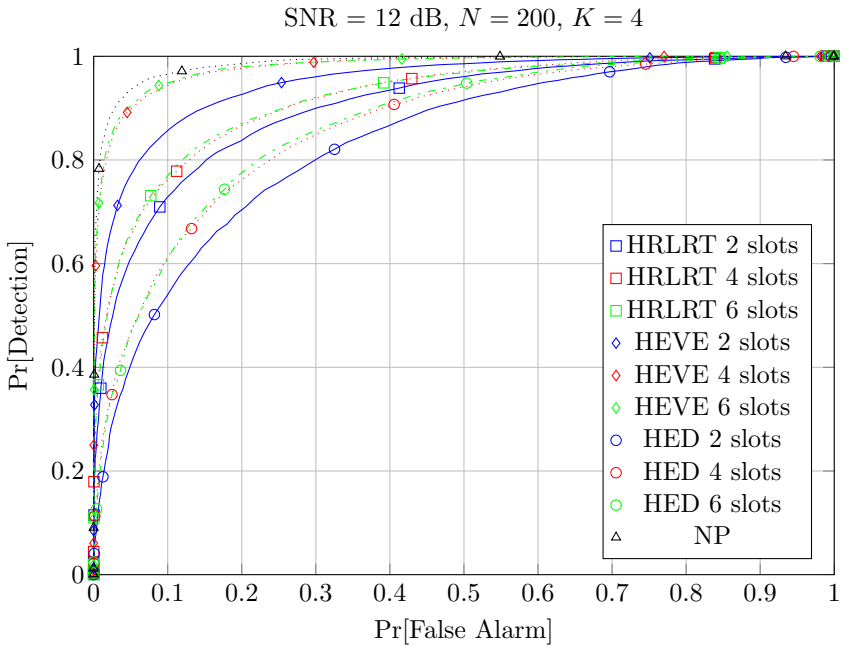


Figure 5.8: ROC curve, with 2, 4, 6 auxiliary slots.

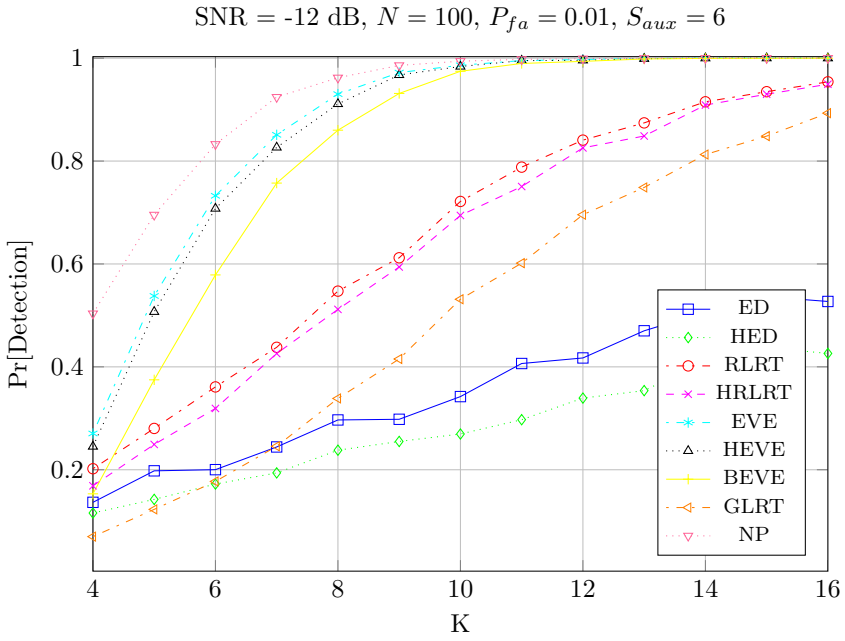


Figure 5.9: Performance curve, Pd vs. K, $N = 100$.

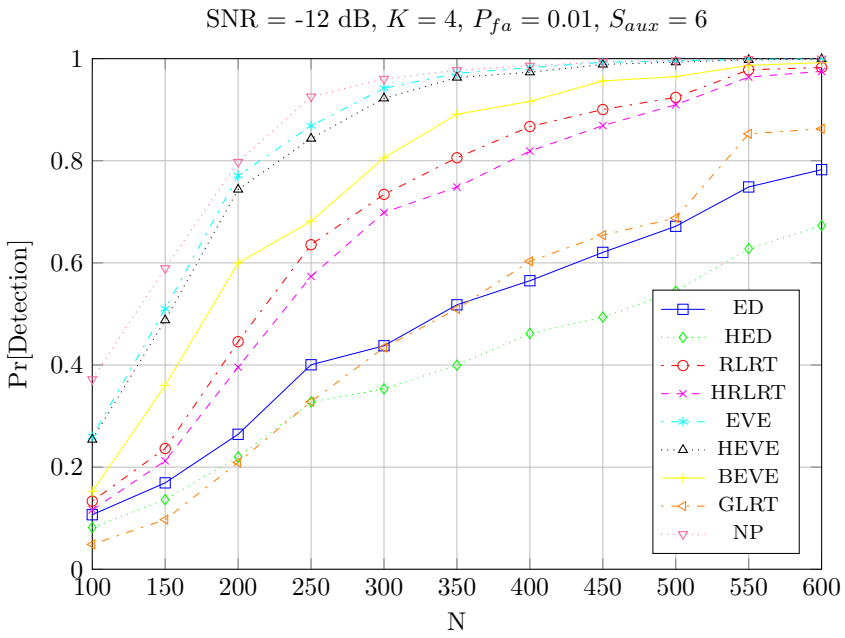


Figure 5.10: Performance curve, Pd vs. N, $K = 4$.

Chapter 6

SNR Wall for multi-antenna Energy Detection

For CR application the most popular sensing algorithm is the simple ED that compares the energy of the received signal against the noise variance σ_v^2. ED requires the perfect knowledge of the noise power at the receiver [10, 22, 23, 24], however, in real systems the detector does not have a prior knowledge of the noise level. In recent years, variation and unpredictability of the precise noise level at the sensing device has become a critical issue, which is also known as *noise uncertainty*.

It is known from [25] that there is a certain SNR threshold in case of noise uncertainty known as *SNR Wall*, which prevents ED from achieving the desired performance even if the detection interval tends to infinity.

This chapter, starting from works [25, 28], extends the condition of SNR Wall [25] for the multi-sensor ED with auxiliary noise variance estimation (offline method) described in Chap. 4. For auxiliary noise variance estimation of White Gaussian Noise samples, the distribution of the estimated variance is studied and linked to the uncertainty bound referred to [25]. The SNR Wall expression is derived for multi-sensor ED and proved to be independent of the number of sensors. It is concluded that the noise uncertainty can be reduced by increasing the number of samples used for noise variance estimation, but the number of samples/slots used for noise estimation exponentially increases as the SNR Wall condition becomes more stringent.

6.1 Multi-antenna ED and SNR Wall

With reference to the system model of Sec. 2.2.1 we briefly summarize the main results of Energy Detection (ED) presented in Sec. 2.3.1. ED test statistic is defined

as

$$T_{ED} = \frac{1}{KN\sigma_v^2} \sum_{k=1}^{K} \sum_{n=1}^{N} |y_k(n)|^2 \tag{6.1}$$

the false alarm probability (P_{fa}) is defined as,

$$P_{fa} = \Pr(T > t|_{\mathcal{H}_0}) = Q\left[(t-1)\sqrt{KN}\right] \tag{6.2}$$

and the detection probability (P_d) as

$$P_d = Pr(T > t|_{\mathcal{H}_1}) = Q\left[\frac{(t-1-\rho)\sqrt{KN}}{\sqrt{K\rho^2 + 2\rho + 1}}\right] \tag{6.3}$$

where $Q(.)$ is the standard normal complementary CDF.

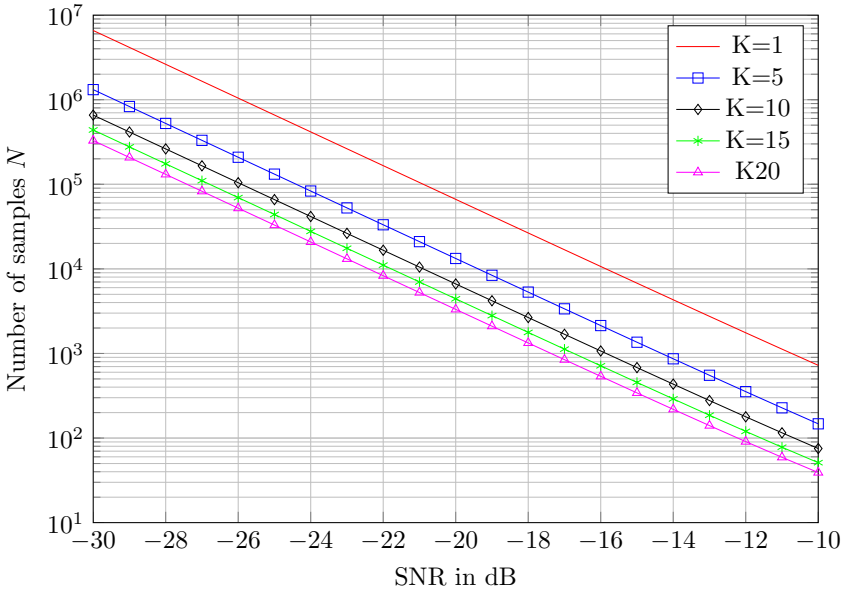


Figure 6.1: Comparison of sample complexity N for a given P_d and P_{fa} in a single and multi-antenna scenario as a function of the SNR, $P_d = 0.9$, $P_{fa} = 0.1$.

If we eliminate the threshold t by using a similar approach as in [25] from (6.2) and (6.3) to calculate the number of samples in a sensing slot to achieve the required P_d and P_{fa}, we get,

$$N = \frac{\left[\sqrt{(K\rho^2 + 2\rho + 1)}Q^{-1}(P_d) - Q^{-1}(P_{fa})\right]^2}{K\rho^2} \tag{6.4}$$

The expression of N in (6.4) exactly matches the expression of N derived in [25] when $K = 1$ (single antenna) and with real signals. Fig. 6.1 plots the sample complexity N for different SNR in single and multi antenna scenario. It is clear that, given prior information of the noise variance, the received signal can theoretically be detected for any SNR value by adjusting the detection interval accordingly.

Now, let us consider the case in which the noise variance is not precisely known but its deviation is known to be bounded in the interval [25] $\left[\frac{1}{\beta}\sigma_v^2, \beta\sigma_v^2\right]$, i.e.,

$$\hat{\sigma}_v^2 \in \left[\frac{1}{\beta}\sigma_v^2, \beta\sigma_v^2\right] \tag{6.5}$$

where σ_v^2 is the nominal (true) noise power and $\beta > 1$ is the parameter that quantifies the level of uncertainty. It is evident that the knowledge of the noise variance is imperative for the optimum performance of ED. In real practice it is not possible to know the exact value of the noise variance, so the only option is to estimate it from the noise samples. Unfortunately, the variation and the unpredictability of the noise variance from the biased estimate of the true noise variance is unavoidable. If we denote the estimate of the noise variance with $\hat{\sigma}_v^2$ and consider the ED detection statistic as,

$$\tilde{T}_{ED} = \frac{1}{KN}\sum_{k=1}^{K}\sum_{n=1}^{N}|y_k(n)|^2, \tag{6.6}$$

the Gaussian approximation of the detection statistic in (6.6) can be written as,

$$\tilde{T}_{ED} = \begin{cases} N_{\mathbb{R}}\left(\hat{\sigma}_v^2, \frac{\hat{\sigma}_v^4}{KN}\right) & \mathcal{H}_0, \\ N_{\mathbb{R}}\left(\|\boldsymbol{h}\|\sigma_s^2 + \hat{\sigma}_v^2, \frac{K\|\boldsymbol{h}\|^2\sigma_s^4 + 2\|\boldsymbol{h}\|\sigma_s^2\hat{\sigma}_v^2 + \hat{\sigma}_v^4}{KN}\right) & \mathcal{H}_1, \end{cases} \tag{6.7}$$

whose performance parameters can be written as,

$$P_{fa} = Q\left[\frac{(t - \hat{\sigma}_v^2)\sqrt{KN}}{\hat{\sigma}_v^2}\right] \tag{6.8}$$

$$P_d = Q\left[\frac{(t - \hat{\sigma}_v^2 - \|\boldsymbol{h}\|\sigma_s^2)\sqrt{KN}}{\sqrt{K\|\boldsymbol{h}\|^2\sigma_s^4 + 2\|\boldsymbol{h}\|\sigma_s^2\hat{\sigma}_v^2 + \hat{\sigma}_v^4}}\right] \tag{6.9}$$

Now, the worst case scenario of P_d and P_{fa}, represented by the notations P_d' and P_{fa}', can be analyzed as,

$$P'_{fa} = \max_{\hat{\sigma}_v^2 \in \left[\frac{1}{\beta}\sigma_v^2, \beta\sigma_v^2\right]} Q\left[\frac{(t-\hat{\sigma}_v^2)\sqrt{KN}}{\hat{\sigma}_v^2}\right] \tag{6.10}$$

$$P'_{fa} = Q\left[\frac{\left(\frac{t}{\sigma_v^2} - \beta\right)\sqrt{KN}}{\beta}\right] \tag{6.11}$$

$$P'_{d} = \min_{\hat{\sigma}_v^2 \in \left[\frac{1}{\beta}\sigma_v^2, \beta\sigma_v^2\right]} Q\left(\frac{(t-\hat{\sigma}_v^2 - \|\boldsymbol{h}\|\sigma_s^2)\sqrt{KN}}{\sqrt{K\|\boldsymbol{h}\|^2\sigma_s^4 + 2\|\boldsymbol{h}\|\sigma_s^2\hat{\sigma}_v^2 + \hat{\sigma}_v^4}}\right) \tag{6.12}$$

$$P'_{d} = Q\left[\frac{\left(\frac{t}{\sigma_v^2} - \rho\frac{1}{\beta}\right)\sqrt{KN}}{\sqrt{K\rho^2 + 2\rho\frac{1}{\beta} + \frac{1}{\beta^2}}}\right] \tag{6.13}$$

By solving (6.11) and (6.13) and eliminating t, we get,

$$N = \frac{\left[\beta Q^{-1}(P'_{fa}) - Q^{-1}(P'_d)\sqrt{K\rho^2 + 2\rho\frac{1}{\beta} + \frac{1}{\beta^2}}\right]^2}{K\left[\rho - \left(\beta - \frac{1}{\beta}\right)\right]^2} \tag{6.14}$$

(6.14) gives the expression of the number of the required samples to detect a signal with the given P_{fa}, P_d and SNR, which is also an extension of the result in [25] for a multi-sensor ED. The number of samples N goes to infinity as SNR approaches $\beta - \frac{1}{\beta}$. This condition is known as the SNR Wall condition, which means that under this SNR Wall value we cannot achieve the required performance in a given level of noise uncertainty even if the sample number is made sufficiently large. Thus, the SNR Wall is given by the expression as shown below,

$$SNR_{Wall}^{ED} = \left(\beta - \frac{1}{\beta}\right) \tag{6.15}$$

The SNR Wall makes clear that the Energy Detector cannot robustly detect the signal if the signal power is less than the uncertainty of the noise power, i.e.,

$$\sigma_s^2 < \left(\beta - \frac{1}{\beta}\right)\sigma_v^2 \tag{6.16}$$

It is clear that there is no difference in the SNR Wall expression by using a multi-antenna or a single antenna in ED. Fig. 6.2 illustrates the SNR Wall condition for single/multi antenna case and the variation of the sample complexity N, as the SNR approaches the SNR Wall for different levels of noise uncertainty.

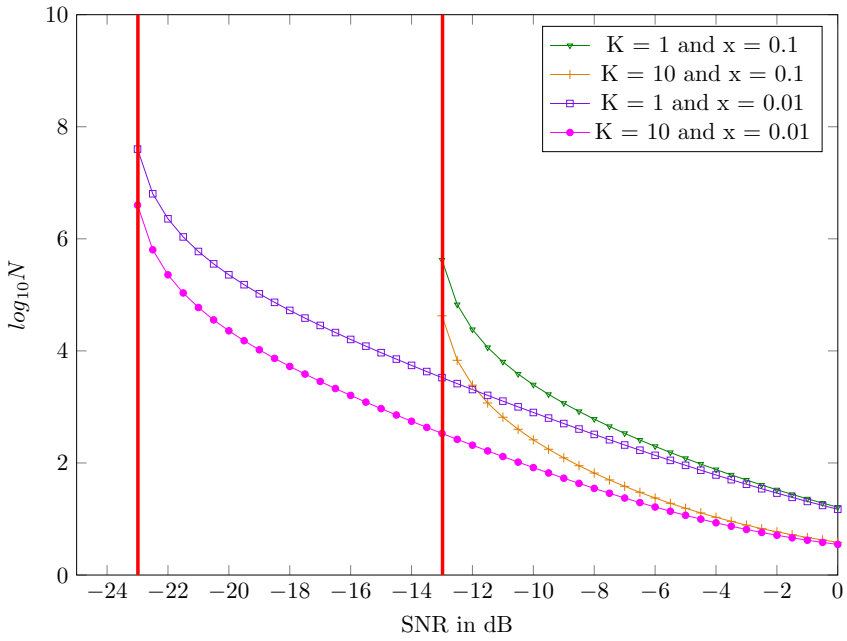


Figure 6.2: Variation of the sample complexity N as the SNR approaches the SNR Wall for ED, $K = 5$, $P'_d = 0.9$, $P'_{fa} = 0.1$ [$x = 10\log_{10}\beta$ in (6.14)].

6.2 Noise uncertainty distribution and formulation of uncertainty bound

Let us assume that the noise variance is estimated in S auxiliary slots with M noise only samples in each slot. Then the ML noise variance estimate using SKM noise-only samples obtained from K receivers with S slots each can be written as,

$$\hat{\sigma}_v^2(S) = \frac{1}{KSM}\sum_{s=1}^{S}\sum_{k=1}^{K}\sum_{m=1}^{M}|v_k(m)|^2 \tag{6.17}$$

If we focus on the distribution of the noise variance under the same scenario, let us consider a random variable V (normalized noise variance estimate), which associates a unique numerical value from $\left(\frac{\hat{\sigma}_v^2}{\sigma_v^2}\right)$ for each noise variance estimation, where $\hat{\sigma}_v^2$ corresponds to (6.17).

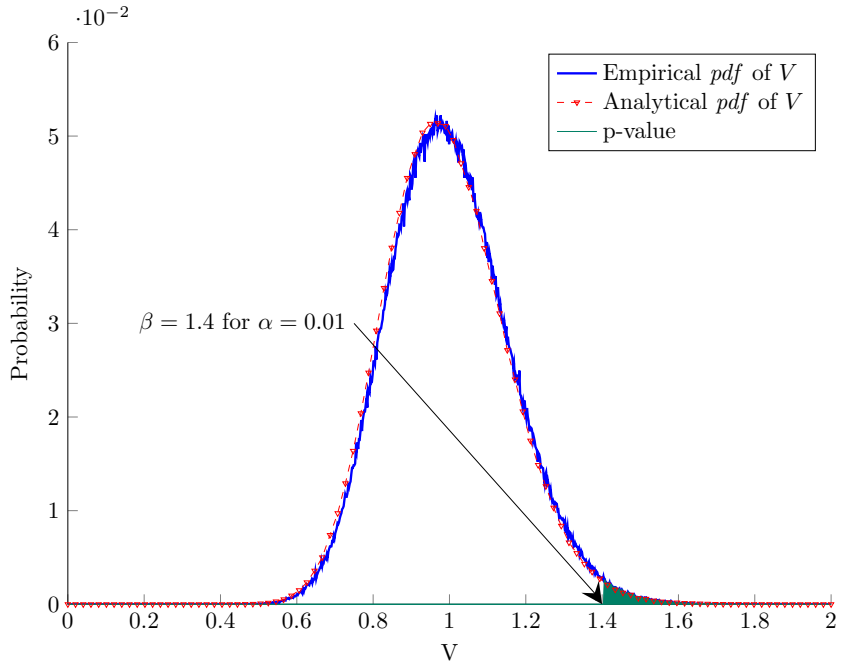


Figure 6.3: Probability distribution of the normalized noise variance estimate V, $S = 2$, $K = 2$, $M = 10$.

Now V can be written as,

$$V = \left(\frac{\hat{\sigma}_v^2}{\sigma_v^2}\right) \tag{6.18}$$

$$= \left(\frac{1}{2KMS}\sum_{s=1}^{S}\sum_{k=1}^{K}\sum_{m=1}^{M}\left|\frac{v_k(m)}{\sqrt{\sigma_v/2}}\right|^2\right) \tag{6.19}$$

It can be easily noted that the expression in (6.19) is the sum of squared standard normal noise samples, thus it has a chi-squared distribution with $2KMS$ degrees of freedom, so it follows,

$$V = \left(\frac{\chi^2_{2KMS}}{2KMS}\right) \tag{6.20}$$

where χ^2_{2KMS} represents a chi-squared random variable with $2KMS$ degrees of freedom.

Fig. 6.3 shows the *pdf* of the random variable V according to (6.20) compared to its empirical *pdf*. Now, the uncertainty bound β can easily be related to V by

using (6.5) and (6.20) as,

$$\beta = \max(V) \tag{6.21}$$

$$= \frac{F_{\chi^2}^{-1}(1-\alpha, 2KMS)}{2MKS} \tag{6.22}$$

where $F_{\chi^2}^{-1}()$ is the inverse CDF of a chi-squared distributed random variable and α is the significance level of a chi-squared distribution. As we know, a chi-squared distributed random variable can take any values in the range $(-\infty, +\infty)$, which leads (6.22) to infinity. In (6.22), $(1-\alpha)$ is the significance level, which gives an insight of percentage coverage of all possible values smaller than the value given by the inverse CDF. Thus,

$$\beta = \frac{F_{\chi^2}^{-1}(1-\alpha, 2KMS)}{2MKS} \tag{6.23}$$

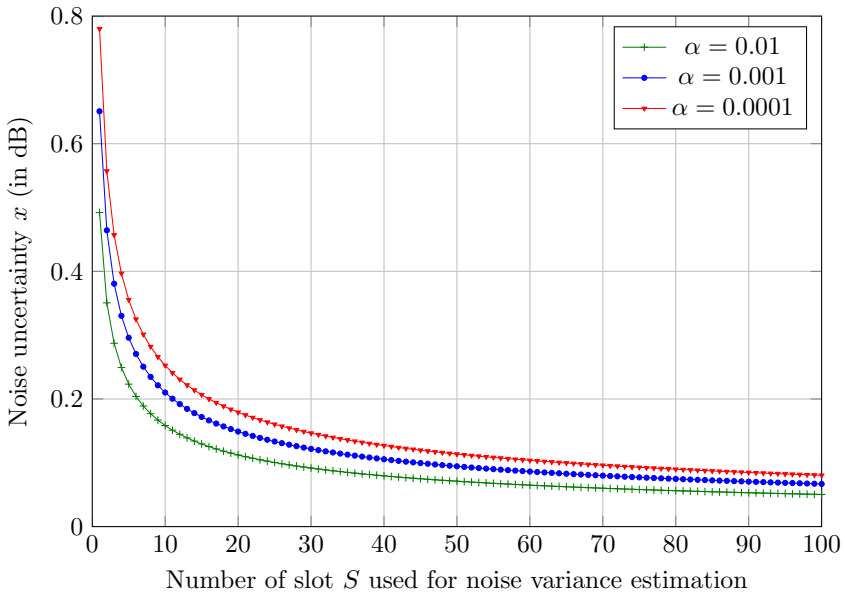


Figure 6.4: Variation of the noise uncertainty level as a function of the slots S used for noise variance estimation, $K = 4$, $M = 100$.

Finally, β can be expressed in terms of the total number of noise samples KMS considered for variance estimation as shown in (6.23) and can be easily evaluated. For example, for $\alpha = 0.01$, β is equal 1.4, which is also shown in Fig. 6.3. It is clear from (6.23) that the level of noise uncertainty decreases as the number of samples and the number of slots averaging the estimation of the noise variance increase.

6.3 Simulation results

The variation of the noise uncertainty bound β, with number of slots S used in noise variance estimation for different significance level, is illustrated in Fig. 6.4. Although the plot seems to have a steep slope for small values of S, for larger number of estimation slots S ($S > 30$) the slope starts to flatten suggesting that larger change in S is required for a small gain in the noise uncertainty bound. Fig. 6.5 illustrates the variation of the SNR Wall as a function of the number of slots used for noise variance estimation with $K = 4$ and $M = 100$. Each curve is plotted for different significance parameter α. As expected, the level of the SNR Wall decreases as the number of slots used in noise estimation increases. As a matter of fact, when $\alpha = 0.001$, the number of slots required to overcome the SNR Wall condition for a SNR level equal to -14 dB is $S > 60$.

Similarly, Fig. 6.6 plots the SNR Wall condition and the sample complexity N as a function of the number of auxiliary slots S with parameters $K = 4$, $\alpha = 0.0001$ and $M = 100$. In the asymptotics of number of slots S (in wide sense, in the asymptotics of total number of samples KMS), the noise uncertainty x (in dB) decreases to zero with $\hat{\sigma}_v^2 = \sigma_v^2$ resulting in no SNR Wall, which proves the finding of [28].

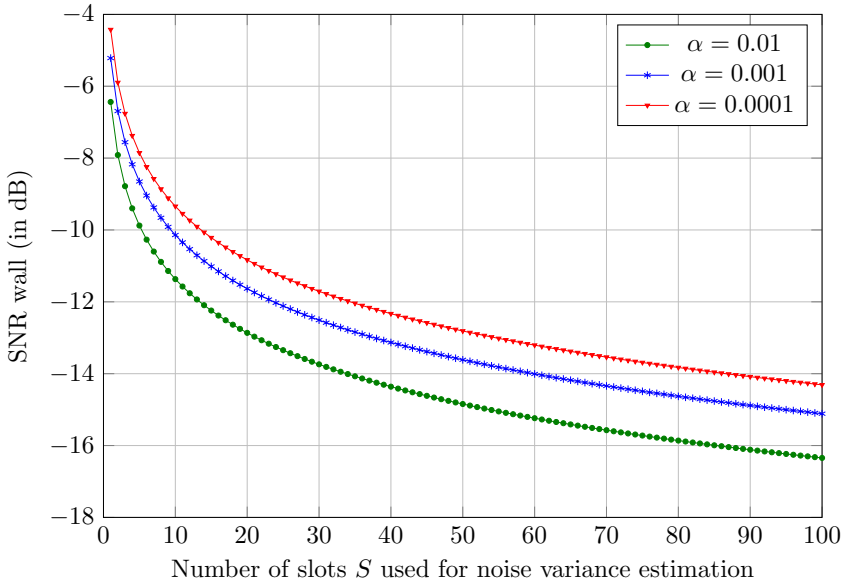


Figure 6.5: Variation of the SNR Wall level as a function of auxiliary slots S used for noise variance estimation, $K = 4$, $M = 100$.

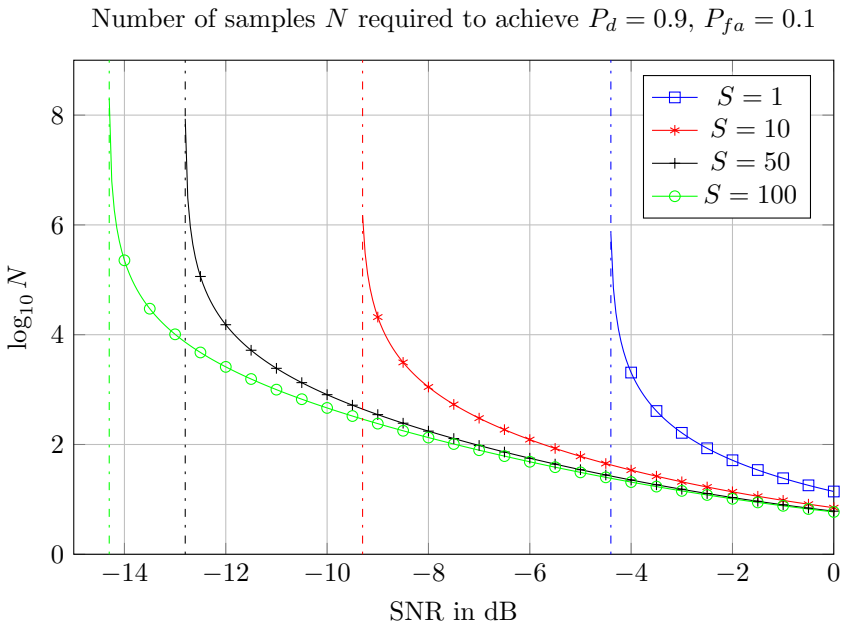


Figure 6.6: Variation of the sample complexity N as the SNR approaches the SNR Wall with different values of auxiliary slots S, $K = 4$, $M = 100$, $\alpha = 0.0001$.

Chapter 7

Effect of primary user traffic on Energy and Eigenvalue-based Detection algorithms

Among many practical imperfections and constraints for spectrum sensing mentioned in literature, the unknown traffic is one of the important constraint that limits the sensing performance of the secondary user. In the existing literature on spectrum sensing, the SUs are assumed to have a perfect knowledge of the exact time slot structure of PU transmissions, which guarantees that PU traffic transitions occur only at the beginning of the SU sensing frames. However, in practical sense, the SU may not have the knowledge of exact time slot structure of PU transmissions or it is also possible that the communications among PUs are not based on synchronous schemes at all [82, 83]. Thus, it is necessary to analyze the sensing performance of existing spectrum sensing methods under unknown PU traffic.

A first attempt to analyze the performance in unknown PU traffic has been presented in [84]. The author analyzed the sensing performance of the well known semi-blind spectrum sensing algorithms including Energy Detection (ED) and Roy's Largest Root Test (RLRT) under bursty PU traffic. The PU traffic model used in this chapter is limited to a constant burst length of PU data whose length is smaller than the SU sensing duration. However, the burst length of the PU may be varying with time following some stochastic models [85, 86]. A more general scenario in which the PU traffic transitions are completely random has been considered in [87, 88, 89, 90, 91]. Modeling PU traffic as a two state Markov process, authors in [87, 88, 89, 90] analyzed the effect of PU traffic on the sensing performance and the sensing-throughput trade-off considering ED as a sensing technique under the half duplex scenario. Moreover, the effect of multiple PUs traffic on the sensing-throughput trade-off of the secondary system has been studied in [91]. Although all the aforementioned contributions recognized the fact that the PU traffic affects the

sensing performance including sensing-throughput trade-off, none of them considered the analysis of the sensing performance of other spectrum sensing algorithms including Eigenvalue-Based Detection (EBD) techniques under unknown primary user traffic.

In this chapter, the effect of PU traffic on the performance of multi-antenna ED and RLRT is evaluated under the complex domain of PU signal, noise and channel. In contrast to the commonly used continuous-time Markov model in the existing literature, a realistic discrete time modelling of PU traffic is proposed which is only based on the discrete time distribution of PU free and busy periods. The proposed model is more realistic and simple compared to the continuous-time Markov model proposed in the previous literature [88, 89, 90, 91]. Moreover, an analytical performance evaluation of both decision statistics under the considered scenario is carried out. It is shown that the time varying PU traffic severely affects the performance of ED and RLRT, moreover the sensing performance decreases asthe length of the sensing slot increases, which is in contrast with the common property of spectrum sensing algorithms under known PU traffic scenario. Also, it is observed that the performance gain due to multiple antennas in the sensing unit is significantly suppressed by the effect of the PU traffic when the PU traffic transitions occur more frequently.

7.1 System model

We consider a single source scenario (single primary transmitter) whereas multiple antennas are employed by an SU. Suppose the SU has K antennas and each antenna receives N samples in each sensing slot. In a given sensing frame, the detector calculates its decision statistic T_D by collecting N samples from each one of the K antennas. Subsequently, the received samples are collected by the detector in the form of a $K \times N$ matrix $\boldsymbol{Y}$. When the primary transmissions are not based on some synchronous schemes or the sensing unit at the SU does not have any information about the primary traffic pattern, the received vector at the sensing unit may consist of partly the samples from one PU state and the remaining from alternate PU state. To simplify the scenario, we begin with the following classification of the sensing slots based on the PU traffic status, which is also illustrated in Fig. 7.1.

1. Steady State (SS) sensing slot: In such type of sensing slot, all the received samples in one sensing slot are obtained from the same PU state.

2. Transient State (TS) sensing slot: In such type of sensing slot, a part of the received samples within the sensing slot are obtained from one PU state and the remaining from the another PU state.

In general, the probabilities of receiving SS and TS sensing slots are dependent on the PUs traffic model. In contrast to the commonly used hypothesis definition in

spectrum sensing literature, we define two hypotheses in the following way:

$\mathcal{H}_0$: the channel is going to be free,
$\mathcal{H}_1$: the channel is going to be busy.

This hypothesis formulation implies that the decision is based on the PU status at the end of the sensing interval. Thus, in a TS sensing slot, a transition from the PU busy state to the PU free state is considered $\mathcal{H}_0$, while a transition from the PU free state to the PU busy state is considered $\mathcal{H}_1$. In the considered scenario, in an SS sensing interval, the generic received signal matrix under each hypothesis can be written as,

$$\boldsymbol{Y}_{SS} = \begin{cases} \boldsymbol{V}_{[K,N]} & (\mathcal{H}_0), \\ \boldsymbol{S}_{[K,N]} & (\mathcal{H}_1), \end{cases} \tag{7.1}$$

where $\boldsymbol{V}_{[K,N]} \triangleq [\boldsymbol{v}(1), \dots \boldsymbol{v}(n), \dots \boldsymbol{v}(N)]$ is the $K \times N$ noise matrix, $\boldsymbol{S}_{[K,N]} = \boldsymbol{h}_{[K,1]}\boldsymbol{s}_{[1,N]} + \boldsymbol{V}_{[K,N]}$ is the $K \times N$ received noisy signal matrix when PU signal is present. $\boldsymbol{h}_{[K,1]} = [h_1, \dots, h_K]^T$ is the channel vector and $\boldsymbol{s}_{[1,N]} \triangleq [s(1), \dots, s(n), \dots, s(N)]$ is a $1 \times N$ PU signal vector. And in the TS sensing interval, the generic received signal matrix under each hypothesis can be written as,

$$\boldsymbol{Y}_{TS} = \begin{cases} [\boldsymbol{S}_{[K,N-D_0]} | \boldsymbol{V}_{[K,D_0]}] & (\mathcal{H}_0), \\ [\boldsymbol{V}_{[K,N-D_1]} | \boldsymbol{S}_{[K,D_1]}] & (\mathcal{H}_1), \end{cases} \tag{7.2}$$

where D_0 represents the number of pure noise samples in TS sensing slot under $\mathcal{H}_0$, D_1 represents the number of noise plus PU signal samples in TS sensing slot under $\mathcal{H}_1$, $\boldsymbol{S}_{[K,N-D_0]} = \boldsymbol{h}_{[K,1]}\boldsymbol{s}_{[1,N-D_0]} + \boldsymbol{V}_{[K,N-D_0]}$ is the $(K \times N - D_0)$ received noisy signal matrix when PU signal is present only for $(N - D_0)$ sample periods. Similarly, $\boldsymbol{S}_{[K,D_1]} = \boldsymbol{h}_{[K,1]}\boldsymbol{s}_{[1,D_1]} + \boldsymbol{V}_{[K,D_1]}$ is the $K \times D_1$ received noisy signal matrix when PU signal is present only for D_1 sample periods. In each of these, the unknown primary transmitted signal $s(n)$ at time instant n is modelled as independent and identically distributed complex Gaussian with zero mean and variance $\sigma_s^2 : s(n) \sim \mathcal{N}_{\mathbb{C}}(0, \sigma_s^2)$. The noise sample $v_k(n)$ at the k^{th} antenna of the SU at the time instant n is also modelled as complex Gaussian with mean zero and variance $\sigma_v^2 : v_k(n) \sim \mathcal{N}_{\mathbb{C}}(0, \sigma_v^2)$. The channel coefficient h_k of the k^{th} antenna is assumed to be constant and memoryless during the sensing interval. The average SNR at the receiver is defined as, $\rho = \frac{\sigma_s^2 \|\boldsymbol{h}\|^2}{K\sigma_v^2}$, where $||.||$ denotes the Euclidean norm.

7.2 Characterization of Primary User traffic

In this section, we characterize the mathematical model of PU traffic. Based on the proposed stochastic PU traffic model, we construct the PU's probability transition

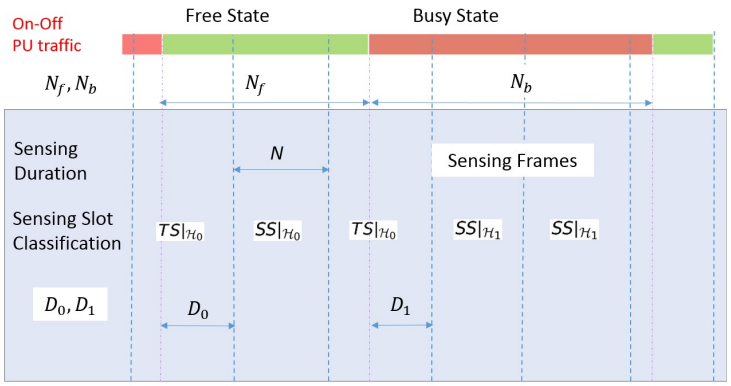


Figure 7.1: Primary user traffic scenario and sensing slot classification.

matrix, which leads to analytical formulation of the SU's probability of receiving SS sensing frame and TS sensing frame under null and alternate hypothesis.

In this paper, we model the PU traffic as a two state Markov process (On-Off process: PU 'On' representing busy state and PU 'Off' representing free state). The length of free as well as busy period are independent geometrically distributed random variables with parameters α and β, respectively. Essentially, the parameters α and β represent the state transition probabilities in single sample duration. The mean length of free period M_f and busy period M_b of PU traffic can be related to parameters α and β as, $M_f = \frac{1}{\alpha}$ and $M_b = \frac{1}{\beta}$, respectively.

At any time instant, the PU is in free state with probability $P_f = \frac{M_f}{M_b+M_f}$ and similarly, in the busy state with probability, $P_b = \frac{M_b}{M_b+M_f}$. We further assume that the parameters (α and β) of geometrically distributed length of PU free and busy periods are constant over time. Thus, the corresponding two-state Markovs process can be considered homogeneous in nature. Using this homogeneity property and the Chapman-Kolmogorov equation gives the PU n-step transition probability matrix as,

$$\begin{aligned} \boldsymbol{P}^n &= \begin{bmatrix} p_{00}^n & p_{01}^n \\ p_{10}^n & p_{11}^n \end{bmatrix} \\ &= \frac{1}{\alpha+\beta} \begin{bmatrix} \beta+\alpha(1-\alpha-\beta)^n & \alpha-\alpha(1-\alpha-\beta)^n \\ \beta-\beta(1-\alpha-\beta)^n & \alpha+\beta(1-\alpha-\beta)^n \end{bmatrix} \end{aligned} \tag{7.3}$$

which reduces to (7.4) for single step transition matrix as,

$$\boldsymbol{P} = \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix} \tag{7.4}$$

As already mentioned earlier in Sec. 7.1, the stochastic nature of the PU state transition gives a mixed nature of received signals in a TS sensing slot resulting in

random variables (RVs) D_0 and D_1. Thus, in each PU state transition from *Busy* to *Free State*, the sensing unit has to decide based on D_0 pure noise samples and $(N - D_0)$ noise plus primary signal samples which actually affects the overall sensing performance. Thus, with the support of above analysis and also keeping (7.1) and (7.2) in reference, it is clear that, to find the distribution of the decision statistic under different hypotheses, the prior deduction of the chances of occurrence of SS sensing slot, TS sensing slot, probability mass function (*pmf*) of D_0 and the *pmf* of D_1 are inevitable.

The following Lemmas compute the *pmf*s of D_0 and D_1 based on the two state PU traffic model described above.

Lemma 1. Given the number of samples in a sensing duration N, the probability transition matrix $\boldsymbol{P}$ as in (7.4) with comparable mean parameters M_f and M_b,

1. The probability of having D_0 noise only (PU signal free) samples in a TS sensing slot under $\mathcal{H}_0$ reduces to,

$$P_{D_0}(d_0)|_{\mathcal{H}_0} = \frac{1}{N-1} \tag{7.5}$$

2. The probability of having D_1 noise-plus-PU-signal samples in a TS sensing slot under $\mathcal{H}_1$ reduces to,

$$P_{D_1}(d_1)|_{\mathcal{H}_1} = \frac{1}{N-1} \tag{7.6}$$

Proof. As mentioned earlier during binary hypothesis formulation, the PU state transition from *Busy State* to *Free State* corresponds to $\mathcal{H}_0$ sensing slot and viceversa for $\mathcal{H}_1$. We consider thus, without loss of generality, while dealing with a $\mathcal{H}_0$ sensing slot, that the TS sensing slot occurs at the beginning of the PU *Free State*. Thus, given that the PU is initially in *Busy State*, the probability of having a PU state transition after $N - D_0$ samples leading D_0 pure noise samples in a TS sensing slot under $\mathcal{H}_0$ is given by,

$$P(TS, D_0, \mathcal{H}_0) = p_b \cdot p_{11}^{N-D_0-1} p_{10} p_{00}^{D_0} \tag{7.7}$$

The probability in (7.7) is the PU state transition probability after $N - D_0$ sample instances from a busy PU state to a free PU one. This probability can be normalized by the sum of the transition probabilities for all range of D_0 and obtain (7.9),

$$P_{D_0}(d_0)|_{\mathcal{H}_0} = \frac{p_b \cdot p_{11}^{N-d_0-1} p_{10} p_{00}^{d_0}}{\sum_{a=1}^{N-1} p_b \cdot p_{11}^{N-a-1} p_{10} p_{00}^{a}} \tag{7.8}$$

$$= \frac{(1-\beta)^{N-d_0-1}(1-\alpha)^{d_0}}{\sum_{a=1}^{N-1}(1-\beta)^{N-a-1}(1-\alpha)^{a}} \tag{7.9}$$

When the mean parameters M_f and M_b are close to each other, (7.9) can be approximated to:

$$P_{D_0}(d_0)|_{\mathcal{H}_0} = \frac{(1-\alpha)^{N-d_0-1}(1-\alpha)^{d_0}}{\sum_{a=1}^{N-1}(1-\alpha)^{N-a-1}(1-\alpha)^{a}} \tag{7.10}$$

$$= \frac{1}{N-1} \tag{7.11}$$

This proves the first claim. By using the same line of reasoning, the proof of the second claim is straightforward. □

The following Lemmas computes the probability of occurrence of SS sensing slot $p_{SS}|_{\mathcal{H}_0}$ under $\mathcal{H}_0$ and the probability of occurrence of TS sensing slot, which is the complementary of $p_{SS}|_{\mathcal{H}_0}$, i.e., $p_{TS}|_{\mathcal{H}_0} = 1 - p_{SS}|_{\mathcal{H}_0}$.

Lemma 2. Given the number of samples in a sensing duration N and the probability transition matrix $\boldsymbol{P}$ as in (7.4),

1. The probability of receiving SS sensing slot under $\mathcal{H}_0$ is given by,

$$P_{SS}|_{\mathcal{H}_0} = \frac{1}{1+\alpha\sum_{d_0=1}^{N}(1-\beta)^{N-d_0-1}(1-\alpha)^{d_0-N}}. \tag{7.12}$$

2. The probability of receiving SS sensing slot under $\mathcal{H}_1$ is given by,

$$P_{SS}|_{\mathcal{H}_1} = \frac{1}{1+\beta\sum_{d_1=1}^{N}(1-\alpha)^{N-d_1-1}(1-\beta)^{d_1-N}}. \tag{7.13}$$

Proof. First of all, the probability of having no PU state transition under $\mathcal{H}_0$is given by,

$$P(SS, \mathcal{H}_0) = p_f \cdot p_{00}^{N} \tag{7.14}$$

Similarly, the probability of having a PU state transition from *Busy State* to *Free State* is given by,

$$P(TS, \mathcal{H}_0) = P\left(\begin{array}{c}\text{PU is in}\\ \textit{Busy State}\end{array}\right) \cdot P\left(\begin{array}{c}\text{PU transits from } \textit{Busy} \text{ to } \textit{Free}\\ \textit{State} \text{ during sensing interval}\end{array}\right) \tag{7.15}$$

Essentially, the PU state transition may occur at any time during the sensing duration,

$$P(TS, \mathcal{H}_0) = p_b \cdot \sum_{d_0}^{N-1} p_{11}^{N-d_0-1} p_{10} p_{00}^{d_0} \tag{7.16}$$

Thus, by using (7.4) and (7.16), the probability of having a SS sensing slot belonging to $\mathcal{H}_0$ is given by,

$$P_{SS}|_{\mathcal{H}_0} = \frac{P(SS, \mathcal{H}_0)}{P(SS, \mathcal{H}_0) + P(TS, \mathcal{H}_0)} \tag{7.17}$$

$$= \frac{p_f \cdot p_{00}^N}{p_f \cdot p_{00}^N + p_b \cdot \sum_{d_0}^{N-1} p_{11}^{N-d_0-1} p_{10} p_{00}^{d_0}} \tag{7.18}$$

By replacing p_f, p_b and the elements of the probability transition matrix by their respective expressions in terms of α and β, further simplification yields (7.12) and proves claim 1 of Lemma 2.

Through the same line of reasoning, the proof of claim 2 of Lemma 1 is straightforward.

□

Corollary 1. When the mean parameters M_f and M_b are comparable, the probabilities in (7.12) and (7.13) reduce to simple expressions given by,

$$P_{SS}|_{\mathcal{H}_0} = \frac{1}{1 + (N-1)\alpha} \tag{7.19}$$

$$P_{SS}|_{\mathcal{H}_1} = \frac{1}{1 + (N-1)\beta} \tag{7.20}$$

7.3 ED performance analysis

Energy Detection computes the average energy of the received signal matrix $\boldsymbol{Y}$ normalized by the noise variance σ_v^2 and compares it with a predefined threshold. With respect to Sec. 2.3.1, we define here T_{ED} as

$$T_{ED} = \frac{1}{\sigma_v^2} \sum_{k=1}^{K} \sum_{n=1}^{N} |y_k(n)|^2. \tag{7.21}$$

where the normalization by N and K is omitted in order to ease the performance analysis in this section. First, it is necessary to express the *pdf* of the decision statistic in case of unknown primary traffic. The following theorem computes the *pdf* of the ED decision statistic under both hypotheses using the PU traffic characterization presented in Sec. 7.2.

Theorem 1. Given a multi-antenna sensing unit with K receiving antennas, N received samples in each slot and the random PU traffic with geometrically distributed free state duration, the *pdf* of the ED decision statistic under respectively $\mathcal{H}_0$ and $\mathcal{H}_1$ is given by:

$$f_{T_{ED}|_{H_0}}(x) = p_{SS}|_{H_0} f_{\mathcal{G}}(x, KN,1) + p_{TS}|_{H_0} \sum_{d_0=1}^{N-1} P_{D_0}(d_0) \left[f_{\mathcal{G}}(x,2Kd_0,1) \right.$$
$$+ f_{\mathcal{G}}(x, N-d_0, K\rho) + \; f_{\mathcal{G}}(x, K(N-d_0),1)$$
$$\left. + f_{\mathcal{N}}(x,0,2\rho K(N-d_0)) \right] \tag{7.22}$$

$$f_{T_{ED}|_{H_1}}(x) = p_{SS}|_{\mathcal{H}_1} \left(f_{\mathcal{G}}(x, N, K\rho) + f_{\mathcal{G}}(x, KN,1) + f_{\mathcal{N}}(x,0,2\rho KN) \right)$$
$$+ p_{TS}|_{\mathcal{H}_1} \sum_{d_1=1}^{N-1} P_{D_1}(d_1) \left[f_{\mathcal{G}}(x, d_1, K\rho) + f_{\mathcal{G}}(x, Kd_1,1) \right.$$
$$\left. + f_{\mathcal{N}}(x,0,2\rho Kd_1) + f_{\mathcal{G}}(x, K(N-d_1),1) \right] \tag{7.23}$$

where $f_{\mathcal{G}}(x, \alpha, \theta)$ is the *pdf* of a Gamma distribution with shape parameter α and scale parameter θ, while $f_{\mathcal{N}}(x, \mu, \sigma^2)$ is the *pdf* of a Gaussian distribution with mean μ and variance σ^2.

Proof. As noted from Sec. 7.1, the term within the summation in (7.21) is different for the SS sensing slot and TS sensing slot. Under null hypothesis $\mathcal{H}_0$, the ED decision statistic in (7.21) can be decomposed as a probabilistic sum of $T_{ED}^{SS}|_{\mathcal{H}_0}$ and $T_{ED}^{TS}|_{\mathcal{H}_0}$.

$$T_{ED}|_{\mathcal{H}_0} = \frac{p_{SS}|_{\mathcal{H}_0}}{2} \sum_{k=1}^{K} \sum_{n=1}^{N} \left| \frac{v_k(n)}{\sigma_v/\sqrt{2}} \right|^2 + \frac{p_{TS}|_{\mathcal{H}_0}}{2} \left[\sum_{k=1}^{K} \sum_{n=1}^{D_0} \left| \frac{v_k(n)}{\sigma_v/\sqrt{2}} \right|^2 \right.$$
$$\left. + \sum_{k=1}^{K} \sum_{n=N-D_0+1}^{N} \left| \frac{h_k s(n) + v_k(n)}{\sigma_v/\sqrt{2}} \right|^2 \right] \tag{7.24}$$

The first sum in (7.24) is in fact the sum of standard normal random variables, hence it follows a central chi-squared distribution with $2KN$ degrees of freedom. D_0 is a random variable with *pmf* given by (7.11) thus, (7.24) can be rewritten as,

$$T_{ED}|_{\mathcal{H}_0} = \frac{p_{SS}|_{\mathcal{H}_0}}{2} \chi^2_{2KN} + \frac{p_{TS}|_{\mathcal{H}_0}}{2} \sum_{d_0=1}^{N-1} P_{D_0}(d_0) \left[\sum_{k=1}^{K} \sum_{n=1}^{d_0} \left| \frac{v_k(n)}{\sigma_v/\sqrt{2}} \right|^2 \right.$$
$$\left. + \sum_{k=1}^{K} \sum_{n=N-d_0+1}^{N} \left| \frac{h_k s(n) + v_k(n)}{\sigma_v/\sqrt{2}} \right|^2 \right] \tag{7.25}$$

Next, the distribution of each remaining sum in (7.25) can be derived as,

$$T_{ED}|_{\mathcal{H}_0} = \frac{p_{SS}|_{\mathcal{H}_0}}{2}\chi^2_{2KN} + \frac{p_{TS}|_{\mathcal{H}_0}}{2}\sum_{d_0=1}^{N-1} P_{D_0}(d_0)\left[\chi^2_{2Kd_0}\right.$$
$$\left. + K\rho\chi^2_{2(N-d_0)} + \chi^2_{2K(N-d_0)} + \mathcal{N}(0,2\rho(N-d_0)K)\right] \tag{7.26}$$

where χ^2_A represents a chi-squared random variable with A degrees of freedom. and $\mathcal{N}(\mu,\sigma^2)$ represents a normal random variable with mean μ and variance σ^2.

By exploiting that $\chi^2_A \sim \mathcal{G}(\frac{A}{2},2)$, where $\mathcal{G}(\alpha,\theta)$ represents a Gamma random variable with shape parameter α and scale parameter θ, we can substitute the chi-squared random variables in (7.26) with the Gamma ones and, taking into account that $k\mathcal{G}(\alpha,\theta) = \mathcal{G}(\alpha,k\theta)$, we obtain,

$$T_{ED}|_{\mathcal{H}_0} = p_{SS}|_{\mathcal{H}_0}\mathcal{G}(KN,1) + p_{TS}|_{\mathcal{H}_0}\sum_{d_0=1}^{N-1} P_{D_0}(d_0)\left[\mathcal{G}(Kd_0,1)\right.$$
$$\left. + \mathcal{G}(N-d_0,K\rho) + \mathcal{G}(K(N-d_0),1) + \mathcal{N}(0,2\rho(N-d_0)K)\right] \tag{7.27}$$

Since the goal is to find the *pdf* of the sum in (7.21) under $\mathcal{H}_0$, we replace the random variables in (7.27) with their respective *pdf*s and we finally obtain (7.22).

In a similar manner, under the alternate hypothesis $\mathcal{H}_1$, the ED decision statistic in (7.21) can be decomposed as a probabilistic sum of $T^{SS}_{ED}|_{\mathcal{H}_1}$ and $T^{TS}_{ED}|_{\mathcal{H}_1}$.

$$T_{ED}|_{\mathcal{H}_1} = \frac{p_{SS}|_{\mathcal{H}_1}}{2}\sum_{k=1}^{K}\sum_{n=1}^{N}\left|\frac{h_k s(n)+v_k(n)}{\sigma_v/\sqrt{2}}\right|^2 + \frac{p_{TS}|_{\mathcal{H}_1}}{2}$$
$$\cdot\left[\sum_{k=1}^{K}\sum_{n=1}^{D_1}\left|\frac{h_k s(n)+v_k(n)}{\sigma_v/\sqrt{2}}\right|^2 + \sum_{k=1}^{K}\sum_{n=N-D_1+1}^{N}\left|\frac{v_k(n)}{\sigma_v/\sqrt{2}}\right|^2\right] \tag{7.28}$$

We use the fact that D_1 is a random variable,

$$T_{ED}|_{\mathcal{H}_1} = \frac{p_{SS}|_{\mathcal{H}_1}}{2}\sum_{k=1}^{K}\sum_{n=1}^{N}\left|\frac{h_k s(n)+v_k(n)}{\sigma_v/\sqrt{2}}\right|^2 + \frac{p_{TS}|_{\mathcal{H}_1}}{2}\sum_{d_1=1}^{N-1} P_{D_1}(d_1)$$
$$\cdot\left[\sum_{k=1}^{K}\sum_{n=1}^{d_1}\left|\frac{h_k s(n)+v_k(n)}{\sigma_v/\sqrt{2}}\right|^2 + \sum_{k=1}^{K}\sum_{n=N-d_1+1}^{N}\left|\frac{v_k(n)}{\sigma_v/\sqrt{2}}\right|^2\right] \tag{7.29}$$

and we derive the distribution of each sum in (7.29),

$$T_{ED}|_{\mathcal{H}_1} = p_{SS}|_{\mathcal{H}_1}\left(\frac{K\rho\chi^2_{2N}}{2} + \frac{\chi^2_{2KN}}{2} + \mathcal{N}(0,2\rho KN)\right) + p_{TS}|_{\mathcal{H}_1}\sum_{d_1=1}^{N-1} P_{D_1}(d_1)\left[\frac{K\rho\chi^2_{2d_1}}{2} + \frac{\chi^2_{2Kd_1}}{2} + \mathcal{N}(0,2\rho Kd_1) + \frac{\chi^2_{2K(N-d_1)}}{2}\right] \tag{7.30}$$

By using the chi-squared to gamma random variable transformation in (7.30), we obtain,

$$T_{ED}|_{\mathcal{H}_1} = p_{SS}|_{\mathcal{H}_1}\left(\mathcal{G}(N,K\rho) + \mathcal{G}(KN,1) + \mathcal{N}(0,2\rho KN)\right) + p_{TS}|_{\mathcal{H}_1}\sum_{d_1=1}^{N-1} P_{D_1}(d_1)\left[\mathcal{G}(d_1,K\rho) + \mathcal{G}(Kd_1,1) + \mathcal{N}(0,2\rho Kd_1) + \mathcal{G}(K(N-d_1),1)\right] \tag{7.31}$$

Finally, we replace the random variables in (7.31) with their respective *pdf*s and we obtain (7.23). □

In essence, the *pdf*s in (7.22) and (7.23) consist of the sum of independent random variables. From a statistical point of view, the sum of two independent *pdf*s can be expressed as the convolution between the two *pdf*s [92], or as the product of their characteristic functions. By using the latter approach, (7.22) and (7.23) can be easily evaluated by using standard Fast Fourier Transform (FFT) techniques.

1. *False alarm probability:* Given the *pdf* of the decision statistic in (7.22), we can compute the false alarm probability. Under $\mathcal{H}_0$, the PU is in free state at the end of the sensing interval, but the decision statistic is erroneously above the threshold τ and the PU signal is declared present. In order to define the probability of false alarm P_{fa} in our case, the following Corollary of Theorem 1 holds.

   Corollary 2. The false alarm probability of the ED test under unknown PU traffic and complex signal space scenario is given by:

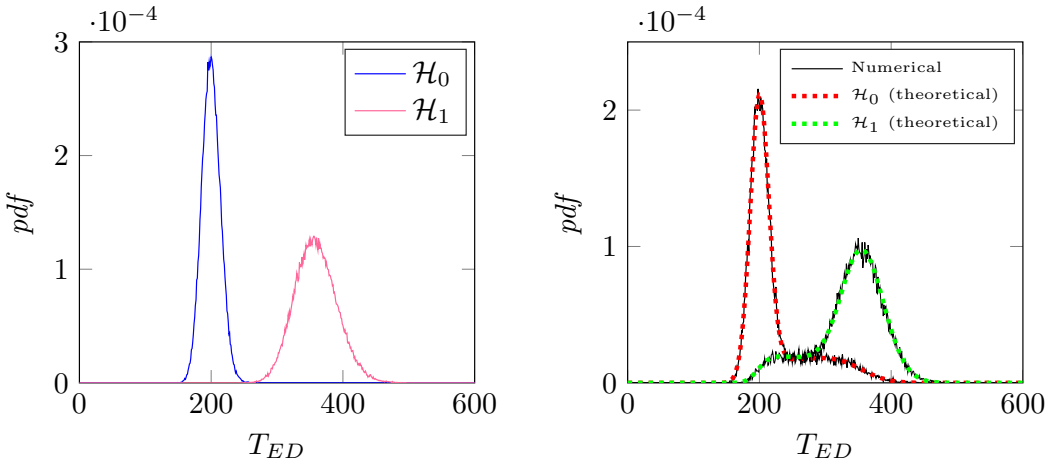
$$P_{fa} = P(T_{ED}|_{\mathcal{H}_0} \geq \tau) \equiv \int_{\tau}^{+\infty} f_{T_{ED}|_{H_0}}(x)dx. \tag{7.32}$$

2. *Detection Probability:* Given the *pdf* of the decision statistic in (7.23), we can compute the detection probability. Under $\mathcal{H}_1$, the PU is in busy state at the end of the sensing interval. Under this scenario, if the decision statistic is above the threshold, the PU signal is declared present. The following Corollary of Theorem 1 holds in order to define the probability of detection P_d.

   Corollary 3. The detection probability of the ED test under unknown PU traffic and the complex signal space scenario is given by:

$$P_d = P(T_{ED}|_{\mathcal{H}_1} \geq \tau) \equiv \int_{\tau}^{+\infty} f_{T_{ED}|_{H_1}}(x)dx. \tag{7.33}$$

7.4 Numerical results for multi-antenna ED



(a) Known PU Traffic (synchronized PU-SU sensing slots)

(b) Unknown PU Traffic

Figure 7.2: ED probability density functions, $N = 50$, $K = 4$, $M_f = 150$, $M_b = 150$, SNR $= -6$ dB.

In this section, the effect of PU traffic on the multi-antenna ED is analyzed based on the traffic model developed in Sec. 7.1. The length of the free and busy periods of the PU traffic are measured in terms of the discrete number of samples where each of them has Geometric distribution with mean parameters M_f and M_b, respectively as described in Sec. 7.2. The analytical expressions derived in Sec. 7.5 are validated via numerical simulations.

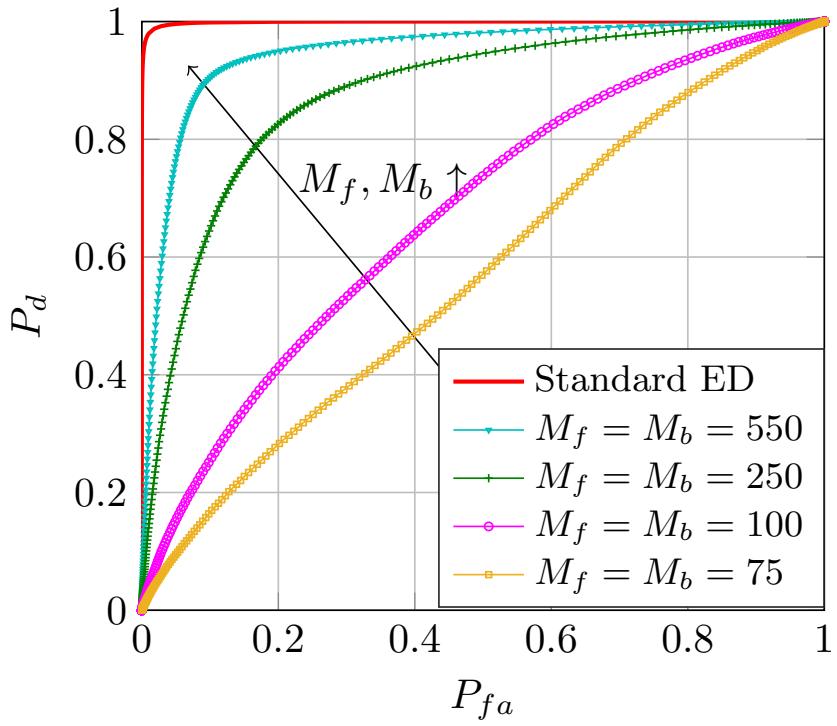


Figure 7.3: ROC curve performance, $N = 100$, $K = 4$, SNR $= -6$ dB.

In Fig. 7.2a and 7.2b, the *pdf* of the decision statistic under ideal PU-SU sensing slot synchronization is compared with the *pdf* of the decision statistic under unknown PU traffic considering both hypotheses. In addition, the accuracy of derived analytical expressions of the *pdf*s is confirmed by the results presented in Fig. 7.2b, where the theoretical formulas are compared against the numerical results obtained by Monte Carlo simulation. The perfect match of the theoretical and the numerical *pdf*s validates the derived analytical expressions. Fig. 7.3 illustrates the ROC curve of the ED for different values of the mean free and busy period of the PU traffic. It shows that as the mean free and busy periods of the primary traffic increase, the detection performance of the SU also increases. The conventional model with perfect synchronization of the PU-SU sensing slots performs better than the one with unknown PU traffic.

The variation of the sensing performance of the detector for different number of receiving antennas is plotted in Fig. 4.8. It can be observed that, unlike the rapid increase in the sensing performance as the number of receiving antennas increase under the synchronized PU-SU sensing slot scenario (rapid decrease in misdetection probability as the number of receiving antennas increase), the sensing performance is almost constant even if we increase the number of antennas under unknown PU traffic.

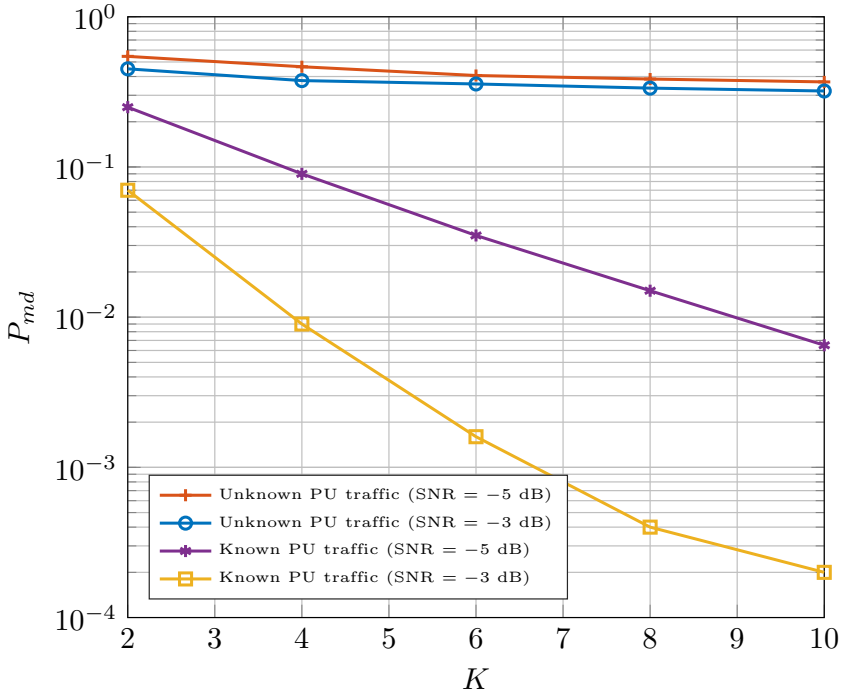


Figure 7.4: Probability of misdetection vs. number of antennas K, $N = 25$, $M_f = 62$, $M_b = 62$, $P_{fa} = 0.1$.

During a TS sensing slot, from each receiving antenna, the received signal samples are a mixture of pure noise samples and samples with both noise and PU signal. Thus, even if we use multiple antennas, the nature of the received signal does not change much and this is the reason why the sensing performance improvement is suppressed by the unknown PU traffic (more specifically, the TS sensing performance), when the length of the free and busy periods of PU traffic are quite small (a few multiples of the length of the sensing window). Fig. 7.5 plots the probability of misdetection as a function of the SNR. The convergence of the performance towards the ideal performance of ED can be noted for increasing lengths of the PU free and busy periods.

7.5 RLRT performance analysis

In this section, we perform the analysis under PU traffic on EBD algorithms. With reference to Sec. 2.2.1, given a $K \times N$ received signal matrix $\boldsymbol{Y}$, the sample covariance matrix is defined as, $\boldsymbol{R} \triangleq \frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{H}}$ and $\lambda_1 \geq \ldots \geq \lambda_K$ its eigenvalues sorted in the decreasing order. We analyze in detail the RLRT method, whose test statistic is defined as:

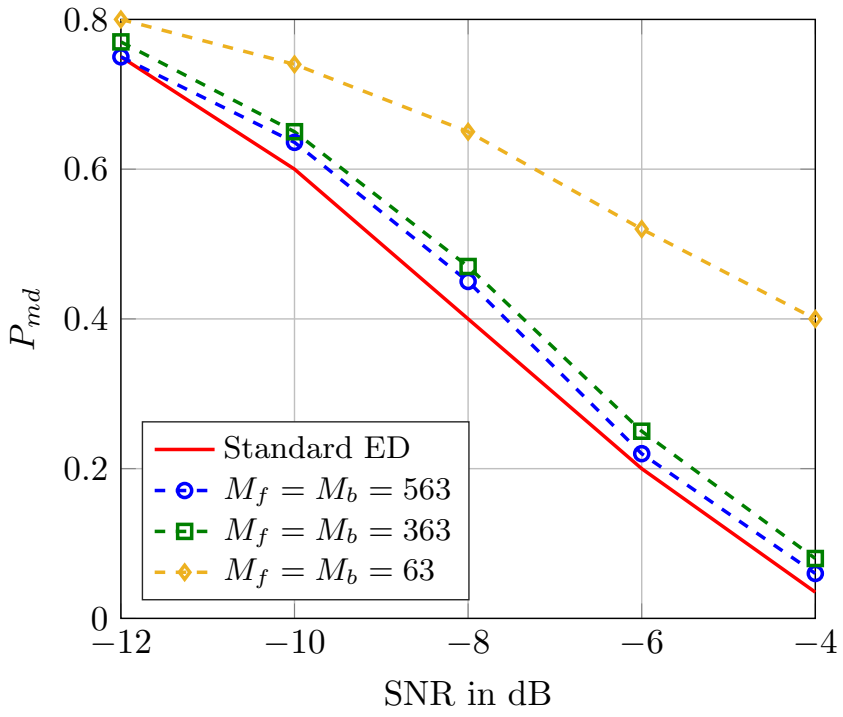


Figure 7.5: Probability of misdetection vs. SNR, $N = 25$, $K = 4$, $P_{fa} = 0.1$.

$$T_{RLRT} \triangleq \frac{\lambda_1}{\sigma_v^2}. \tag{7.34}$$

Although we only consider RLRT in this analysis, the results can be extended to other EBD methods as well. First, it is necessary to express the *pdf* of RLRT for the case of unknown PU traffic. The following theorem computes the *pdf* of the test decision statistic under both hypotheses using the PU traffic characterization presented in Sec. 7.2.

Theorem 2. Given a multi-antenna sensing unit with K receiving antennas, N received samples in each slot and random PU traffic with geometrically distributed free and busy state duration, let $c \triangleq \frac{K}{N}$ and N_s an independent parameter and define:

$$\mu_1(N_s) = \left(\frac{N_s}{N}K\rho + 1\right)\left(1 + \frac{K-1}{N_s K\rho}\right) \tag{7.35}$$

$$\sigma_1^2(N_s) = \frac{N_s}{N^2}(K\rho+1)\left(1 - \frac{K-1}{N_s K^2\rho^2}\right). \tag{7.36}$$

$$\mu_{N,K} = \left[1+\sqrt{c}\right]^2 \tag{7.37}$$

$$\sigma_{N,K} = N^{-2/3}\left[1+\sqrt{c}\right]\left[1+\frac{1}{\sqrt{c}}\right]^{1/3} \tag{7.38}$$

Then, the *pdf*s of RLRT decision statistic under $\mathcal{H}_0$ and $\mathcal{H}_1$ are given by (7.39) and (7.40) respectively,

$$\begin{aligned} f_{T_{RLRT}|\mathcal{H}_0}(x) &= P_{SS}|_{\mathcal{H}_0} f_{TW2}\left(\frac{x-\mu_{N,K}}{\sigma_{N,K}}\right) \\ &\quad + P_{TS}|_{\mathcal{H}_0}\sum_{d_0=1}^{N-1} P_{D_0}(d_0) f_D(x, N-d_0), \end{aligned} \tag{7.39}$$

$$f_{T_{RLRT}|\mathcal{H}_1}(x) = P_{SS}|_{\mathcal{H}_1} f_D(x,N) + P_{TS}|_{\mathcal{H}_1}\sum_{d_1=1}^{N-1} P_{D_1}(d_1) f_D(x,d_1) \tag{7.40}$$

where,

$$f_D(x,d) = \begin{cases} f_{\mathcal{N}}(\mu_1(d), \sigma_1^2(d)) & \text{if, } d > \frac{K-1}{K^2\rho^2}, \\ f_{TW2}(\mu_{N,K}, \sigma_{N,K}) & \text{otherwise} \end{cases} \tag{7.41}$$

In (7.41), $f_{\mathcal{N}}(\mu_1(d), \sigma_1^2(d))$ denotes a Gaussian *pdf* with mean $\mu_1(N_s)$ and variance $\sigma_1^2(N_s)$ provided in (7.35) and (7.36) at $N_s = d$. Next, $f_{TW2}(\mu_{N,K}, \sigma_{N,K})$ is the *pdf* of the Tracy-Widom distribution of order 2 with parameters $\mu_{N,K}$ and $\sigma_{N,K}$ provided in (7.37) and (7.38).

Proof. As noted from Sec. 7.1, the nature of the received signal matrix is different for the SS sensing slot and TS sensing slot. Under null hypothesis $\mathcal{H}_0$, the sample covariance matrix $\boldsymbol{R}|_{\mathcal{H}_0}$ can be decomposed as the probabilistic sum of $\boldsymbol{R}_{SS}|_{\mathcal{H}_0}$ and $\boldsymbol{R}_{TS}|_{\mathcal{H}_0}$ in the following way,

$$\boldsymbol{R}|_{\mathcal{H}_0} = P_{SS}|_{\mathcal{H}_0}\boldsymbol{R}_{SS}|_{\mathcal{H}_0} + P_{TS}|_{\mathcal{H}_0}\boldsymbol{R}_{TS}|_{\mathcal{H}_0} \tag{7.42}$$

Since each sensing slot is independent from one another, we treat each covariance matrix in (7.42) independently. Given a SS sensing slot under null hypothesis, all

the received samples $y_k(n)$ are homogeneous in nature comprising the independent and identically distributed Gaussian noise samples with mean zero and variance σ_v^2. Thus, the sample covariance matrix $\boldsymbol{R}_{SS}|_{\mathcal{H}_0}$ follows a Wishart distribution whose largest eigenvalue normalized by noise variance can be expressed by a TW2 [60].

$$\frac{\lambda_1^{SS}|_{\mathcal{H}_0}}{\sigma_v^2} = f_{TW2}\left(\frac{x-\mu_{N,K}}{\sigma_{N,K}}\right) \tag{7.43}$$

where $\mu_{N,K}$ and $\sigma_{N,K}$ are given in (7.37) and (7.38).

Next, given a TS sensing slot under null hypothesis, all the received samples $y_k(n)$ are not homogeneous in nature. To provide a better understanding, we express the covariance matrix in a TS sensing slot under $\mathcal{H}_0$ as,

$$\boldsymbol{R}_{TS}|_{\mathcal{H}_0} = \boldsymbol{R}_S(N-D_0) + \boldsymbol{R}_N(D_0), \tag{7.44}$$

where,

$$\boldsymbol{R}_S(N-D_0) \triangleq \frac{1}{N-D_0}\boldsymbol{S}_{[K,N-D_0]}\boldsymbol{S}^H_{[K,N-D_0]}, \tag{7.45}$$

$$\boldsymbol{R}_N(D_0) \triangleq \frac{1}{D_0}\boldsymbol{V}_{[K,D_0]}\boldsymbol{V}^H_{[K,D_0]} \tag{7.46}$$

are the partial covariance matrices constructed respectively from signal-plus-noise and only-noise samples. $\boldsymbol{R}_S(N-D_0)$ is a standard spiked population covariance matrix of rank-1 and $\boldsymbol{R}_N(D_0)$ is Wishart matrix. The largest eigenvalue of $\boldsymbol{R}_N(D_0)$ is negligible compared to the largest eigenvalue of $\boldsymbol{R}_S(N-D_0)$ provided that the signal identifiability condition is met [84]. It is known that the fluctuations of the largest eigenvalue of a rank-1 spiked population matrix normalized by the noise variance are asymptotically Gaussian [60, 44] if the signal identifiability condition is met, otherwise its distribution is again a TW2.

$$\frac{\lambda_1^{TS}|_{\mathcal{H}_0}}{\sigma_v^2} = f_D\left(x,(N-D_0)\right) \tag{7.47}$$

By using the results from (7.43) and (7.47), the RLRT decision statistic under null hypothesis can be written as,

$$T_{RLRT}|_{\mathcal{H}_0} = \frac{\lambda_1|_{\mathcal{H}_0}}{\sigma_v^2} \tag{7.48}$$

$$= p_{SS}|_{\mathcal{H}_0}\frac{\lambda_1^{SS}|_{\mathcal{H}_0}}{\sigma_v^2} + p_{TS}|_{\mathcal{H}_0}\frac{\lambda_1^{TS}|_{\mathcal{H}_0}}{\sigma_v^2} \tag{7.49}$$

$$= p_{SS}|_{\mathcal{H}_0} f_{TW}\left(\frac{t-\mu_{N,K}}{\sigma_{N,K}}\right) + p_{TS}|_{\mathcal{H}_0} f_D(x, N-D_0) \tag{7.50}$$

By using the fact that D_0 is a random variable distributed as in (7.5), we obtain the final distribution of the decision statistic of RLRT test under null hypothesis as in (7.39).

We consider now the case when the PU signal is present (hypothesis $\mathcal{H}_1$). In this case, an error is made if the presence of the PU signal is not detected. Under alternate hypothesis $\mathcal{H}_1$, the sample covariance matrix $\boldsymbol{R}|_{\mathcal{H}_1}$ can be decomposed as the probabilistic sum of $\boldsymbol{R}_{SS}|_{\mathcal{H}_1}$ and $\boldsymbol{R}_{TS}|_{\mathcal{H}_1}$.

$$\boldsymbol{R}|_{\mathcal{H}_1} = p_{SS}|_{\mathcal{H}_1}\boldsymbol{R}_{SS}|_{\mathcal{H}_1} + p_{TS}|_{\mathcal{H}_1}\boldsymbol{R}_{TS}|_{\mathcal{H}_1} \tag{7.51}$$

Since $\boldsymbol{R}_{SS}|_{\mathcal{H}_1}$ is a standard spiked population covariance matrix of rank-1, the distribution of the largest eigenvalue normalized by the noise variance in a SS sensing slot under $\mathcal{H}_1$ can be approximated as in [44],

$$\frac{\lambda_1^{SS}|_{\mathcal{H}_1}}{\sigma_v^2} = f_D(x, N) \tag{7.52}$$

With the same line of reasoning as in $\mathcal{H}_0$, we get,

$$\frac{\lambda_1^{TS}|_{\mathcal{H}_1}}{\sigma_v^2} = f_D(x, D_1) \tag{7.53}$$

By using (7.52) and (7.53), the distribution of the RLRT decision statistic under alternate hypothesis can be written as,

$$T_{RLRT}|_{\mathcal{H}_1} = \frac{\lambda_1|_{\mathcal{H}_1}}{\sigma_v^2} \tag{7.54}$$

$$= p_{SS}|_{\mathcal{H}_1}\frac{\lambda_1^{SS}|_{\mathcal{H}_1}}{\sigma_v^2} + p_{TS}|_{\mathcal{H}_1}\frac{\lambda_1^{TS}|_{\mathcal{H}_1}}{\sigma_v^2} \tag{7.55}$$

$$= p_{SS}|_{\mathcal{H}_1} f_D(x, N) + p_{TS}|_{\mathcal{H}_1} f_D(x, D_1) \tag{7.56}$$

Incorporating the *pmf* of D_1 (derived in (7.5)) in (7.56) yields (7.40). □

□

1. *False alarm probability:* Given the *pdf* of the decision statistic in (7.39), we can compute the false alarm probability. Under $\mathcal{H}_0$, the PU is in free state at the end of the sensing interval, but the decision statistic is erroneously above the threshold τ and the PU signal is declared present. In order to define the probability of false-alarm P_{fa} in our case, the following Corollary of Theorem 2 holds.

Corollary 4. The false-alarm probability of the RLRT test under unknown PU traffic and complex signal space scenario is:

$$P_{fa} = P(T_{RLRT}|\mathcal{H}_0 \geq \tau) \equiv \int_{\tau}^{+\infty} f_{T_{RLRT}|\mathcal{H}_0}(x)dx \tag{7.57}$$

2. *Detection probability:* Given the *pdf* of the decision statistic in (7.40), we can compute the detection probability. Under $\mathcal{H}_1$, the PU is in busy state at the end of the sensing interval. Under this scenario, if the decision statistic is above the threshold, the PU signal is declared present. The following Corollary of Theorem 2 holds for defining the probability of detection P_d.

   Corollary 5. The detection probability of the RLRT test under unknown PU traffic and the complex signal space scenario is:

$$P_d = P(T_{RLRT}|\mathcal{H}_1 \geq \tau) \equiv \int_{\tau}^{+\infty} f_{T_{RLRT}|\mathcal{H}_1}(x)dx. \tag{7.58}$$

7.6 Numerical results for RLRT

In this section, the effect of PU traffic on the RLRT detection method is analyzed based on the traffic model developed in Sec. 7.2.

In Fig. 7.6 and 7.7, the sensing performance of RLRT under unknown PU traffic is compared with the ideal RLRT performance. It can be well understood that the conventional model with perfect synchronization of the PU-SU sensing slots performs better than the one with unknown PU traffic. In addition, the accuracy of the derived analytical expressions of P_{fa} and P_d are confirmed where the theoretical formulae are compared against the numerical results obtained by Monte-Carlo simulations. The perfect match of the theoretical and numerical curves validates the derived analytical expressions. The Receiver Operating Characteristics (ROC) curve performance of RLRT in the considered PU traffic model for different PU traffic parameters is presented in Fig. 7.6. The sensing performance degrades significantly when the mean lengths of busy and free periods are comparable with the length of the sensing interval or in a few multiples of it. However, an improvement in the sensing performance can be seen if the length of the mean parameters M_f and M_b is increased. We present in Fig. 7.7, the misdetection probability (P_{md}) as a function of the SNR. It can be seen that, for a given PU traffic parameters, increasing the SNR improves the sensing performance for certain lower range of SNR. However, in contrast to RLRT sensing performance under known PU traffic, the RLRT sensing performance under unknown

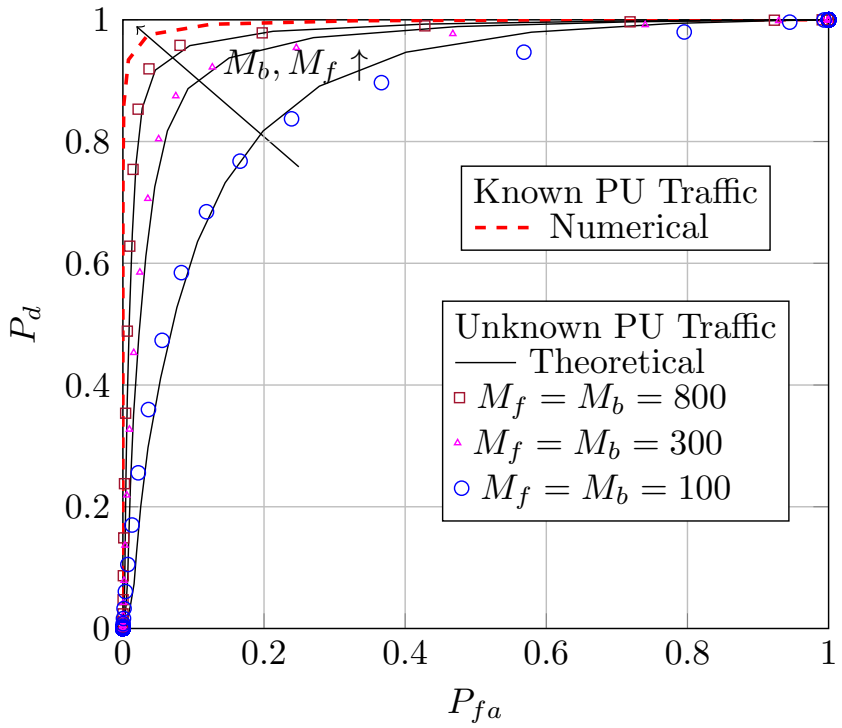


Figure 7.6: ROC curve performance, $N = 50$, $K = 4$, SNR = -6 dB.

PU traffic levels to some point ($1 > P_{md} >> 0$) above a certain SNR. This is due to the effect of the mixing of the PU signal-plus-noise and only-noise samples in the TS sensing slot.

In Fig. 7.8 and 7.9, the RLRT sensing performance is plotted as a function of sensing parameters N and K. The variation of the sensing performance of RLRT for different number of receiving antennas (K) is plotted in Fig. 7.8. It can be observed that, as in the previous case for ED, the RLRT sensing performance under unknown PU traffic is almost constant even if we increase the number of antennas. During a TS sensing slot, from each receiving antenna, the received signal samples are the mixture of pure noise samples and the samples with both noise and PU signal. Thus, even if we use multiple antennas, the nature of the received signal doesn't change much which is the reason the sensing performance improvement is suppressed by the unknown PU traffic (more specifically, the TS sensing performance) when the length of the free and busy periods of PU traffic are quite small (a few multiples of the length of the sensing window). Furthermore, we present in Fig. 7.9, the numerical simulation of detection probability (P_d) as a function of sensing window (N). Note that, unlike RLRT detection probability under known PU traffic which monotonically

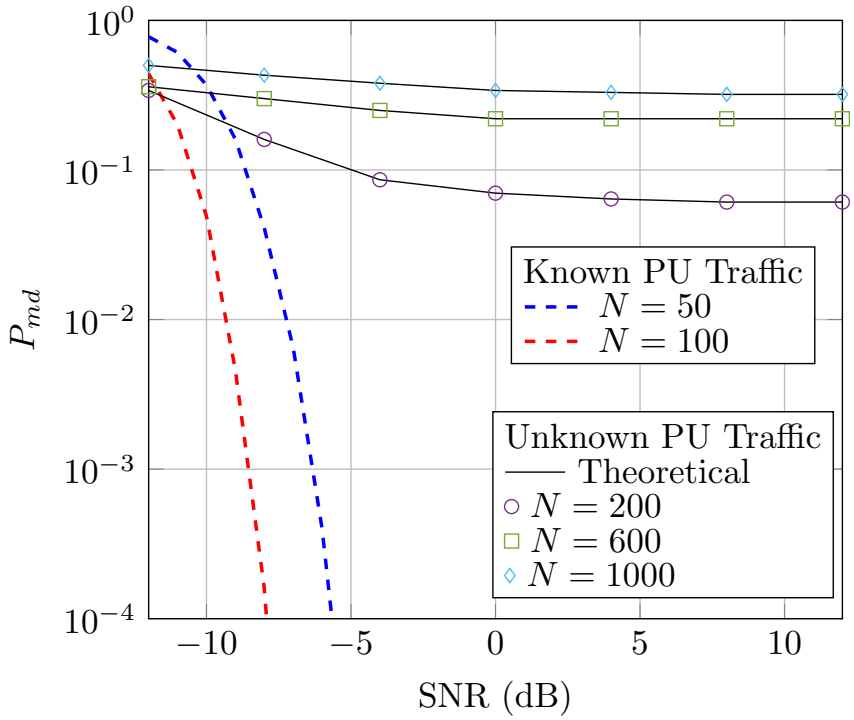


Figure 7.7: Probability of misdetection vs. SNR, $K = 8$, $P_{fa} = 0.01$, $M_f = M_b = 3000$.

increases indefinitely until '$P_d = 1$' with increasing length of sensing window, the detection probability of RLRT under unknown PU traffic do not have a monotonic property as a function of the length of the sensing window.

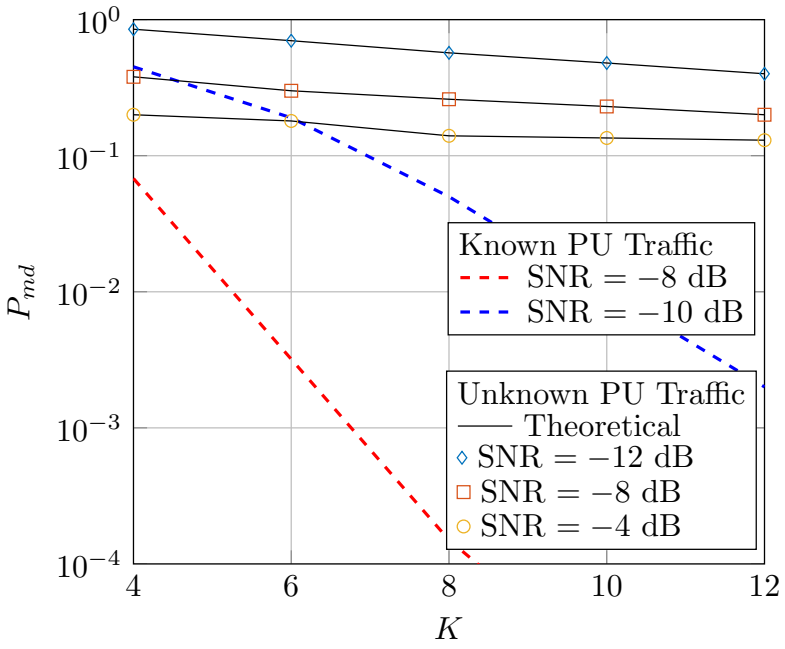


Figure 7.8: Misdetection probability vs. number of antennas K, $N = 100$, $P_{fa} = 0.01$, $M_f = M_b = 1500$.

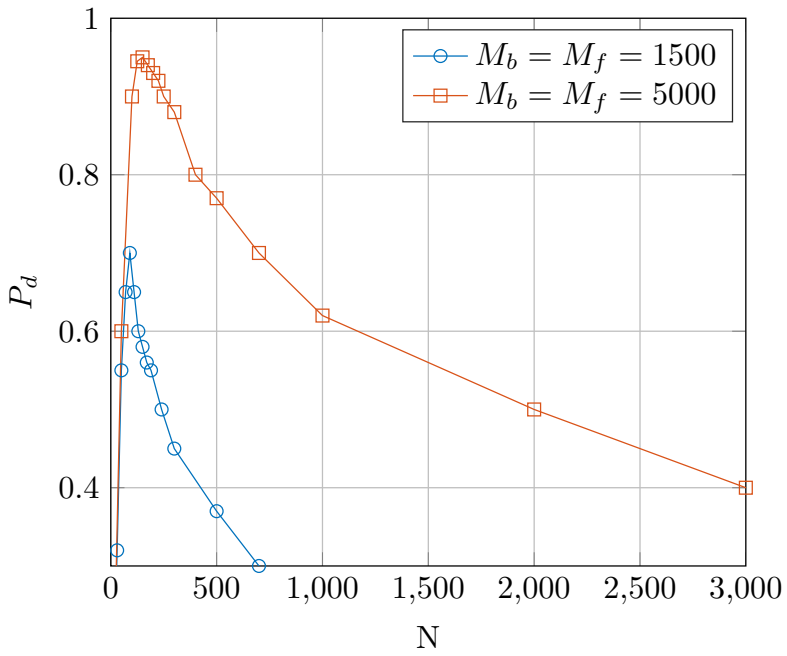


Figure 7.9: Detection probability vs. number of samples N, SNR= -10 dB, $K = 8$, $P_{fa} = 0.01$.

Part II

GNU Radio Software defined-radio (SDR) implementation

Chapter 8

SDR testbeds and GNU Radio

In this chapter, after defining the concept of *software-defined radio* (SDR), a general overview on SDR testbeds is given. The GNU Radio SDR software platform is then presented and, since the implementation of ED and EBD algorithms required the design and writing of custom applications in GNU Radio, the chapter focus is mainly on this aspect.

8.1 SDR concept

A software-defined radio system, or SDR, is a radio communication system where components that have been typically implemented in hardware (e.g., mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded computing devices. While the concept of SDR is not new, the rapidly evolving capabilities of digital electronics render practical many processes which used to be only theoretically possible.

The term "software radio" was coined in 1984 by a team at the Garland Texas Division of E-Systems, now Raytheon. It referred to a prototype digital baseband receiver equipped with an array of processors that performed adaptive filtering for interference cancellation and demodulation of broadband signals.

Joseph Mitola is referred to by many as the godfather of software radio, when he published his paper "Software Radio: Survey, Critical Analysis and Future Directions" [13] and gave the following definition:

> *"A software radio is a radio whose channel modulation waveforms are defined in software. That is, waveforms are generated as sampled digital signals, converted from digital to analog via a wide band Digital to Analog (D/A) converter and then possibly up converted from IF to RF. The receiver, similarly, employs a wide band Analog to Digital (A/D) converter that captures all of the channels of the software radio node. The receiver*

then extracts, down converts and demodulates the channel waveform using software on a general purpose processor" [13].

Mitola is recognized as having coined the term "software radio", despite E-Systems prior use. The E-Systems prototype was a receiver only and therefore not a complete radio.

In the ideal concept of SDR shown in Fig 8.1, analog-to-digital (A/D) and digital-to-analog (D/A) conversions would be performed at the Radio Frequency (RF) section. In the receiver scheme, a digital signal processor (DSP) or General Purpose Processor (GPP) would read the A/D coverter and the software transform the stream of data to any other form the application requires; in the transmitter scheme a DSP or GPP would generate a stream of numbers and these would be sent to the D/A converter connected to a radio antenna.

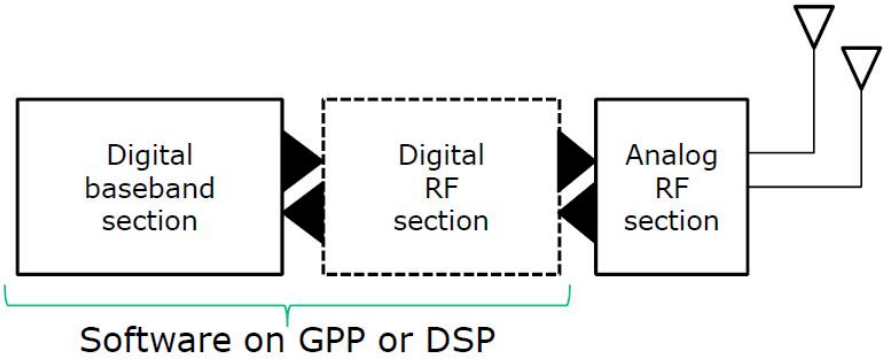


Figure 8.1: Ideal SDR concept.

The ideal scheme is not completely realizable due to the actual limits of the technology. The main problem in both directions is the difficulty of conversion between the digital and the analog domains at a high enough rate and a high enough accuracy at the same time, and without relying upon physical processes like interference and electromagnetic resonance for assistance. In the typical SDR scheme, receivers use a variable-frequency oscillator, mixer, and filter to tune the desired signal to a common intermediate frequency, usually performed by analog application specific integrated circuits (ASICs), and it is then sampled by the analog-to-digital converter. A DSP or FPGA performs digital down-conversion for GPP or host application.

A software-defined radio can be flexible enough to avoid the "limited spectrum" assumptions of designers of previous kinds of radios, in fact SDR and cognitive radio system technologies are expected to provide additional flexibility and offer improved efficiency to overall spectrum use. These technologies can be combined or deployed independently, and can be implemented in systems of any radiocommunication service.

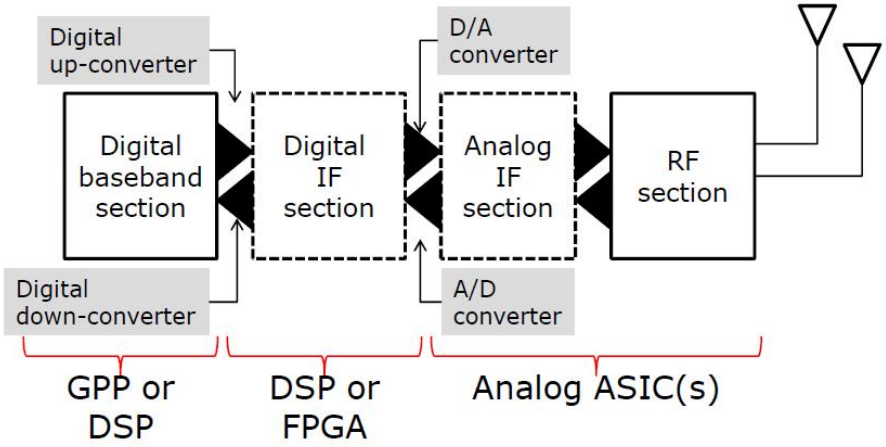


Figure 8.2: Typical SDR scheme.

The full implementation of the software-defined radio and cognitive radio systems concept is likely to be realized gradually for a number of reasons, including the current state of the technology.

8.2 SDR and testbeds

Software Defined Radio (SDR) is currently regarded as the main technology enabler supporting the implementation of experimental cognitive radio prototypes. The market is nowadays offering a vast variety of research platforms [93, 94, 95], ranging from fairly low to very high prices, involving of course a change in performance as well. Together with what can already be considered Commercial-Off-The-Shelf (COTS) hardware solutions there are also associated several software ones. Software platforms for research purposes are generally open source, and freely available.

Testbeds have been widely used in experimental research to validate new concepts and ideas stemming from theoretical research in wireless communications. Due to the broadness of research in this field, different testbeds have been developed focusing on distinct aspects of wireless communications: the physical layer, where broadband transmission and MIMO systems prevail, the MAC layer and multiple access techniques, advanced radio resource management techniques and cognitive radio.

Along with multiuser MIMO [96, 97, 98], cognitive radio is the field where most experimental research activities are being carried out through testbeds [99, 100, 101, 102, 103]. Here, the main focus is the dynamic use of spectrum by networks of systems coordinated by cognitive entities which reside on the network nodes.

Primary users must be accounted for in the testbed by allocating resources for their emulation. Moreover, spectrum sensing is needed for the detection of primary users. Finally, the cognitive layer, consisting of the several functional elements that enable increased adaptation through learning, must be included. Each of these issues has been faced and different solutions have been proposed, leading to testbeds with different, often complementary characteristics. Starting from the available testbeds, the Cognitive Radio Experimentation World (CREW) testbed [104] has been developed by federating five testbeds available at different locations throughout Europe.

Several of the aforementioned activities are being developed using a handful of hardware radio front-ends and of software radio platforms and tools, which are available for free in some cases.

8.2.1 SDR peripherals

Among the hardware front-ends, the widely used Universal Software Radio Peripheral (USRP) [93] has to be mentioned as one of the most widely used platforms. The USRP product family includes a variety of models that use a similar architecture. A motherboard provides the following subsystems: clock generation and synchronization, FPGA, A/D and D/A converters, host processor interface, and power regulation. These are the basic components that are required for baseband processing of signals. A modular front-end, called a daughterboard, is used for analog operations such as up/down-conversion, filtering, and other signal conditioning. This modularity permits the USRP to serve applications that operate between DC and 6 GHz.

In stock configuration the FPGA performs several DSP operations, which ultimately provide translation from real signals in the analog domain to lower-rate, complex, baseband signals in the digital domain. In most use-cases, these complex samples are transferred to/from applications running on a host processor, which perform DSP operations. The code for the FPGA is open-source and can be modified to allow high-speed, low-latency operations to occur in the FPGA.

Its mate software toolkit is the well-known GNU Radio software toolkit [14].

In systems developed using the GNU Radio toolkit, the digital baseband is fully implemented in software. In typical configurations, the software-defined digital baseband delivers the time-discrete complex envelope of the modulated signal to the front-end, which only performs the necessary interpolation/decimation and filtering needed to reach the D/A or A/D converter ports. The final frequency conversion to/from RF and amplification are performed in the analog front-end section.

Although widely used, however, the USRP lacks some essential features for the implementation of full-duplex real-time transceivers, like a low latency in the communication between the SDR process running in the workstation and the front-end, which makes very hard to implement Time Division Multiple Access (TDMA)

and Code Division Multiple Access (CSMA) transceivers without modifying the device hardware. As far as synchronization capabilities are concerned, some of the USRP products feature characteristics that enable synchronization across multiple devices to perform MIMO processing, but it becomes somehow expensive and cumbersome to build MIMO systems starting from several of these devices.

Several projects have been developed to overcome these limitations. Most of these address multi-antenna systems by accommodating several RF front-ends, like the WARP software radio platform [105], or try to reduce the latency by using parallel bus (e.g., PCIe) connections between the workstation and the front-end [106]. Others conform to a processing model wherein the baseband is to be implemented in on-board processing resources that reside on the front-end unit itself [107].

As for the proposed signal processing model, some of these platforms fully rely on software, leaving to hardware (typically, a FPGA) only the basic interpolation, decimation and high-rate filtering operations needed to cope with the high A/D and D/A sampling rates. In other cases, powerful FPGAs or DSPs are made available on the device and can be used to perform the most computationally intensive tasks reliving the CPUs from heavy processing loads [107].

8.3 GNU Radio

GNU Radio [14] is a free and open-source software development toolkit for the implementation of Software-Defined Radio (SDR) transceivers. GNU Radio was firstly conceived to be installed and used under GNU/Linux, but today it has become a truly cross-platform project, since it can be executed on Microsoft Windows, on the Mac OS X operating system, and on several Linux distributions, even those running on less common processor architectures like those based on the ARM processors. The following discussion will stick to a GNU Radio environment installed on a GNU/Linux system, the used GNU Radio version is 3.7.8.

The GNU Radio framework is aimed at the definition and handling of complex signal processing structures through the use of custom or predefined blocks available in the GNU Radio library. GNU Radio can be used to perform simulations as well, without using any RF device. Although it is aimed at real-time implementations, it lacks some important features of performance assessment. As Software-Defined Radio framework, when a front-end RF device such as the Universal Software Radio Peripheral (USRP) is connected to a host computer, it can be used to implement a full transceiver.

The GNU Radio environment provides an underlying layer of C++ classes describing the flow graph, the scheduler, the buffers and default blocks. On top of this, the Simple Wrapper and Interface Generator (SWIG) [108] tool provides a wrapping of C++ classes and makes them accessible by a set of higher-layer modules written in

Python [109]. Finally, the Python modules are connected to form a flow graph that defines the transceiver signal processing subsystem. The flow graph instantiation and execution are performed through a high-level process written in Python or directly in C++. A graphical application called GNU Radio Companion (GRC) provides an intuitive graphical user interface for the description of flow graphs. GRC uses the eXtensible Markup Language (XML) to describe both the flow graph and the interface of each of the signal processing modules. The low-level signal processing code of each block is written in C++ for maximum efficiency. After installing GNU Radio by following the instructions given on the web site, the first step is to run a simple simulation by using GNU Radio Companion. GRC enables the definition of GNU Radio flow graphs and automatically generates the corresponding Python code. For example you can type in your shell :

```
$ gnuradio-companion GNURadio_Installation_Path \
/share/gnuradio/examples/grc/simple/variable_config.grc
```

This command opens a GRC interface with a little example that allows the user to modify the parameters of a wave signal generated by the GNU Radio signal generator block.

8.3.1 How to create a custom block

Once the use and connection of library blocks is well understood, we can start to create our own block. Please note that this procedure applies to all GNU Radio 3.7.x versions. For the creation of a standard block [110, 111], which follows the GNU Radio structure, the utility *gr_ modtool* can be used. *gr_ modtool* is a script created by Martin Braun, that creates a source tree containing the basic structure of a GNU Radio block, including the files needed to compile it and install it (Makefiles and the configuration files), which will be necessarily modified according to the project under development. *gr_ modtool* is available in the GNU Radio source tree from version 3.6.4, and it is installed by default.

The procedure described hereafter leads to the creation of a new GNU Radio signal processing block, its Python and XML interface description through *gr_ modtool*: from the directory where the source tree of the new block should be created, type the following command:

```
$ gr_modtool newmod mymodule
```

Now, a new module structure under subdirectory *gr-mymodule* has been created in the current directory. It contains the following 8 subdirectories:

- *apps*

- *docs*
- *grc*
- *cmake*
- *include*
- *lib*
- *python*
- *swig*

and the *CMakeLists.txt* file.

At this point a block can be added using the ***gr_ modtool add*** command and answering the questions on the command line. Several block types are available in GNU Radio: sink, source, general, interpolator, decimator and hierarchical. When a new block is created, the type must be indicated. A source block has one or more outputs and no inputs. Vice versa, a sink block consumes all the data coming through its input port(s), while generating no output data. A general block has both input and output ports that are connected to other blocks. A flow graph must contain at least one source block and at least one sink block.

Multiple blocks can be added to each module executing ***gr_ modtool add*** repeatedly, however, in our example we will consider the simple case of a module with a single generic block.

Before compiling, we must at least specify our input and output data types. In the *lib* folder there are the C++ implementation source code and the header file. The *include* folder contains the public header file, while in the *swig* folder, we find the wrapper code that allows us to call our C++ functions and methods from Python code. A single wrapper code file is automatically generated during compilation based on all the public header files of the module. The *GRC* folder contains the XML files that describe the interface of each block. These files are read by *gnuradio-companion* to extract the relevant information needed to correctly represent and manage these blocks in the GUI. More in detail, 4 files must be edited in order to define a new block. Let us assume that the name of our new block is *Example_ block*:

- *gr-mymodule/lib/Example_ block_ impl.cc*
- *gr-mymodule/lib/Example_ block_ impl.h*
- *gr-mymodule/include/mymodule/Example_ block.h*
- *gr-mymodule/grc/mymodule_ Example_ block.xml*

Once we have created our generic block, we must appropriately change the inputs and outputs according to the desired behaviour. In the C++ file, the *io_signature*s defining the input/output data type and size must be modified.

```
Example_block_impl::Example_block_impl()
: gr::block("Example_block",
  gr::io_signature::make(<+MIN_IN+>, <+MAX_IN+>, sizeof (<+float+>)),
  gr::io_signature::make(<+MIN_IN+>, <+MAX_IN+>, sizeof (<+float+>)))
  {}
```

The first call to `gr_make_io_signature` specifies the input signature, where the MIN_IN e MAX_IN parameters indicate, respectively, the minimum and maximum number of input ports. The second call to `gr_make_io_signature` specifies the output signature with the same syntax. Similarly, the XML file given in Listing 8.1 in the *GRC* subdirectory must be coherently modified with the specification given through the calls to `gr_make_io_signature`.

```
<?xml version="1.0"?>
<block>
  <name>Example_block</name>
  <key>newblock_Example_block</key>
  <category>newblock</category>
  <import>import newblock</import>
  <make>newblock.Example_block()</make>
  <!-- Make one 'param' node for every Parameter
       Sub-nodes:
       * name
       * key (makes the value accessible as $keyname)
       * type -->
  <param>
    <name>...</name>
    <key>...</key>
    <type>...</type>
  </param>

  <!-- Make one 'sink' node per input. Sub-nodes:
       * name (an identifier for the GUI)
       * type
       * vlen
       * optional (set to 1 for optional inputs) -->
  <sink>
    <name>in</name>
    <type><!-- e.g., int, complex, vector..--></type>
  </sink>

```

```
29   <!-- Make one 'source' node per output. Sub-nodes:
30        * name (an identifier for the GUI)
31        * type
32        * vlen
33        * optional (set to 1 for optional inputs) -->
34   <source>
35     <name>out</name>
36     <type><!-- e.g., int, complex, vector..--></type>
37   </source>
38 </block>
```

Listing 8.1: Example of XML GRC block configuration.

After the block interfaces have been specified both in the C++ and in the XML code, we are ready to write the signal processing code. This code can be split in three parts:

- resource allocation and initialization;
- runtime signal processing;
- resource deallocation.

Resource allocation and initialization code belong to the constructor of the newly created class. It typically contains memory allocation and initialization of data structures by preliminary computation of data. Other resources may be allocated as well, such as file handlers or other operating system facilities. This code is executed once at the time of instantiation of the block.

Resource deallocation code belongs to the destructor of the newly created class. It typically contains memory deallocation and deallocation of other resources like, e.g., closure of file handlers or of other operating system facilities.. Again, this code is executed once at the time of block destruction.

The runtime processing code, instead, is executed multiple times throughout the life of each block and determines the runtime performance of the whole system. It belongs to the `general_work` function of our *Example_block* class. Two arguments of the `general_work` functions are *noutput_items* and *ninput_items*. As shown in Fig. 8.3, *ninput_items* is a vector and indicates the total number of available items in each input buffer, while `noutput_items` indicates the total number of output items that can be written to each output buffer. `noutput_items` is an integer and not a vector, since in GNU Radio all output buffers have the same size. *input_items* (*output_items*) is a vector of input (output) buffers, in which each element corresponds to an input (output) port. An item in GNU Radio can be anything that can be digitally represented: real samples, complex samples, integer types, and most importantly GNU Radio allows for an item both scalar types and

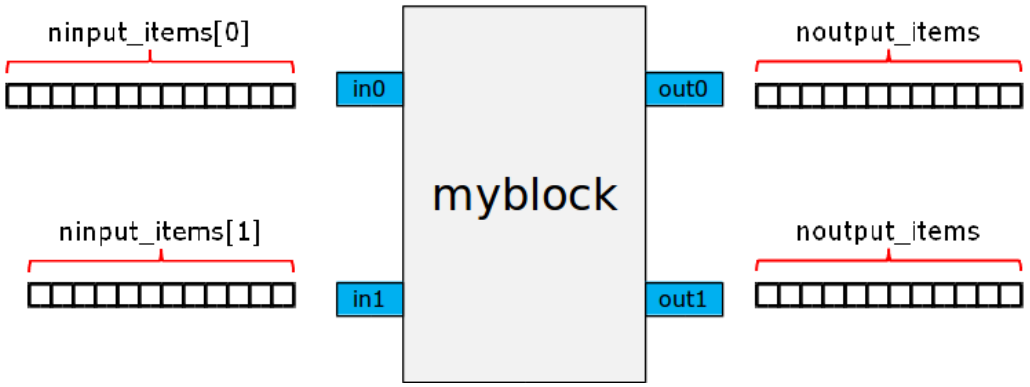


Figure 8.3: Example of a block with two inputs and two outputs.

vectors. When the items of a block are scalar types, items are processed sample per sample like a stream (e.g., as in a filter), instead when we work with vectors, it means that the block needs a certain amount of samples before starting to process them, a typical example of a vector block is the FFT block.

```
1  int Example_block_impl::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          ...
9      }
```

Listing 8.2: General work example.

In the simpler case of synchronous blocks, a simplified version of runtime processing function called `work` is defined in place of the `general_work`. It has a reduced set of input arguments: the `ninput_items` vector is not provided, since it is assumed that synchronous blocks feature the same buffer sizes both for inputs and outputs.

After compiling and installing, the block can be connected to other blocks. The blocks are connected together to form a flow graph. In order to connect the blocks together and create the flow graph, several options are available: C++, Python, or the graphical tool GRC. The latter is the easiest way, because a flow graph can be created in a few simple steps. This tool is also able to generate the corresponding Python code.

In Python, you can connect the source and the destination with the `connect` method.

```
1  self.connect (src0, (dst, 0))
```

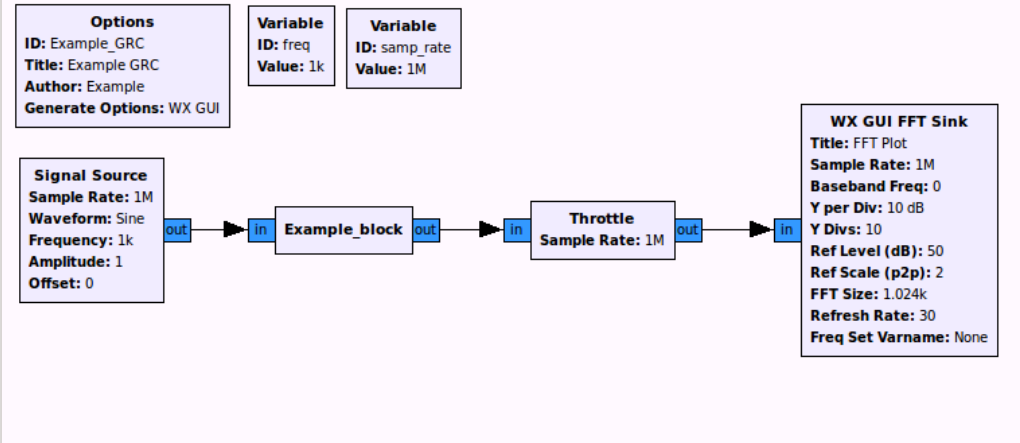


Figure 8.4: Example of connected blocks in GRC.

```
2 self.connect (src1, (dst, 1))
```

Listing 8.3: Example of block connection in Python.

In Listing 8.4 you can see the Python code generated by the GRC file shown in Fig. 8.4.

```
1  self.Add(self.wxgui_fftsink2_0.win)
2  self.newblock_Example_block = newblock.Example_block()
3  self.gr_throttle = gr.throttle(gr.sizeof_gr_complex*1, samp_rate)
4  self.gr_sig_source = gr.sig_source_c(samp_rate, gr.GR_SIN_WAVE, freq, 1, 0)
5  ##################################################
6  # Connections
7  ##################################################
8  self.connect((self.gr_sig_source_x, 0), (self.newblock_Example_block, 0))
9  self.connect((self.newblock_Example_block, 0), (self.gr_throttle, 0))
10 self.connect((self.gr_throttle, 0), (self.wxgui_fftsink2, 0))
```

Listing 8.4: Example of block connections in Python code generated by GRC.

After the Python code has been generated, it is possible to execute the flow graph. This is the point where a hidden entity, the GNU Radio scheduler, comes into play. It is responsible for the allocation of processing resources across the blocks within a flow graph. There are two types of scheduler: single-threaded scheduler (STS) and thread-per-block (TPB) scheduler; each of them can be selected by setting the environment variable `GR_SCHEDULER` respectively to `STS` or `TPB`. When the single-threaded scheduler is selected, it allocates only one thread for each flow graph so that the runtime signal processing functions are executed sequentially rather than in parallel.

As far as the scheduler is concerned, a GNU Radio flow graph can be in one of the four following different states: *start*, *run*, *stop* and *wait*.

When the flow graph is started through the `run` or the `start` function (the former has a blocking behaviour, while the latter does not), the scheduler calls each block executor and polls periodically all of them to verify which one has enough input items and an available output buffer. Every time this occurs, the corresponding block starts its runtime processing. Iterated polling follows in the same order from source to sink until a `wait` or a `stop` method is called. The `wait` call allows a flow graph to be reconfigured so that blocks and/or connections between them can be changed. After a non-blocking `start`, the `stop` method forces the end of the execution. In the thread-per-block scheduler mode, each block thread continuously cycles over the `work` method that defines the runtime signal processing function.

Chapter 9

Parallel Energy Detector implementation

In this chapter, the implementation of the parallel Energy Detector is presented. The main goal of the project, realized in collaboration with CSP - ICT Innovation [112], was the construction of a spectrum occupancy dataset for Digital Mobile Radio (DMR) [113] channels, i.e., the collection of occupancy percentages of each 12.5 KHz DMR channel from 136 MHz to 174 MHz.

The radio peripheral used for this testbed is the Ettus Research USRP1. Occupancy statistics have been computed through a GNU Radio flowgraph, in which 3 custom blocks have been added to the existing ones. The name of the created module for the custom blocks is *spectrum_sensing*. Finally, results are sent through UDP packets to an external Graphical User Interface (GUI) developed by CSP - ICT Innovation.

9.1 Digital Mobile Radio (DMR)

Digital Mobile Radio (DMR) [113] is an open digital radio standard defined in the European Telecommunications Standards Institute (ETSI) Standard TS 102 361 parts 1–4, which sets out a digital radio specification for professional, commercial and private radio users. In practice, DMR manufacturers have focused on building products for the professional and commercial markets for both licensed conventional mode operation (known as DMR Tier II) and licensed trunked mode operation (known as DMR Tier III).

The DMR standard operates within the existing 12.5 kHz channel spacing used in land mobile frequency bands globally, but achieves two voice channels through two-slot TDMA technology built around a 30 ms structure. The TDMA implementation in DMR offers a spectrum-efficiency of 6.25 kHz per channel, the speech coding

standard is the proprietary AMBE+2 vocoder.

The modulation is 4-state FSK, which creates four possible symbols over the air at a rate of 4,800 symbols/s, corresponding to 9,600 b/s. After overhead, forward error correction, and splitting into two channels, there is 2,450 b/s left for a single voice channel using DMR, compared to 64,000 b/s with traditional telephone circuits.

The standards that define DMR consist of four documents:

- TS 102 361-1: the DMR air interface protocol.
- TS 102 361-2: the DMR voice and generic services and facilities.
- TS 102 361-3: the DMR data protocol
- TS 102 361-4: the DMR trunking protocol.

DMR covers the RF range 30 MHz to 1 GHz.

9.2 USRP1

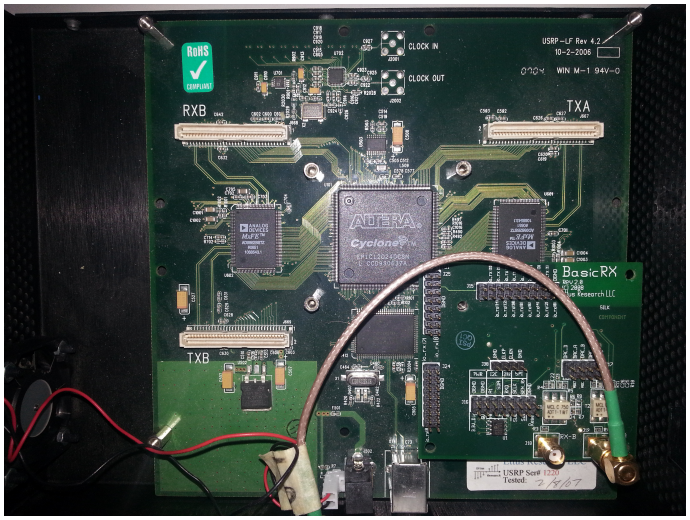


Figure 9.1: USRP1 with Basic RX daughterboard used for the testbed.

The USRP1 provides an entry-level platform and a modular design allowing the hardware to operate from DC to 6 GHz. The architecture includes an Altera Cyclone FPGA, 64 MS/s dual A/D converter, 128 MS/s dual D/A converter and USB 2.0 connectivity to provide data-to-host processors. The USRP1 can stream from 250

kS/s up to 16 MS/s to host applications. On-board digital down and up converters provide 15 mHz of tuning resolution and adjustable sample rates [114]

The USRP1 includes connectivity for two daughtercards, enabling two complete transmit/receive chains. In our case, we used only the receive chain by connecting a Basic RX daughterboard, which provides receiving capability to the USRP from 1 to 250 MHz. Fig. 9.1 shows the USRP1 with the Basic RX daughterboard used for the testbed, while Fig. 9.2 shows the schematic of USRP1 [114].

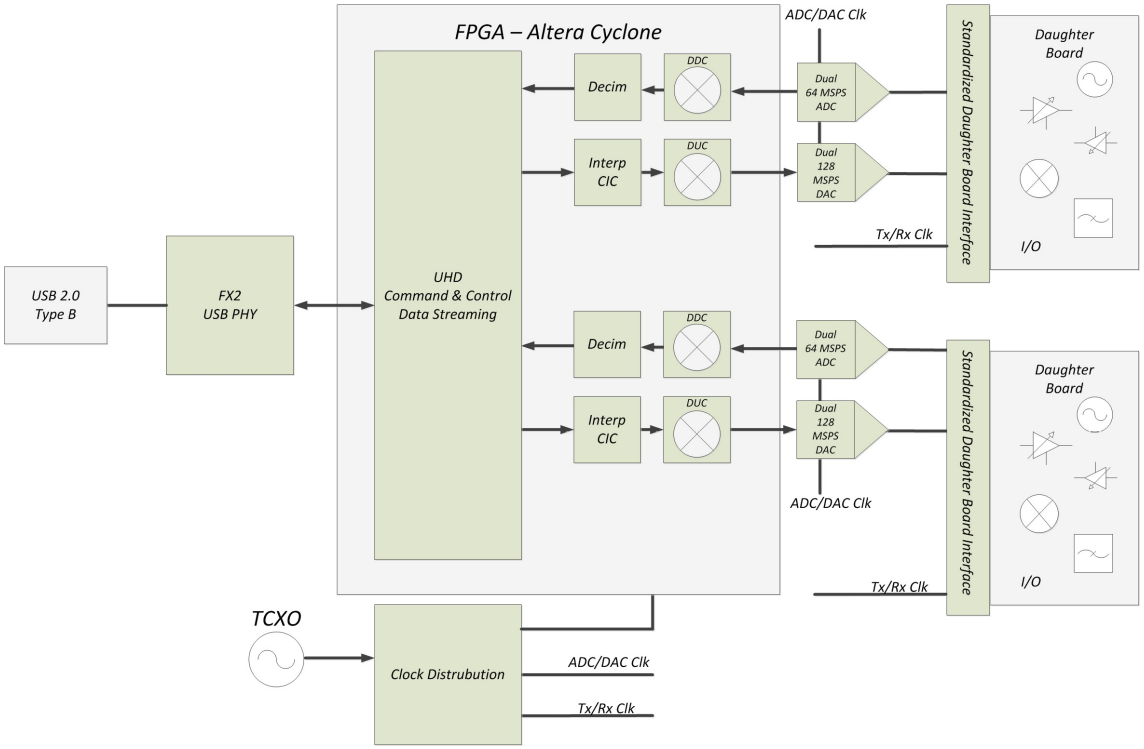


Figure 9.2: USRP1 schematic.

9.3 GNU Radio flowgraph

The idea of the parallel Energy Detector originated by the two following constraints:

1. the minimum sample rate of USRP1 is 250 kS/s;
2. channel spacing of DMR is 12.5 kHz.

The sensing of a single DMR channel would have required decimation by a factor 20 and, most of all, a huge waste of sampling and computational capabilities of the USRP1. Fig. 9.3 shows the general scheme of the whole process.

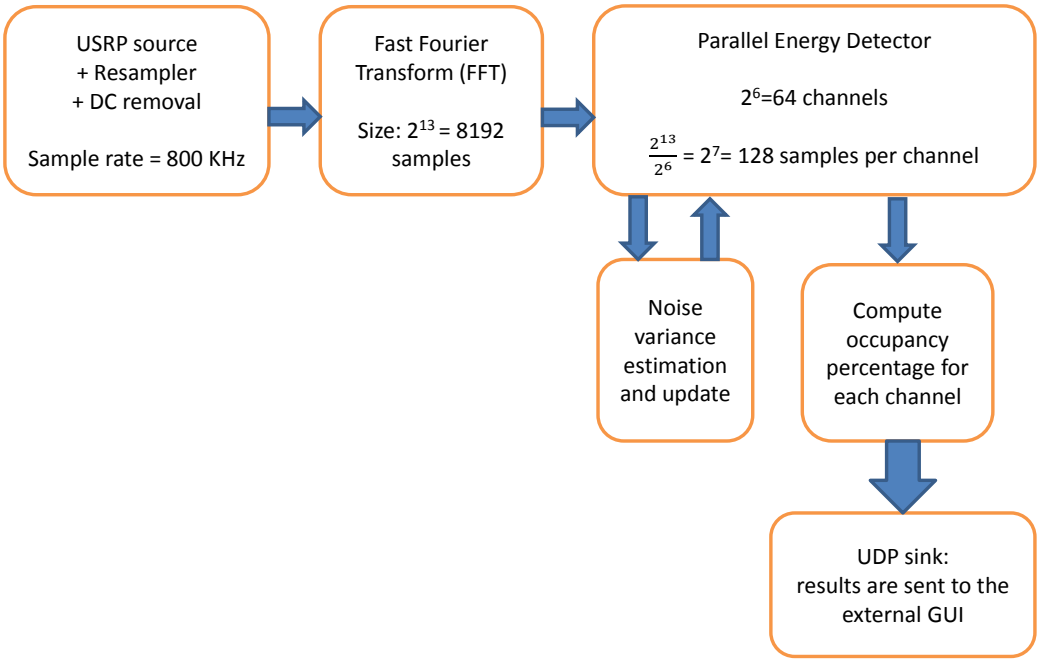


Figure 9.3: Parallel Energy Detector diagram.

After receiving the I/Q samples through the USRP1, a Fast Fourier Transform (FFT) is performed on every $2^{13} = 8192$ samples; the Energy Detector block demultiplexes the FFT samples into $2^6 = 64$ DMR channels and computes ED test statistic on each channel by using $\frac{2^{13}}{2^6} = 2^7 = 128$ samples per channel; the noise variance is estimated and updated according to the hybrid approach HED2 (online noise estimation) shown in Sec. 4.1.1; occupation percentages are then computed through consecutive binary decisions and finally the results are sent to the external GUI.

Fig. 9.4 shows the GNU Radio flowgraph and each operation is described in detail as follows:

- **Signal acquisition and resampling**: GNU Radio allows to acquire samples from the USRP through the *UHD: USRP Source* block. The parameters to be set are the central frequency, which is set in the transition band between 2 DMR channels, and the sample rate set to 1 MHz. In order to obtain N parallel DMR channel, with N a power of 2, we resample our signal to 800 kHz with the *Rational Resampler* block by a decimation factor equal to 5 and interpolation factor to 4, in kHz $1000 \cdot \frac{4}{5} = 800$.

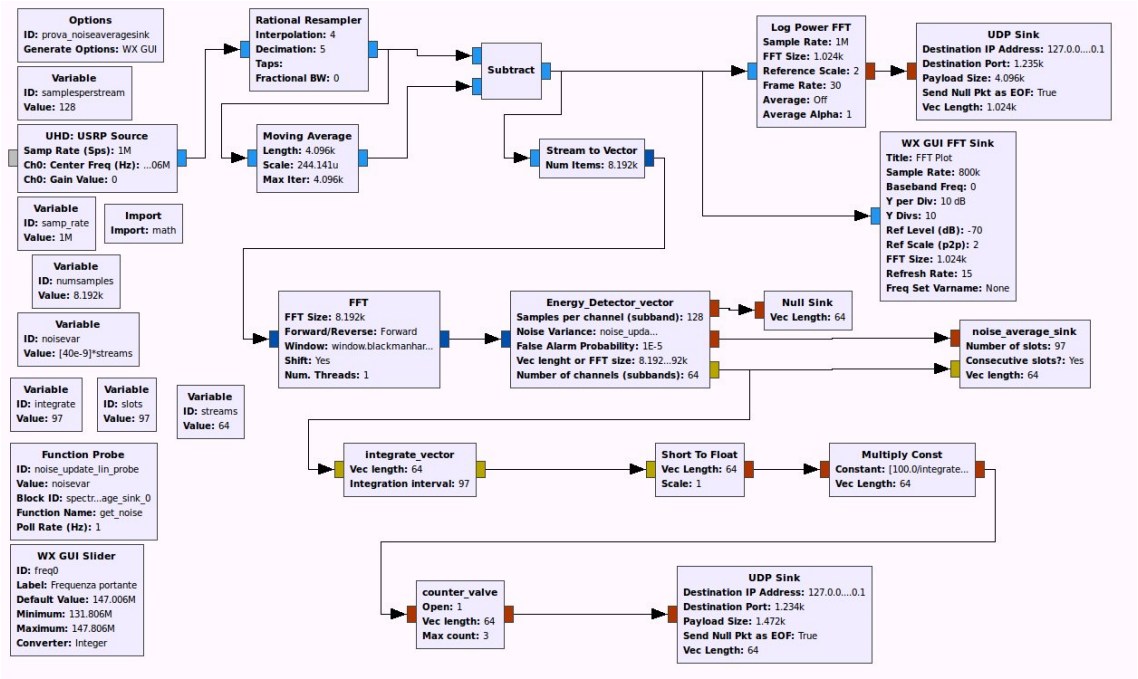


Figure 9.4: GNU Radio flowgraph for parallel Energy Detector.

- **DC component removal**: Any mixed-signal system like a SDR-based receiver has spurs and this translates into a spike at zero frequency of about 10 dB above the noise floor. In order to remove this component, we can compute the moving average of the signal and subtract it from the signal itself. This is accomplished with the *Moving Average* and *Subtract* block.

- **Fast Fourier Transform**: Before converting our signal from time to frequency domain, we need to pack our samples into vectors of 8192 samples, this is accomplished through the *Stream to Vector* block. Then, the FFT is computed on every 8192 samples in the *FFT* block.

- **Energy Detection**: The *Energy_Detector_vector* custom block computes for each channel the test statistic and compares it against a predefined threshold. Implementation details are described in Sec. 9.4.

- **Noise estimation and update**: Different blocks have been utilized to implement the hybrid approach 2 mechanism described in Sec. 4.1.2, among which the *noise_average_sink* custom block. Details are described in Sec. 9.5.

- **Computation of occupancy percentages**: The *integrate_vector* custom block is very similar to the exiting GNU Radio *Integrate* block, but it works

on the received vectors of binary decisions and sums all 0s or 1s with the same vector index (same channel). The 2 parameters are the vector length and the integration interval, which tells the block how many vectors it will receive before producing one output. The output vector is sent to the *Short To Float* block and then to the *Multiply Const* block where the averaging operation and percentage computation are completed by multiplying all the elements of the vector by 100 and dividing them by the integration interval.

- **Carrier frequency change and buffer cleaning**: The *WX GUI Slider* block allows to change the carrier frequency. This is done runtime by GNU Radio: the advantage is that the flowgraph is not stopped and there is no time waste for USRP re-calibration, however, with a continuous flow, the initial results on the new central frequency might be affected by the buffers of the previous central frequency. The *counter_valve* custom block is somehow similar to the existing GNU Radio *Valve* block: when the valve is opened, items are just copied to the output buffer; however, in this case, when the frequency is changed, the valve closes. This can be implemented by a slight modification of the Python code of the flowgraph, in which the public set method `set_open` of the *counter_valve* block is set to 'False' whenever the central frequency of the USRP is changed by the slider:

```
1 def set_freq0(self, freq0):
2                     self.freq0 = freq0
3                     self.uhd_usrp_source_0.set_center_freq(self.freq0, 0)
4                     self._freq0_slider.set_value(self.freq0)
5                     self.spectrum_sensing_counter_valve_0.set_open(False)
```

Listing 9.1: Carrier frequency setter method in Python.

where `set_freq0` is the method of the *WX GUI Slider* block that changes the central frequency. The *Max count* parameter of the custom block tells the valve that it will have to wait *Max count* input items before re-opening it, and thus allowing to clean the buffers of all the other blocks.

- **Flowgraph outputs**: The spectrum can be visualized in GRC through the *WX GUI FFT Sink* block, which uses the wxWidgets libraries [115] (as the *WX GUI Slider* block). The occupancy statistics and the magnitude of the frequency spectrum (computed through the *Log Power FFT* block) are sent to the external GUI as UDP packets through two *UDP Sink* blocks.

9.4 Energy detection

The *Energy_ Detector_ vector* block has the following parameters that can be set by the user:

- *Samples per channel*: 128.
- *Noise variance*: a vector of noise variance estimates with size equal to the number of channels.
- *False alarm probability*: different values of P_{fa} can be selected from 10^{-5} to 0.5.
- *FFT size*: 8192.
- *Number of channels*: 64.

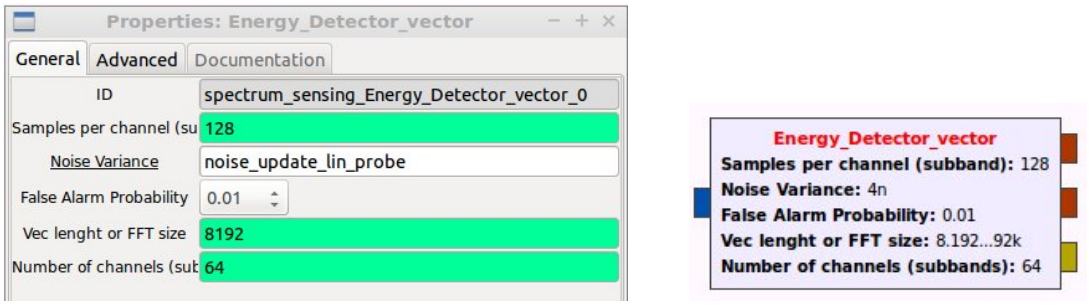


Figure 9.5: Parallel Energy Detector block with parameters.

The following test statistic for each channel has been computed:

$$T_{ED} = \frac{1}{N \cdot \sigma_v^2 \cdot \mathrm{FFT}_{\mathrm{size}}} \sum_{k=1}^{N} |Y_k|^2 \tag{9.1}$$

where N represents the samples per channel, Y_k a FFT sample. In order to have the same statistical result of (2.14), the normalization by the factor $1/\mathrm{FFT}_{\mathrm{size}}$ has been applied according to the Parseval theorem:

$$\sum_{n=0}^{N-1} |x_n|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X_k|^2 \tag{9.2}$$

Starting from (2.36), the threshold of ED can be expressed as:

$$t_{ED} = \frac{Q^{-1}(P_{fa})}{\sqrt{N}} + 1 \tag{9.3}$$

Since we use a very large number of slots to estimate the noise variance, the distributions of the P_{fa} expressions of HED1 (4.5) and HED2 (4.13) converge to (2.36). Many threshold values have been previously computed offline in the Matlab environment for different values of N and P_{fa} and stored into a file, which will work as a lookup table for the block.

The block has 1 input vector of size equal to the FFT and produces 3 output vectors of size equal to number of channels. The outputs are:

- `test`: a vector of floats representing the pure test statistic;
- `test_no_var`: a vector of floats representing the received energies not normalized by the noise variance;
- `bin`: a vector of short integers representing the binary decisions (0 or 1) after the comparison against the threshold;

with the following `io_signature`:

```
1 gr::io_signature::make(1, 1, fftsize * sizeof (gr_complex)),
2 gr::io_signature::make3(3, 3, streams * sizeof (float), streams * sizeof (float), streams * sizeof (
      short))),
```

The public method `set_noisevar` allows to update the noise variance estimation. Listings 9.2, 9.3 and 9.4 show respectively the public header, the implementation header and the implementation source files of the *Energy_Detector_vector* block.

```
1  #include <spectrum_sensing/api.h>
2  #include <gnuradio/block.h>
3  #include <iostream>
4  #include <stdio.h>
5
6  namespace gr {
7    namespace spectrum_sensing {
8
9      class SPECTRUM_SENSING_API Energy_Detector_vector : virtual public gr::block
10     {
11      public:
12       typedef boost::shared_ptr<Energy_Detector_vector> sptr;
13
14        static sptr make(int samplesperstream, std::vector<float> noisevar, int prob_false_alarm
              , int fftsize, int streams);
15        virtual void set_noisevar(std::vector<float> noisevar) = 0;
16     };
17   }
18 }
```

Listing 9.2: Energy_Detector_vector.h - Public header file.

```
#include <spectrum_sensing/Energy_Detector_vector.h>

namespace gr {
  namespace spectrum_sensing {

    class Energy_Detector_vector_impl : public Energy_Detector_vector
    {
     private:
        int sps;
        std::vector<float> d_noisevar;
        int d_prob_false_alarm;
        float **threshold;
        FILE *fp;
        int d_fftsize;
        int d_streams;
     public:
         Energy_Detector_vector_impl(int samplesperstream, std::vector<float> noisevar, int
             prob_false_alarm, int fftsize, int streams);
        ~Energy_Detector_vector_impl();
      void forecast (int noutput_items, gr_vector_int &ninput_items_required);
      void set_noisevar(std::vector<float> noisevar);
      int general_work(int noutput_items,
              gr_vector_int &ninput_items,
              gr_vector_const_void_star &input_items,
              gr_vector_void_star &output_items);
    };
  }
}
```

Listing 9.3: Energy_Detector_vector_impl.h - Implementation header file.

```
#include "config.h"
#include <gnuradio/io_signature.h>
#include "Energy_Detector_vector_impl.h"
#define SAMPLES 4096
#define PFA 13
#define STEP 1

using namespace std;

namespace gr {
  namespace spectrum_sensing {

    Energy_Detector_vector::sptr
    Energy_Detector_vector::make(int samplesperstream, std::vector<float> noisevar, int
        prob_false_alarm, int fftsize, int streams)
    {
      return gnuradio::get_initial_sptr
```

```
17          (new Energy_Detector_vector_impl(samplesperstream, noisevar, prob_false_alarm,
                fftsize, streams));
18      }
19
20      Energy_Detector_vector_impl::Energy_Detector_vector_impl(int samplesperstream, std::
            vector<float> noisevar, int prob_false_alarm, int fftsize, int streams)
21        : gr::block("Energy_Detector_vector",
22                    gr::io_signature::make(1, 1, fftsize * sizeof (gr_complex)),
23                    gr::io_signature::make3(3, 3, streams * sizeof (float), streams * sizeof (float),
                          streams * sizeof (short))),
24                    sps(samplesperstream), d_noisevar(noisevar), d_prob_false_alarm(
                          prob_false_alarm), d_fftsize(fftsize), d_streams(streams)
25      {
26      set_relative_rate(1.0);
27                      threshold = new float*[PFA];
28                      for(int k=0;k<PFA;k++)
29                          threshold[k] = new float [SAMPLES];
30
31                            if ((fp = fopen ("threshold.dat","rb")) == NULL)
32                            {
33                              cout << "File vuoto\n" << endl;
34                            }
35                    for (int i=0;i<PFA;i++)
36                            fread(threshold[i],sizeof(float),SAMPLES,fp);
37                        fclose(fp);
38      }
39
40      Energy_Detector_vector_impl::~Energy_Detector_vector_impl()
41      {
42        for(int k=0;k<PFA;k++)
43        delete [] threshold;
44       delete [] threshold;
45      }
46
47      void Energy_Detector_vector_impl::set_noisevar(std::vector<float> noisevar)
48     {
49          d_noisevar = noisevar;
50      }
51
52      void
53      Energy_Detector_vector_impl::forecast (int noutput_items, gr_vector_int &
            ninput_items_required)
54      {
55          ninput_items_required[0] = noutput_items;
56      }
57
58      int
59      Energy_Detector_vector_impl::general_work (int noutput_items,
60                              gr_vector_int &ninput_items,
```

```
61                          gr_vector_const_void_star &input_items,
62                          gr_vector_void_star &output_items)
63      {
64           if((sps % STEP!=0) || (sps>SAMPLES)) {
65              printf("Samples must be multiple of 10 and less than %d.\n", SAMPLES);
66              return -1;
67            }
68            if(d_streams != d_noisevar.size()) {
69              cout << "Number of streams and noise vector size DON'T match." << endl;
70              return -1;
71          }
72            if(sps*d_streams > d_fftsize) {
73              cout << "Too many samples per stream." << endl;
74              return -1;
75          }
76            gr_complex *in = (gr_complex *) input_items[0];
77            float *out0= (float *) output_items[0];
78            float *out1= (float *) output_items[1];
79            short *out2= (short *) output_items[2];
80            float T_ED[d_streams];
81            int z = 0;
82            int col = sps/STEP-1;
83            int start_index = (d_fftsize - sps*d_streams) / 2 - 1;
84            if (start_index < 0)
85            start_index = 0;
86
87            while(z < noutput_items) {
88
89                in += start_index;
90                memset(T_ED, 0, d_streams * sizeof(float));
91
92                for(int i=0; i<d_streams; i++){
93                    for(int j=i*sps; j<(i+1)*sps; j++) {
94                        T_ED[i] += abs(in[j]*conj(in[j]));
95                    }
96                    float temp = T_ED[i]/(d_fftsize*sps);
97                    out1[i] = temp;
98                    out0[i] = temp/d_noisevar[i];
99
100                   if(out0[i]>threshold[d_prob_false_alarm][col])
101                       out2[i] = 1;
102                   else
103                       out2[i] = 0;
104               }
105               in += d_fftsize;
106               out0 += d_streams;
107               out1 += d_streams;
108               out2 += d_streams;
109               z++;
```

```
110         }
111         consume_each (noutput_items);
112         return noutput_items;
113       }
114   }
115 }
```

Listing 9.4: Energy_Detector_vector_impl.cc - Implementation source file.

The XML block definition (not all parameters and inputs are listed) looks like the following:

```
1  <?xml version="1.0"?>
2  <block>
3    <name>Energy_Detector_vector</name>
4    <key>spectrum_sensing_Energy_Detector_vector</key>
5    <category>spectrum_sensing</category>
6    <import>import spectrum_sensing</import>
7    <make>spectrum_sensing.Energy_Detector_vector($samplesperstream, $noisevar, $(
         prob_false_alarm.fcn), $fftsize, $streams)</make>
8    <callback>set_noisevar($noisevar)</callback>
9
10  <param>
11     <name>Samples per channel (subband)</name>
12     <key>samplesperstream</key>
13     <value>128</value>
14     <type>int</type>
15   </param>
16
17    ....
18
19   <sink>
20     <name>in</name>
21     <type>complex</type>
22     <vlen>$fftsize</vlen>
23   </sink>
24
25   .....
26
27 <source>
28     <name>bin</name>
29     <type>short</type>
30     <vlen>$streams</vlen>
31   </source>
32 </block>
```

Listing 9.5: Sections of the XML code for the *Energy_ Detector_ vector* block.

9.5 Noise estimation and update

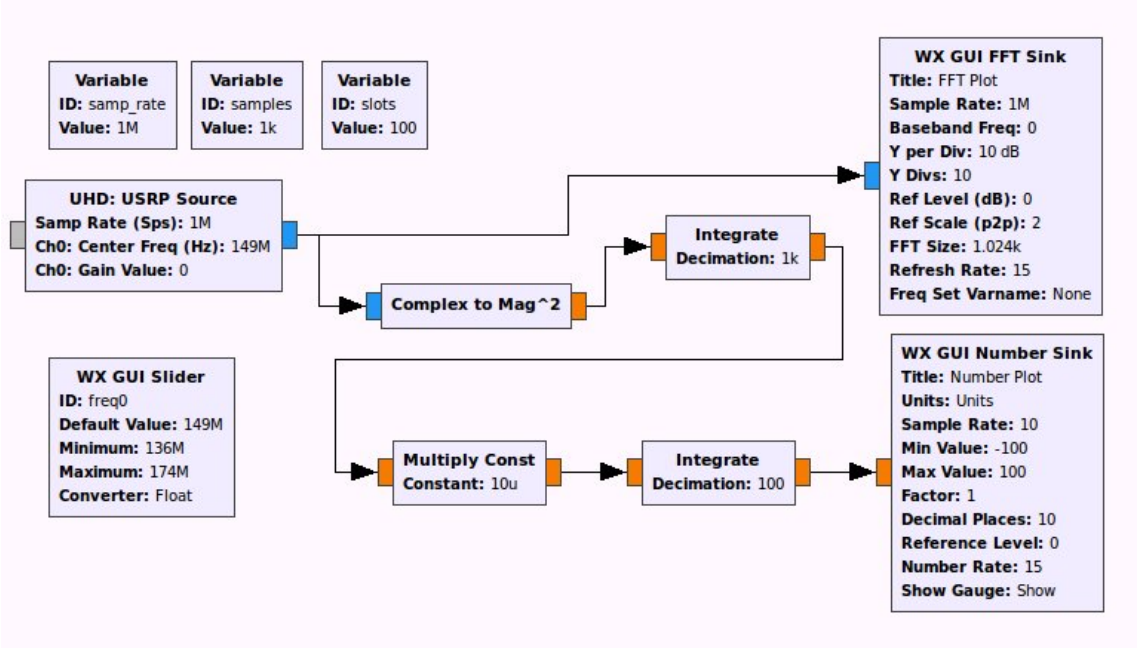


Figure 9.6: Flowgraph for initial noise estimation.

An initial noise estimation could be obtained for example with the flowgraph in Fig. 9.6. By checking the FFT GUI, if the spectrum is flat and the value provided by the *WX GUI Number Sink* is stable enough, we can use it as initial noise variance estimation and store it in the `noisevar` variable in the flowgraph of Fig. 9.4. The slider allows to change the carrier frequency in case the band of interest is in use.

GNU Radio does not allow flowgraphs with feedback. However, one way to implement asynchronous feedbacks is through threads and callback methods; these special methods are public methods of the C++ implementation and can be used in Python or GRC by other blocks of the flowgraph. The noise estimation and update mechanism can be described as follows:

1. **Noise estimation in $\mathcal{H}_0$ slots**: The *noise_ average_ sink* custom block has the following public header:

```
#include <spectrum_sensing/api.h>
#include <gnuradio/sync_block.h>
#include <iostream>
#include <stdio.h>

namespace gr {
  namespace spectrum_sensing {
    class SPECTRUM_SENSING_API noise_average_sink : virtual public gr::
        sync_block
    {
     public:
      typedef boost::shared_ptr<noise_average_sink> sptr;
       static sptr make(int slots, int consecutive, int veclen);
       virtual std::vector<float> get_noise() = 0;
    };
  }
}
```

Listing 9.6: noise_average_sink.h - Public header file.

And the following class definition in the implementation header

```
#include <spectrum_sensing/noise_average_sink.h>

namespace gr {
  namespace spectrum_sensing {
   class noise_average_sink_impl : public noise_average_sink
    {
     private:
      int *current_slots;
      int d_slots;
      int d_consecutive;
      float *average;
      int d_veclen;
      std::vector<float> noise;
     public:
      noise_average_sink_impl(int slots, int consecutive, int veclen);
      ~noise_average_sink_impl();
      std::vector<float> get_noise();
      float get_noise_subband();
      int work(int noutput_items,
          gr_vector_const_void_star &input_items,
          gr_vector_void_star &output_items);
    };
  }
}
```

Listing 9.7: noise_average_sink_impl.h - Implementation header file.

The block has 2 input vectors and does not produce any output (sink block), the first input vector represents the unnormalized test statistics of the *Energy_Detector_vector* block, the second vector the binary decisions of the same block. The sliding window mechanism illustrated in Fig. 4.2 is implemented in the `work` method of Listing 9.8. The block sums the received energy samples only when the slots are declared $\mathcal{H}_0$ up to the parameter `slots` and the noise average is locally updated; the parameter `consecutive` is a flag that allows noise estimation only when the $\mathcal{H}_0$ slots are consecutive (if true). This process is performed for each channel independently. The public method `get_noise` allows other blocks of the flowgraph to use the constantly updated vector of noise estimates.

```
1  #include "config.h"
2  #include <gnuradio/io_signature.h>
3  #include "noise_average_sink_impl.h"
4
5  using namespace std;
6
7  namespace gr {
8    namespace spectrum_sensing {
9
10     noise_average_sink::sptr
11     noise_average_sink::make(int slots, int consecutive, int veclen)
12     {
13       return gnuradio::get_initial_sptr
14         (new noise_average_sink_impl(slots, consecutive, veclen));
15     }
16     noise_average_sink_impl::noise_average_sink_impl(int slots, int consecutive, int
           veclen)
17       : gr::sync_block("noise_average_sink",
18                gr::io_signature::make2(2, 2, veclen*sizeof(float), veclen*sizeof(short)),
19                gr::io_signature::make(0, 0, 0)),
20                d_slots(slots), d_consecutive(consecutive), d_veclen(veclen)
21     {
22         noise.resize(d_veclen);
23         average = new float [d_veclen];
24         current_slots = new int [d_veclen];
25         memset(average, 0, d_veclen*sizeof(float));
26         memset(current_slots, 0, d_veclen*sizeof(int));
27         for(int i=0;i<d_veclen; i++)
28             noise[i] = 25e-9;
29     }
30
31     noise_average_sink_impl::~noise_average_sink_impl()
32     {
33         delete [] average;
34         delete [] current_slots;
```

```
35      }
36
37      std::vector<float> noise_average_sink_impl::get_noise() {
38          return noise;
39
40      }
41
42      float noise_average_sink_impl::get_noise_subband() {
43          return noise[0];
44
45      }
46
47      int noise_average_sink_impl::work(int noutput_items,
48              gr_vector_const_void_star &input_items,
49              gr_vector_void_star &output_items)
50      {
51          const float *in0 = (const float *) input_items[0];
52          const short *in1 = (const short *) input_items[1];
53          int z=0;
54          while(z < noutput_items) {
55
56              for(int i=0; i<d_veclen; i++) {
57
58                  if(in1[i] == 0) {
59                      average[i] += in0[i];
60                      current_slots[i]++;
61                      if(current_slots[i] == d_slots) {
62                          noise[i] = average[i]/d_slots;
63                          current_slots[i] = 0;
64                          average[i] = 0.0;
65                   }
66                  }
67                  else {
68                      if(d_consecutive == 1) {
69                          average[i] = 0.0;
70                          current_slots[i] = 0;
71                      }
72                  }
73              }
74              in0 += d_veclen;
75              in1 += d_veclen;
76              z++;
77          }
78      return noutput_items;
79      }
80    }
81  }
```

Listing 9.8: noise_average_sink_impl.cc - Implementation source file.

2. **Estimation update**: GNU Radio creates threads through the *Function Probe* block. The thread calls the public method `get_noise` of *noise_average_sink* every second, as it can be seen from Fig. 9.7. The value `noisevar` is the initial noise estimation vector. By using the ID of the probe `noise_update_lin_probe`

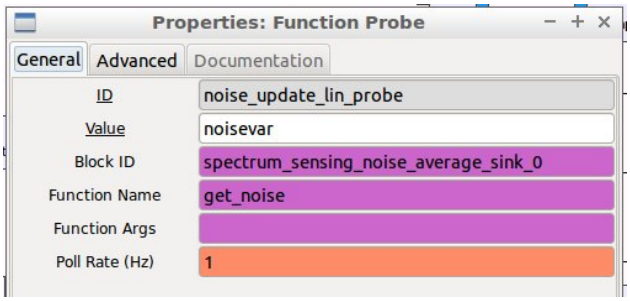


Figure 9.7: Function probe block.

as noise variance parameter of the *Energy_Detector_vector* block (see Fig. 9.5), the thread will call the `set_noisevar` callback method of the *Energy_Detector_vector* block and update the noise variance vector. This closes the feedback loop and in Python this is accomplished as follows:

```
1     def _noise_update_lin_probe_probe():
2         while True:
3             val = self.spectrum_sensing_noise_average_sink_0.get_noise()
4             try:
5                 self.set_noise_update_lin_probe(val)
6             except AttributeError:
7                 pass
8             time.sleep(1.0 / (1))
9     _noise_update_lin_probe_thread = threading.Thread(target=
          _noise_update_lin_probe_probe)
10    _noise_update_lin_probe_thread.daemon = True
11    _noise_update_lin_probe_thread.start()
12
13    def set_noise_update_lin_probe(self, noise_update_lin_probe):
14    self.noise_update_lin_probe = noise_update_lin_probe
15    self.spectrum_sensing_Energy_Detector_vector_0.set_noisevar(self.
          noise_update_lin_probe)
```

Listing 9.9: Function probe Python code.

The public methods of the implementations are accessible in GRC thanks to the `<callback>` tag of the XML definition file:

```
1 <callback>set_noisevar($noisevar)</callback>
```

9.6 External Graphical User Interface (GUI)

Starting from the GNU Radio UDP packets, An external GUI architecture has been developed by CSP - ICT Innovation. Fig. 9.8 shows a flowgraph with the technologies involved in the architecture. The GNU Radio UDP packets are processed through multiple stages and finally the spectrum of the signal and the waterfall with occupancy of each channel are illustrated in the browser. We briefly illustrate the steps:

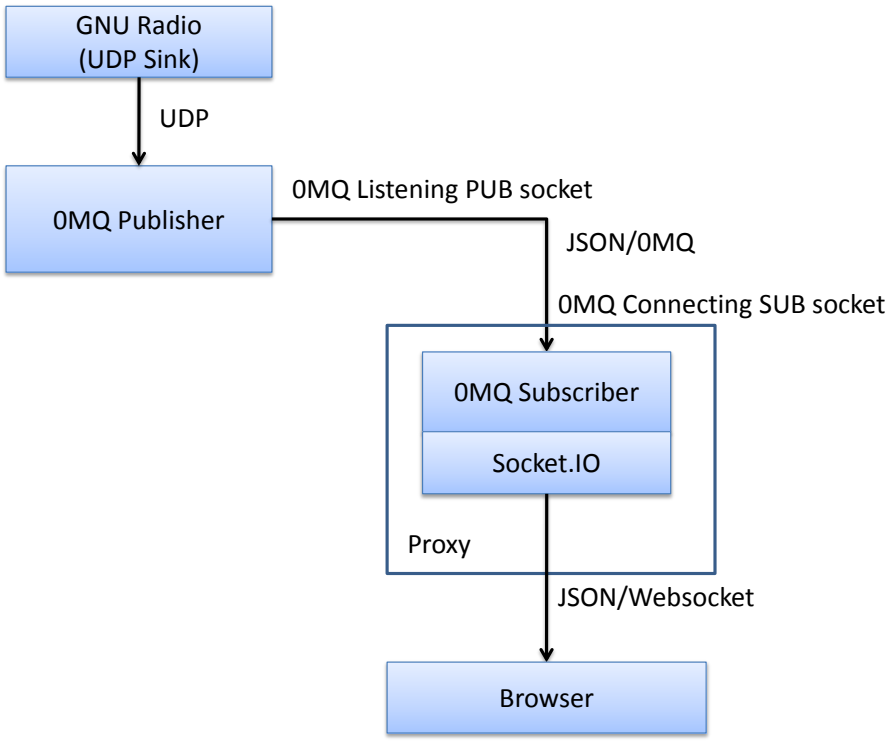


Figure 9.8: External GUI architecture.

1. The UDP datagrams are read and a ZeroMQ (0MQ) [116] socket is opened for each service (spectrum and waterfall) according to the broker-less *publish-subscribe* paradigm. This allows to establish a one-to-many connection. The

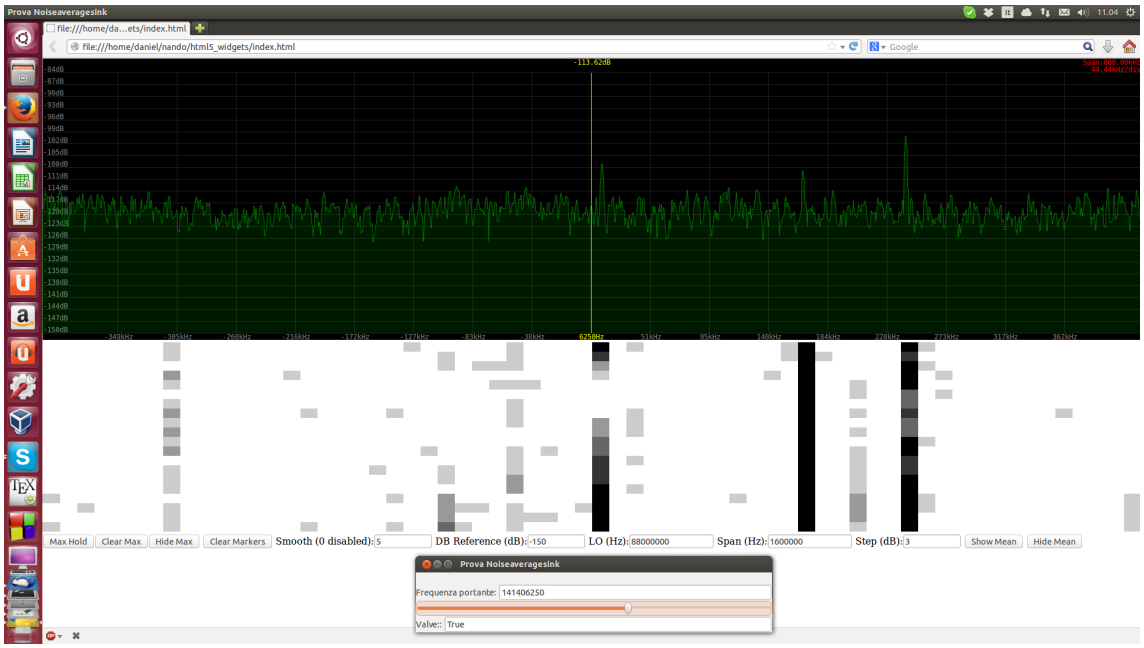


Figure 9.9: Signal spectrum and waterfall.

0MQ Publisher acts like a TCP server while the 0MQ Subscriber as a TCP client. The binary data coming from GNU Radio are converted into the JavaScript Object Notation (JSON) [117] format before publishing.

2. A proxy is created between the ZeroMQ sockets and websockets through the Socket.IO JavaScrpit library [118]. This allows a browser to act as a subscriber of the messages generated by the ZeroMQ publisher process.

3. Real time plots of the waterfall and the spectrum are implemented in Hyper-Text Markup Language (HTML) and Javascript by using the Canvas HTML element [119].

Waterfall and spectrum are shown in Fig. 9.9. The waterfall has one column per DMR channel, different grey scales represent different occupancy percentages. The thresholds for each color are at 1%, 5%, 30%, 60% and 90%.

Chapter 10

Eigenvalue-based detector implementation

In this chapter, the GNU Radio implementation of the eigenvalue-based detector is presented. The *eigenbased* custom block is described in detail: threshold computation for RLRT/GLRT and the chosen eigenvalue algorithm are illustrated. Finally, simulation results are given to prove the correctness of the eigenvalue algorithm.

10.1 Description of the *eigenbased* block

The *eigenbased* block has the following parameters:

- *Antennas*: it corresponds to K, the number of sensors, up to 8 antennas can be selected.
- *Samples*: it corresponds to N, the number of samples stored on each row of the $\boldsymbol{Y}$ matrix.
- *False Alarm Probability*: different values of P_{fa} can be selected from 0.001 to 0.2.
- *Test statistic*: it can be chosen RLRT or GLRT.
- *Noise variance*: noise power estimation is required only for RLRT.

The *eigenbased* block has K input vectors, which represent the rows of the $\boldsymbol{Y}$ matrix, and has 3 outputs, like the parallel energy detector block, but in this case they are all scalars:

- `out_test`: a float representing the pure test statistic: (2.63) for RLRT or (2.77) for GLRT;

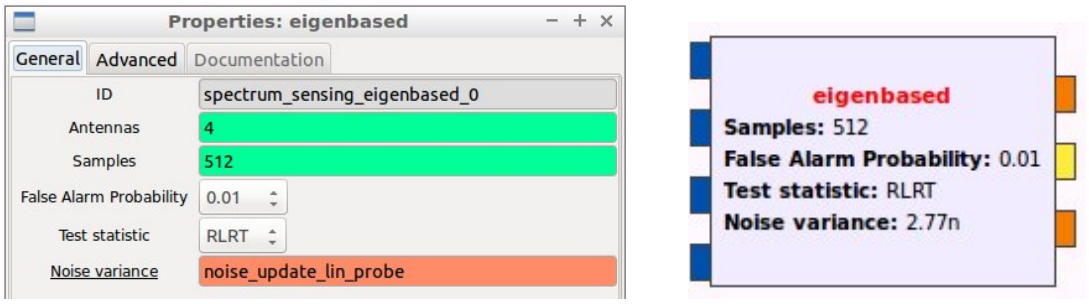


Figure 10.1: Eigenbased block with parameters.

- `out_bin`: a short integer representing the binary decision (0 or 1) after the comparison against the threshold;
- `out_noise_est`: a float equal to $\frac{1}{K}\text{tr}(\boldsymbol{R})$, which represents the mean energy of the K received streams.

unlike the previous case, this block works on time samples and outputs one test result per call. Its `io_signature` is the following:

```
gr::io_signature::make(2, 8, N * sizeof (gr_complex)),
gr::io_signature::make3(3, 3, sizeof (float), sizeof (unsigned short int), sizeof(float))),
```

with 8 the maximum number of antennas. In XML, the command `nports` allows to shape the block in GRC with as many input ports as the selected number of antennas:

```
<sink>
    <name>in1</name>
    <type>complex</type>
    <vlen>$N</vlen>
    <nports>$K</nports>
</sink>
```

10.2 Threshold computation

Threshold values have been computed offline in the Matlab environment for both RLRT and GLRT.

10.2.1 RLRT

When a large number of slots is used for noise variance estimation, the distributions of the P_{fa} of HRLRT1 (4.20) and HRLRT2 (4.30) converge to (2.69), so the threshold

has been computed according to (2.70)

$$t_{RLRT}(\alpha) = \mu + F_{TW2}^{-1}(1-\alpha)\xi$$

where F_{TW2}^{-1} is the inverse of the CDF of the TW2, μ and ξ are centering and scaling parameters. Two Matlab scripts have been used for RLRT: the first one computes $F_{TW2}^{-1}(1-\alpha)$ for different $P_{fa} = 1-\alpha$ values. The tables for computing the Tracy-Widom density and distribution functions have been computed by Momar Dieng's MATLAB package "RMLab" [120], the functions `tw` and `twdist` are part of that package. The bisection method is used to find the approximate value of $F_{TW2}^{-1}(1-\alpha)$ for each P_{fa}, the values are finally stored in a file:

```
tw;
pfa = [0.001 0.002 0.005 0.01 0.02 0.05 0.1 0.2]';
N = length(pfa);
toll = 2^(-23);
x0 = -5*ones(N,1);
x1 = 3*ones(N,1);
x_med = (x0+x1)./2;
target = twdist(2,1,x_med);
c=zeros(N,1);
for ii=1:N
        while (abs(target(ii)-(1-pfa(ii))) > toll)
        if ((1-pfa(ii)) < target(ii))
                x1(ii) = x_med(ii);
                else
                x0(ii) = x_med(ii);
                end
    x_med(ii) = (x0(ii)+x1(ii))/2;
    target(ii) = twdist(2,1,x_med(ii));
    c(ii) = c(ii)+1;
end
end
[pfa (1-pfa) target' x_med c]
fid = fopen('tw2.dat','wb');
fwrite(fid,x_med,'float32');
fclose(fid);
```

In the second script, μ and ξ are computed for different values of antennas K and samples N. They will be used as lookup tables in the C++ implementation file:

```
K = 2:20;
N = 1:1:10000;
mu = zeros(length(K), length(N));
xi = zeros(length(K), length(N));
```

```
for ii=1:length(K)
    for jj=1:length(N)
        mu(ii,jj) = (sqrt(N(jj))+ sqrt(K(ii))).^2;
        xi(ii,jj) = sqrt(mu(ii,jj)).*(N(jj).^(-1/2)+K(ii).^(-1/2))^(1/3);
    end
end
fid = fopen('mu.dat','wb');
fid2 = fopen('xi.dat','wb');
fwrite(fid,mu.','float32');
fwrite(fid2,xi.','float32');
fclose(fid);
fclose(fid2);
```

The final threshold computation is performed in the C++ implementation source file

```
thresh = (tw2[d_p_fa]*xi[row][col]+mu[row][col])/d_N;
```

where `tw2`, `xi` and `mu` are arrays associated to the stored files, `d_p_fa`, `row` and `col` are proper indices derived by the block parameters P_{fa}, K and N. The final normalization by N is due to the fact that the sample covariance matrix is defined as $\boldsymbol{R} = \frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^H$.

10.2.2 GLRT

Prof. Boaz Nadler has provided the function `TW_trace_ratio_threshold` [121] that numerically inverts (2.79)

$$\Pr\left[\frac{T_{GLRT}-\mu}{\xi} < s\right] \approx F_{TW2}(s) - \frac{1}{2NK}\left(\frac{\mu}{\xi}\right)^2 F''_{TW2}(s)$$

Pre-computed tables of the Tracy-Widom distributions and their derivatives using Matlab code have been provided by Prof. Folkmar Bornemann and based on [122]

```
pfa = [0.001 0.002 0.005 0.01 0.02 0.05 0.1 0.2]';
K = 2:1:20;
N = 1:1:10000;
GLRT_thresholds = zeros(length(pfa),length(K),length(N));
counter = 0
for pp= 1:length(pfa)
    for kk=1:length(K)
        for nn=1:length(N)
            [counter pp kk nn]
            [GLRT_thresholds(pp,kk,nn), ~] = TW_trace_ratio_threshold(K(kk),N(nn
                ),2,pfa(pp));
            counter= counter+1;
```

```
12            end
13        end
14    end
15
16    fid = fopen('GLRT_thresholds.dat','wb');
17    fwrite(fid, GLRT_thresholds(:,:,1).','float32');
18    fclose(fid);
19
20    for ii = 2:length(pfa)
21        fid = fopen('GLRT_thresholds.dat','a');
22        fwrite(fid, GLRT_thresholds(:,:,ii).','float32');
23        fclose(fid);
24    end
```

The stored file contains a 3D-array in this case. Finally the proper threshold value is selected as in the RLRT case in the C++ implementation source file:

```
thresh = glrt_thresh[d_p_fa][row][col];
```

with `glrt_thresh` the 3D-array associated to the `GRLT_thresholds` file.

10.3 Eigenvalue algorithm

The most significant part of *eigenbased* block consists in the computation of eigenvalues of the covariance matrix $\boldsymbol{R}$. Such computation is performed through 2 stages: the Lanczos algorithm and the bisection method.

10.3.1 Lanczos algorithm

In order to choose the best possible solution, we have to take into account that the covariance matrix $\boldsymbol{R}$ is Hermitian, hence the Lanczos method can be applied.

The Lanczos algorithm can be viewed as a simplified Arnoldi's algorithm [123] in that it applies to Hermitian matrices. In the algorithm a series of orthonormal vectors, $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$, is generated, which satisfy:

$$\boldsymbol{T} = \boldsymbol{Q}^T \boldsymbol{R}\, \boldsymbol{Q} \tag{10.1}$$

The matrix $\boldsymbol{T}$ is tridiagonal and similar to the matrix $\boldsymbol{R}$:

$$\boldsymbol{T}_j = \begin{pmatrix} \alpha_1 & \beta_2 & & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \beta_{j-1} & \\ & & & \beta_{j-1} & \alpha_{j-1} & \beta_j \\ 0 & & & & \beta_j & \alpha_j \end{pmatrix}$$

the iterative procedure is based on this three-term recurrence relation:

$$\beta_{j+1}\,\boldsymbol{q}_{j+1} = \boldsymbol{R}\,\boldsymbol{q}_j - \alpha_j\,\boldsymbol{q}_j - \beta_j\,\boldsymbol{q}_{j-1} \tag{10.2}$$

More details on the procedure can be found in [124] and [125]. Algorithm 2 shows a short description of the Lanczos algorithm in pseudocode. The main problem

Algorithm 2 Lanczos algorithm in pseudocode.

function LANCZOS($\boldsymbol{R}, K$)
  $\boldsymbol{q_1}$ uniformly distributed random vector
  $\boldsymbol{q_1} = \boldsymbol{q_1}/||\boldsymbol{q_1}||$
  $\alpha_1 = \boldsymbol{q}_1^H\,\boldsymbol{R}\,\boldsymbol{q}_1$
  $\boldsymbol{w}_1 = \boldsymbol{R}\,\boldsymbol{q}_1 - \alpha_1\,\boldsymbol{q}_1$
  $\beta_1 = 0$
  $\beta_2 = ||\boldsymbol{w}_1||$
  for $i = 2 \rightarrow K$ **do**
    $\boldsymbol{q}_i = \boldsymbol{w}_{i-1}/\beta_i$
    $\alpha_i = \boldsymbol{q}_i^H\,\boldsymbol{R}\,\boldsymbol{q}_i$
    $\boldsymbol{w}_i = \boldsymbol{R}\,\boldsymbol{q}_i - \alpha_i\,\boldsymbol{q}_i - \beta_i\,\boldsymbol{q}_{i-1}$
    if $i < K$ **then**
      $\beta_{i+1} = ||\boldsymbol{w}_i||$
    end if
  end for
end function

of the Lanczos algorithm is stability, in fact, by using floating point arithmetic, the orthogonality of the vectors $\boldsymbol{q}_j$ is quickly lost. Several stable orthogonalization schemes have been proposed, such as [126], although none of these methods have been applied in our block, since we are basically interested in the computation of the maximum eigenvalue of the covariance matrix. As a matter of fact, even with such loss of orthogonality, the algorithm generates very good approximations of the largest eigenvalue. As proven in [127] the Lanczos algorithm produces faster and more accurate results than the power method.

10.3.2 Bisection algorithm

Once that, after K iterations, the tridiagonal matrix $\boldsymbol{T}$ has been obtained, the maximum eigenvalue can be simply computed through the bisection algorithm (also called spectral bisection). The algorithm is an iterative procedure based on the computation of the modified Sturm sequence, as explained in [128] or [129]. The

original Sturm sequence, for any number c, is based on the following recursive relation:

$$\begin{aligned} p_0(c) &= 1 \\ p_1(c) &= \alpha_1 - c \\ p_i(c) &= (\alpha_i - c)p_{i-1}(c) - \beta_i^2 p_{i-2}(c) \end{aligned} \tag{10.3}$$

where $[\alpha_1, \ldots, \alpha_K]$ and $[\beta_2, \ldots, \beta_K]$ are respectively the diagonal and off-diagonal elements of the matrix $\boldsymbol{T}$. The number $f(c)$ of disagreements in sign between consecutive number of the sequence is equal to the number of eigenvalues smaller than c. However, the original sequence suffers from underflow and overflow problems in floating-point arithmetic, thus, the original sequence $p_i(c)$ is replaced by the sequence $q_i(c)$ defined as

$$q_i(c) = p_i(c)/p_{i-1}(c) \tag{10.4}$$

now the number of eigenvalues smaller than c, $f(c)$ is given by the number of negative $q_i(c)$. Hence the new recursive relation is:

$$\begin{aligned} q_1(c) &= \alpha_1 - c \\ q_i(c) &= (\alpha_i - c) - \beta_i^2/q_{i-1}(c) \end{aligned} \tag{10.5}$$

The algorithm implemented in the block is a slightly modified version of the bisection method described in [128], which computes only the largest eigenvalue. The first part of the algorithm exploits the Gershgorin circle theorem to estimate the upper and lower bounds of the eigenspectrum of the matrix $\boldsymbol{T}$. As shown in Algorithm 3, the bisection function has 6 parameters:

- $\boldsymbol{\alpha}$: diagonal elements of the matrix $\boldsymbol{T}$;
- $\boldsymbol{\beta}$: off-diagonal elements of $\boldsymbol{T}$;
- K: order of the tridiagonal matrix;
- m: eigenvalue λ_m is computed, in this algorithm λ_K is the largest eigenvalue (opposite notation w.r.t. Sec. 2.2.1), hence $m = K$;
- γ: required precision for the computation of the eigenvalue, which affects the number of iterations;
- ϵ: machine epsilon, the smallest number for which $1 + \epsilon > 1$ in the computer.

Algorithm 3 Bisection algorithm in pseudocode.

```
1: function BISECTION(α, β, K, m, γ, ϵ)
2:     xmin = α_K − |β_K|
3:     xmax = α_K + |β_K|
4:     for i = K − 1 → 1 do
5:         h = |β_i| + |β_{i+1}|
6:         if α_i + h > xmax then
7:             xmax = α_i + h
8:         end if
9:         if α_i − h < xmin then
10:            xmin = α_i − h
11:        end if
12:    end for
13:    u = xmax
14:    x = xmax
15:    v = xmin
16:    while (u − v) > 2ϵ(|v| + |u| + γ) do
17:        p = (u + v)/2
18:        a = 0
19:        q = 1
20:        for i = 1 → K do
21:            if q ≠ 0 then
22:                q = α_i − p − β_i^2/q
23:            else
24:                q = α_i − p − |β/ϵ|
25:            end if
26:            if q < 0 then
27:                a = a + 1
28:            end if
29:        end for
30:        if a < m then
31:            v = p
32:        else
33:            u = p
34:        end if
35:    end while
36:    x = (u + v)/2
37:    return x
38: end function
```

10.4 *Eigenbased* block code

Listing 10.1 shows the public header file of the *Eigenbased* block. The `set_noisevar` public method is the callback function that allows to update the noise variance estimate in case the online noise estimation approach is considered, with the same feedback loop mechanism explained in Sec. 9.5

```
1  #include <spectrum_sensing/api.h>
2  #include <gnuradio/block.h>
3  #include <gnuradio/gr_complex.h>
4  #include <gnuradio/math.h>
5  #include <iostream>
6  #include <random>
7  #include <cmath>
8  #include <chrono>
9  #include <cstring>
10 #include <cfloat>
11 #include <ctime>
12
13 namespace gr {
14   namespace spectrum_sensing {
15
16     class SPECTRUM_SENSING_API eigenbased : virtual public gr::block
17     {
18      public:
19       typedef boost::shared_ptr<eigenbased> sptr;
20
21       static sptr make(int N, int K, unsigned short int p_fa, unsigned short int test, float
             noisevar);
22       virtual void set_noisevar(float noisevar) = 0;
23     };
24   }
25 }
```

Listing 10.1: eigenbased.h - Public header file.

Besides the `general_work`, the `forecast` and the aforementioned `set_noisevar` public methods, the implementation contains 5 private methods:

- the `trace` method computes the trace of a matrix;
- the `norml2` computes the Euclidean or L^2 norm of vector;
- given a matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$, the method `vetmatvetprod` returns the result of $\boldsymbol{b}^H \boldsymbol{A}\boldsymbol{b}$;
- the `lanczos` method applies the Lanczos algorithm to the matrix $\boldsymbol{R}$ and outputs the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, which respectively contain the diagonal and off-diagonal elements of the matrix $\boldsymbol{T}$;

- the `bisection` method applies the spectral bisection algorithm to $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ and returns the largest eigenvalue of $\boldsymbol{T}$, which is the same of $\boldsymbol{R}$.

Listings 10.2 and 10.3 respectively show the implementation header and source file.

```
1  #include <spectrum_sensing/eigenbased.h>
2
3  namespace gr {
4    namespace spectrum_sensing {
5
6      class eigenbased_impl : public eigenbased
7      {
8       private:
9       int d_K;
10      int d_N;
11      unsigned short int d_p_fa;
12      unsigned short int d_test;
13      float d_noisevar;
14      float *tw2;
15      float **mu;
16      float **xi;
17      float ***glrt_thresh;
18      FILE *fp;
19      gr_complex **mat_R;
20      float trace(gr_complex **);
21      float norml2(gr_complex *);
22      float vetmatvetprod(gr_complex **, gr_complex *);
23      void lanczos();
24      float bisect(float *, float *, int, int, float, float);
25      float *alfa;
26      float *beta;
27      float maxeig;
28      float thresh;
29
30       public:
31       eigenbased_impl(int N, int K, unsigned short int p_fa, unsigned short int test, float
             noisevar);
32       ~eigenbased_impl();
33      void set_noisevar(float noisevar);
34       void forecast (int noutput_items, gr_vector_int &ninput_items_required);
35       int general_work(int noutput_items,
36         gr_vector_int &ninput_items,
37         gr_vector_const_void_star &input_items,
38         gr_vector_void_star &output_items);
39     };
40   }
41 }
```

Listing 10.2: eigenbased_impl.h - Implementation header file.

```
1  #include "config.h"
2
3  #include <gnuradio/io_signature.h>
4  #include "eigenbased_impl.h"
5  #define COUNT 100000
6  #define SAMPLES 10000
7  #define PFA 8
8  #define STEP 1
9  #define ANTENNAS 19
10
11 using namespace std;
12
13 namespace gr {
14   namespace spectrum_sensing {
15
16     eigenbased::sptr
17     eigenbased::make(int N, int K, unsigned short int p_fa, unsigned short int test, float
           noisevar)
18     {
19       return gnuradio::get_initial_sptr
20         (new eigenbased_impl(N, K, p_fa, test, noisevar));
21     }
22     eigenbased_impl::eigenbased_impl(int N, int K, unsigned short int p_fa, unsigned short int
           test, float noisevar)
23       : gr::block("eigenbased",
24                   gr::io_signature::make(2, 8, N * sizeof (gr_complex)),
25                   gr::io_signature::make3(3, 3, sizeof (float), sizeof (unsigned short int), sizeof(float))
                         ),
26               d_K(K),
27               d_N(N),
28               d_p_fa(p_fa),
29               d_test(test),
30               d_noisevar(noisevar)
31     {
32   set_noisevar(noisevar);
33   set_relative_rate(1.0);
34
35    if(d_test == 1) {
36
37         mu = new float *[ANTENNAS];
38             for(int k=0;k<ANTENNAS;k++)
39                 mu[k] = new float [SAMPLES];
40         xi = new float *[ANTENNAS];
41             for(int k=0;k<ANTENNAS;k++)
42                 xi[k] = new float [SAMPLES];
43         tw2 = new float [PFA];
44
```

```
45                      if ((fp = fopen ("/home/daniel/Scrivania/gr-spectrum_sensing_3.7.3/mu.
                            dat","rb")) == NULL)
46                          cout << "File vuoto1\n" << endl;
47                      for (int i=0;i<ANTENNAS;i++)
48                          fread(mu[i],sizeof(float),SAMPLES,fp);
49                      fclose(fp);
50
51                      if ((fp = fopen ("/home/daniel/Scrivania/gr-spectrum_sensing_3.7.3/xi.
                            dat","rb")) == NULL)
52                          cout << "File vuoto2\n" << endl;
53                      for (int i=0;i<ANTENNAS;i++)
54                          fread(xi[i],sizeof(float),SAMPLES,fp);
55                      fclose(fp);
56
57                      if ((fp = fopen ("/home/daniel/Scrivania/gr-spectrum_sensing_3.7.3/tw2.
                            dat","rb")) == NULL)
58                          cout << "File vuoto3\n" << endl;
59                      fread(tw2, sizeof(float), PFA, fp);
60                      fclose(fp);
61
62                      if ((fp = fopen ("/home/daniel/Scrivania/gr-spectrum_sensing_3.7.3/tw2.
                            dat","rb")) == NULL)
63                          cout << "File vuoto3\n" << endl;
64                      fread(tw2, sizeof(float), PFA, fp);
65                      fclose(fp);
66
67      } else if(d_test==2) {
68
69              glrt_thresh = new float **[PFA];
70              for(int p=0;p<PFA;p++) {
71                  glrt_thresh[p] = new float *[ANTENNAS];
72                  for (int k=0;k<ANTENNAS;k++) {
73                      glrt_thresh[p][k] = new float [SAMPLES];
74                  }
75              }
76              if ((fp = fopen ("/home/daniel/Dropbox/gr-spectrum_sensing_3.7.3/
                        GLRT_thresholds.dat","rb")) == NULL)
77                  cout << "File vuoto1\n" << endl;
78              for(int p=0;p<PFA;p++) {
79                  for (int k=0;k<ANTENNAS;k++) {
80                      fread(glrt_thresh[p][k],sizeof(float),SAMPLES,fp);
81                  }
82              }
83              fclose(fp);
84        }
85          mat_R = new gr_complex *[d_K];
86          for(int i=0;i<d_K;i++)
87              mat_R[i] = new gr_complex [d_K];
88          alfa = new float [d_K];
```

```
89          beta = new float [d_K];
90      }
91
92      eigenbased_impl::~eigenbased_impl()
93      {
94          for(int i=0; i<d_K; i++)
95              delete [] mat_R[i];
96          delete [] mat_R;
97          if(d_test==1) {
98              for(int i=0;i<ANTENNAS;i++)
99                  delete [] mu[i];
100             delete [] mu;
101             for(int i=0;i<ANTENNAS;i++)
102                 delete [] xi[i];
103             delete [] xi;
104             delete [] tw2;
105         }
106         else if(d_test==2) {
107             for(int i=0;i<PFA;i++) {
108                 for(int j=0;j<ANTENNAS;j++)
109                     delete [] glrt_thresh[i][j];
110                 delete [] glrt_thresh[i];
111             }
112             delete [] glrt_thresh;
113         }
114         delete [] alfa;
115         delete [] beta;
116     }
117
118     void eigenbased_impl::set_noisevar(float noisevar)
119     {
120         if(noisevar != 0)
121         d_noisevar = noisevar;
122     }
123
124     void eigenbased_impl::forecast (int noutput_items, gr_vector_int &ninput_items_required
            )
125     {
126         for(int i=0; i<d_K; i++) {
127             ninput_items_required[i] = noutput_items;
128         }
129     }
130
131     float eigenbased_impl::trace(gr_complex **mat)
132     {
133         float tot = 0;
134         for(int i=0;i<d_K;i++) {
135             tot += mat[i][i].real();
136         }
```

```
137         return tot;
138     }
139
140     float eigenbased_impl::norml2(gr_complex *vet)
141     {
142         float tot = 0;
143         for(int i=0; i<d_K; i++) {
144             tot += norm(vet[i]);
145         }
146         tot = sqrt(tot);
147         return tot;
148     }
149
150     float eigenbased_impl::vetmatvetprod(gr_complex **mat, gr_complex *vet)
151     {
152         float alfan;
153         gr_complex tot = 0;
154         gr_complex tempvet[d_K];
155         for(int i=0; i<d_K; i++) {
156             for(int j=0; j<d_K; j++) {
157                 tempvet[i] += (mat[i][j] * vet[j]);
158             }
159             tot += (conj(vet[i])*tempvet[i]);
160         }
161         alfan = tot.real();
162         return alfan;
163     }
164
165     void eigenbased_impl::lanczos()
166     {
167         gr_complex v[d_K];
168         gr_complex w[d_K];
169         gr_complex v_old[d_K];
170         gr_complex w_old[d_K];
171         memset(w, 0, d_K*sizeof(gr_complex));
172         memset(w_old, 0, d_K*sizeof(gr_complex));
173         unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
174         std::minstd_rand0 g1 (seed);
175         std::uniform_real_distribution<float> uniform(-1.0,1.0);
176
177         for(int i=0;i<d_K;i++) {
178             v[i] = gr_complex(uniform(g1), uniform(g1));
179         }
180         float b0 = norml2(v);
181         for(int i=0;i<d_K;i++) {
182             v[i] = v[i] / gr_complex(b0,0);
183         }
184         beta[0] = 0;
185         alfa[0] = vetmatvetprod(mat_R, v);
```

```
186
187         for(int i=0; i<d_K; i++) {
188             for(int j=0; j<d_K; j++) {
189                 w[i] += (mat_R[i][j] * v[j]);
190              }
191             w[i] = w[i] - (gr_complex(alfa[0], 0) * v[i]);
192         }
193         beta[1] = norml2(w);
194         memcpy(v_old, v, d_K*sizeof(gr_complex));
195         memcpy(w_old, w, d_K*sizeof(gr_complex));
196         memset(v, 0, d_K*sizeof(gr_complex));
197         memset(w, 0, d_K*sizeof(gr_complex));
198
199         for(int m=1; m<d_K; m++) {
200             for(int i=0;i<d_K;i++) {
201                 v[i] = w_old[i] / gr_complex(beta[m],0);
202              }
203             alfa[m] = vetmatvetprod(mat_R, v);
204             for(int i=0; i<d_K; i++) {
205                 for(int j=0; j<d_K; j++) {
206                     w[i] += (mat_R[i][j] * v[j]);
207                 }
208                 w[i] = w[i] - (gr_complex(alfa[m], 0) * v[i]) - (gr_complex(beta[m], 0) *
                        v_old[i]);
209             }
210             if(m < (d_K-1)) {
211                 beta[m+1] = norml2(w);
212              }
213             memcpy(v_old, v, d_K*sizeof(gr_complex));
214             memcpy(w_old, w, d_K*sizeof(gr_complex));
215             memset(v, 0, d_K*sizeof(gr_complex));
216             memset(w, 0, d_K*sizeof(gr_complex));
217         }
218     }
219
220     float eigenbased_impl::bisect(float *c, float *b, int n, int m, float eps, float relfeh)
221     {
222         float xmin = c[n-1] - abs(b[n-1]);
223         float xmax = c[n-1] + abs(b[n-1]);
224
225         for(int i=n-2; i>=0; i--) {
226             float h = abs(b[i]) + abs(b[i+1]);
227             if((c[i]+h) > xmax)
228                 xmax = c[i]+h;
229             if((c[i]-h) < xmin)
230                 xmin = c[i]-h;
231         }
232         float x0 = xmax;
233         float x = xmax;
```

```
234         float xu = xmin;
235         float x1, q;
236         int a; //z=0;
237 
238         while ((x0-xu) > (2*relfeh*(abs(xu)+abs(x0))+eps)) {
239             x1 = (xu+x0)/2;
240             a = 0;
241             q = 1.0;
242             for (int i=0;i<n;i++) {
243                 if (q!=0)
244                     q=c[i]-x1-b[i]*b[i]/q;
245                 else
246                     q=c[i]-x1-abs(b[i]/relfeh);
247                 if(q<0) a++;
248             }
249             if (a<m) xu = x1;
250             else x0=x1;
251         }
252         x = (x0+xu)/2;
253         return x;
254     }
255 
256     int eigenbased_impl::general_work (int noutput_items,
257                         gr_vector_int &ninput_items,
258                         gr_vector_const_void_star &input_items,
259                         gr_vector_void_star &output_items)
260     {
261         if((d_K > 8) || (d_K < 2)) {
262             cout << "Number of sensors must be between 2 and 8.\n" << endl;
263             return -1;
264         }
265         int z = 0;
266         const gr_complex *input[d_K];
267         for(int i=0;i<d_K;i++) {
268             input[i] = (const gr_complex *) input_items[i];
269         }
270         float *out_test = (float *) output_items[0];
271         unsigned short int *out_bin = (unsigned short int *) output_items[1];
272         float *out_noise_est = (float *) output_items[2];
273 
274         while(z < noutput_items) {
275 
276             memset(alfa, 0, d_K*sizeof(float));
277             memset(beta, 0, d_K*sizeof(float));
278             for(int i=0; i<d_K; i++) {
279                 memset(mat_R[i], 0, d_K * sizeof(gr_complex));
280             }
281 
282             for (int i = 0; i < d_K; i++){
```

```
283                 for(int m = 0; m <= i; m++) {
284                     for(int j=0; j < d_N; j++) {
285                         mat_R[i][m] += (input[i][j]) * conj(input[m][j]);
286                     }
287                     mat_R[i][m] *= gr_complex(1.0/d_N,0);
288                     if(i==m) {
289                         mat_R[i][m] = gr_complex((float)(mat_R[i][m].real()), 0 );
290                     }
291                     else {
292                         mat_R[m][i] = conj(mat_R[i][m]);
293                     }
294                 }
295             }
296             lanczos();
297             maxeig = bisect(alfa, beta, d_K, d_K, FLT_EPSILON, FLT_EPSILON);
298 
299             if(d_test == 1) {
300                 out_test[z] = maxeig / d_noisevar;//RLRT
301             }
302             else if(d_test == 2) {
303                 out_test[z] = (maxeig * d_K) / (trace(mat_R)); //GLRT
304             }
305             else {
306                 out_test[z] = 0;
307                 cout << "Wrong test selection" << endl;
308             }
309 
310             const int row = d_K - 2;
311             const int col = d_N/STEP-1;
312 
313             if(d_test==1)
314                 thresh = (tw2[d_p_fa]*xi[row][col]+mu[row][col])/d_N;
315             else if(d_test==2)
316                 thresh = glrt_thresh[d_p_fa][row][col];
317 
318             if(out_test[z] >= thresh) {
319                 out_bin[z] = 1;
320             }
321             else {
322                 out_bin[z] = 0;
323             }
324             out_noise_est[z] = abs(trace(mat_R))/d_K;
325 
326             for(int i=0;i<d_K;i++) {
327                 input[i]+=d_N;
328             }
329             z++;
330         }
331 
```

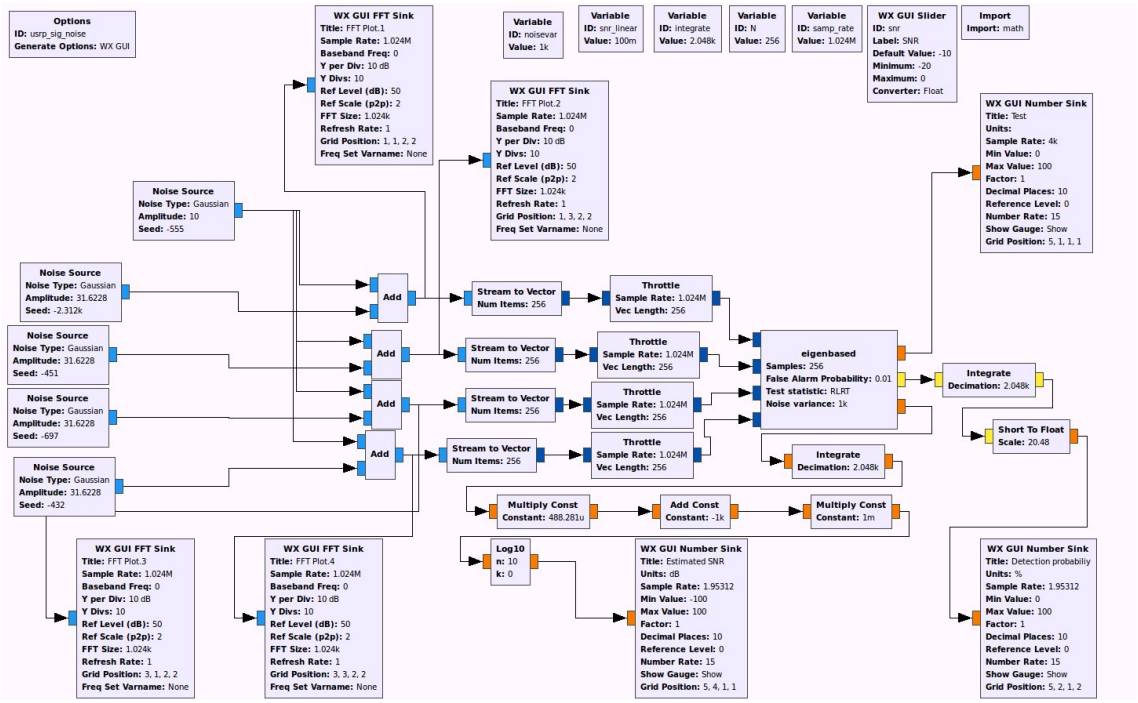


Figure 10.2: Simulation flowgraph for the *eigenbased* block.

```
332             consume_each (noutput_items);
333             return noutput_items;
334         }
335         }
336 }
```

Listing 10.3: eigenbased_impl.cc - Implementation source file.

.

10.5 Simulation results

The block has been tested in order to check the correctness of the eigenvalue algorithm. GNU Radio offers some useful simulation tools, like the *Noise Source* block. In the flowgraph of Fig. 10.2, the *eigenbased* block has been tested with $K = 4$ antennas, hence 4 *Noise Source* blocks have been used all with different seeds, and 1 *Noise Source* block as wideband input signal, $N = 256$ and $P_{fa} = 0.01$.

A snapshot of the performance results is shown in the WX GUI of Fig. 10.3. The GUI shows the FFT of each of the 4 inputs. A slider allows to change the SNR from -20 to 0 dB, SNR is also estimated in the flowgraph for further verification. Test

statistic, detection probability and estimated SNR are shown through three *WX GUI Number Sink* blocks. Fig. 10.3 shows $P_d = 0.587$ for $SNR = -12$ dB. By varying the SNR and storing the corresponding P_d values, we obtain the performance curve of Fig. 10.4.

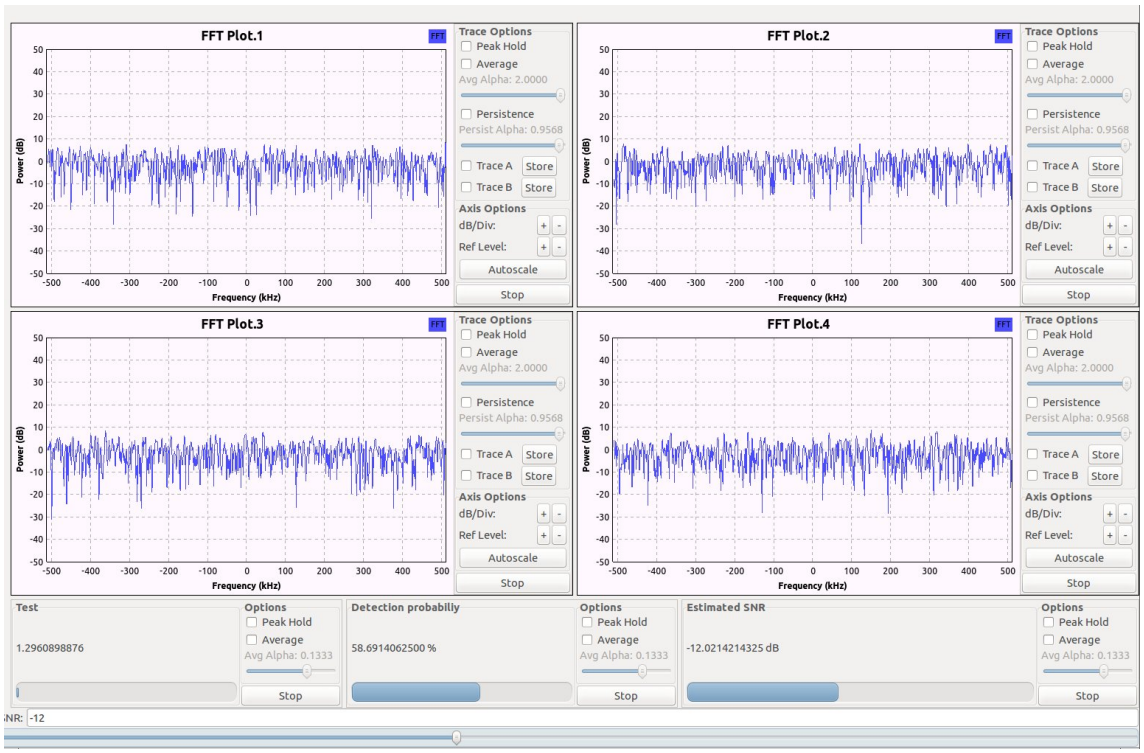


Figure 10.3: WX GUI of the simulation flowgraph for *eigenbased* block.

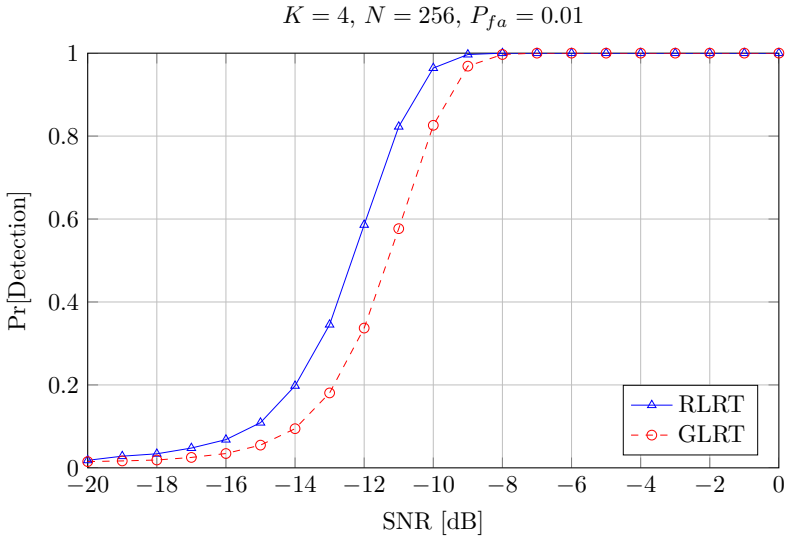


Figure 10.4: Performance curve (P_d vs. SNR) for the *eigenbased* block.

Chapter 11

Software-defined radio peripherals for multi-antenna cognitive testbeds

In this chapter, we present two sets of software-defined radio peripherals, which have been deployed for preliminary multi-antenna cognitive testbeds in the CITI Laboratory of INSA-Lyon [130]. The radio peripherals are:

- National Instruments USRP 2920,
- Nutaq PicoSDR 4x4.

First of all, for the first radio, a short description is given and a preliminary multi-antenna testbed on eigenvalue-based detection is illustrated, results and encountered issues are then discussed. Secondly, the PicoSDR 4x4 is presented and details on how to use it in GNU Radio are given.

11.1 NI USRP-2920

NI USRP-2920 is based on Ettus Research hardware and it is ideally suited for applications requiring high RF performance and great bandwidth. NI-USRP is based on Ettus Resarch USRP N210 and WBX daughterboard as RF section.

USRP N210 architecture includes a Xilinx Spartan 3A-DSP 3400 FPGA, 100 MS/s dual A/D converter, 400 MS/s dual D/A converter and Gigabit Ethernet connectivity to stream data to and from host processors. A modular design allows the USRP N210 to operate from DC to 6 GHz, while an expansion port allows multiple USRP N210 series devices to be synchronized and used in pairs in a MIMO configuration through the so called "MIMO cable". An optional GPS disciplined oscillator (GPSDO) module can also be used to discipline the USRP N210 reference clock to within 0.01 ppm of the worldwide Global Positioning System (GPS) standard.

The USRP N210 can stream up to 50 MS/s to and from host applications. Fig. 11.1 shows the schematic of the USRP N210 [131].

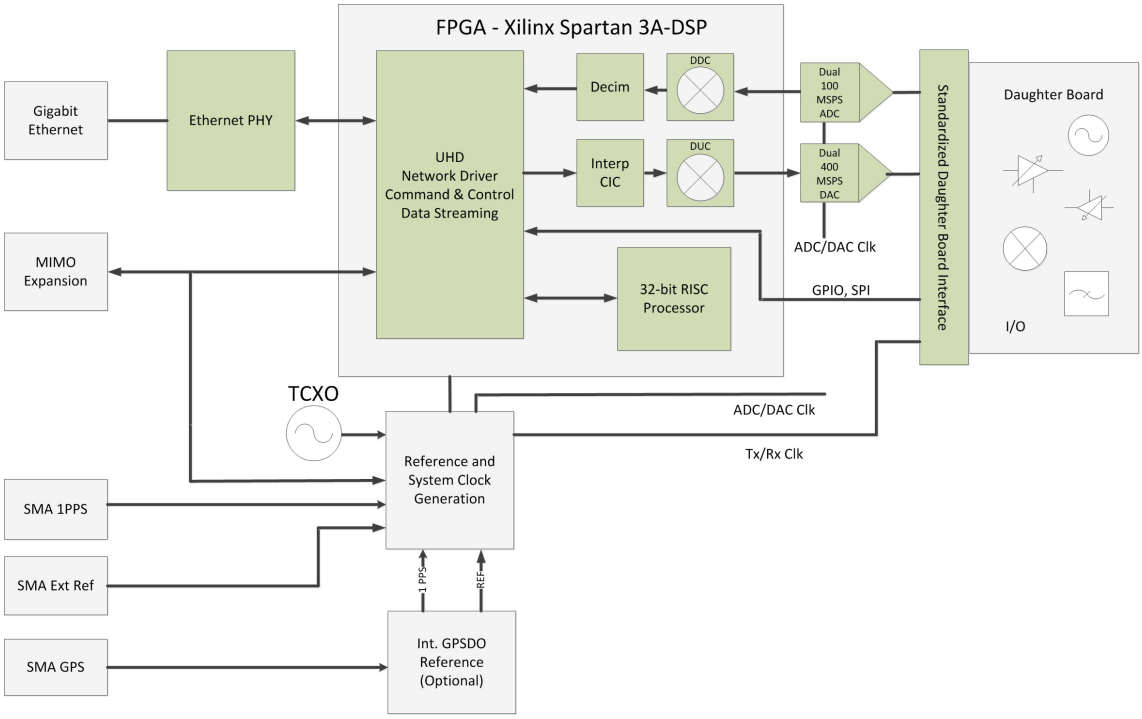


Figure 11.1: USRP N210 schematic.

The WBX daughterboard is a wide bandwidth full-duplex transceiver that provides up to 100 mW of output power and a noise figure of 5 dB. The local oscillators for the receive and transmit chains operate independently, but can be synchronized for MIMO operation. The WBX provides 40 MHz of bandwidth capability and is ideal for applications requiring access to a number of different bands within its range - 50 MHz to 2.2 GHz.

11.1.1 GNU Radio eigenvalue-based testbed

In order to check the performance of eigenvalue-based detectors with real time signals, a GNU Radio based testbed has been carried out in the CITI Laboratory - INSA Lyon. The scenario is composed of a single primary transmitting signal and $K = 4$ receivers. The testbed, shown in Fig. 11.2 has involved the following equipment:

TX

- 1 host PC;

- 1 NI USRP-2920;
- 1 Aaronia OmniLOG 70600 omni-directional antenna [132].

RX

- 1 host PC;
- 4 NI USRP-2920;
- 2 MIMO cables;
- 4 Aaronia OmniLOG 70600 omni-directional antennas [132].

It is important to point out that in this testbed the receivers were synchronized only in pairs. Eigenvalue-Based detection (EBD) algorithms require for optimal performance sample synchronization among all antennas; in this case 2 MIMO cables allowed to synchronize the 4 USRPs only in pairs, which means that we expected a substantial decrease of the performance.

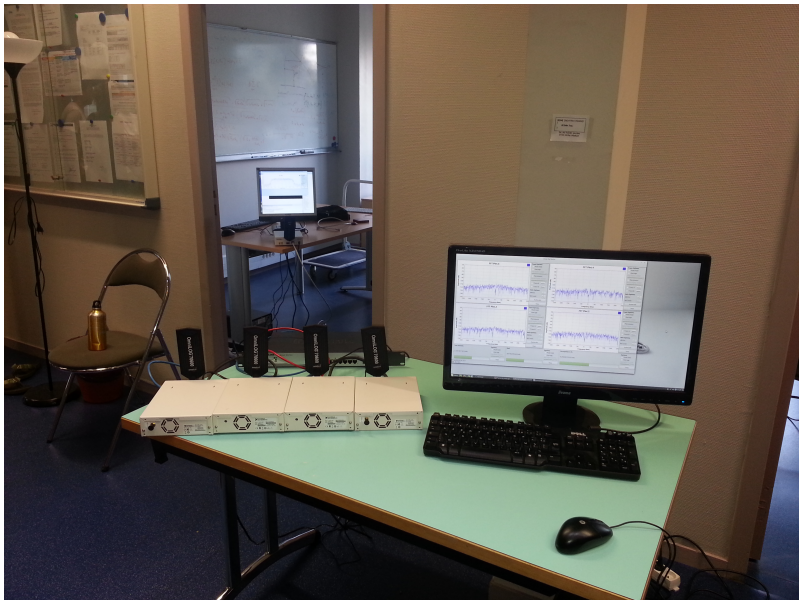


Figure 11.2: Multi-antenna eigenvalue-based detection testbed.

The GNU Radio flowgraph of Fig. 11.3 shows that the primary user signal is a QPSK modulated signal. The *WX GUI Text Box* allows to change a block parameter during runtime, in this case the `gain` variable is used in the *Multiply Const* block, which enables power tuning at the transmitter side.

The flowgraph at the receiving host is shown in Fig. 11.4 and shows many similarities with the flowgraph used for simulation purposes of Fig. 10.2. *Noise Source* and *Add* blocks have been replaced by 4 *UHD: USRP Source* blocks. USRPs at both transmitter and receiver are tuned at 2.49 GHz with a sample rate of 500 kS/s. For DC component removal, it has been used the *DC blocker* block filter with 32 taps instead of the *Moving Average* plus the *Subtract* blocks described in Sec. 9.3.

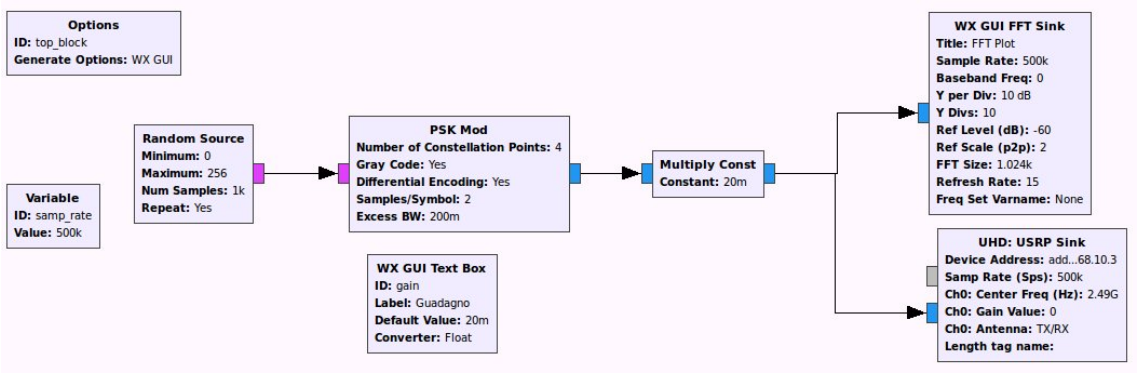


Figure 11.3: Flowgraph at the TX host with 1 USRP sink.

11.1.2 Results and issues

The results showed a decrease in the performance of RLRT and GLRT of about 4 dB with respect to the results presented in Sec. 10.5, thus nullifying the performance advantage of RLRT and GLRT with respect to Energy Detection.

As before mentioned, in this preliminary testbed a fully synchronized *single-input and multiple-output* (SIMO) system was not available. One pair of USRP N210 connected through a MIMO cable allows to build a synchronized 2x2 MIMO system with a common reference signal. USRP devices take two reference signals in order to synchronize clocks and time:

- A 10 MHz reference to provide a single frequency reference for both devices.
- A pulse-per-second (PPS) to synchronize the sample time across devices.
- A MIMO cable transmits an encoded time message from one device to another.

The 4 USRPs of the testbeds used 2 common reference signals and the lack of synchronization caused:

- frequency offset,

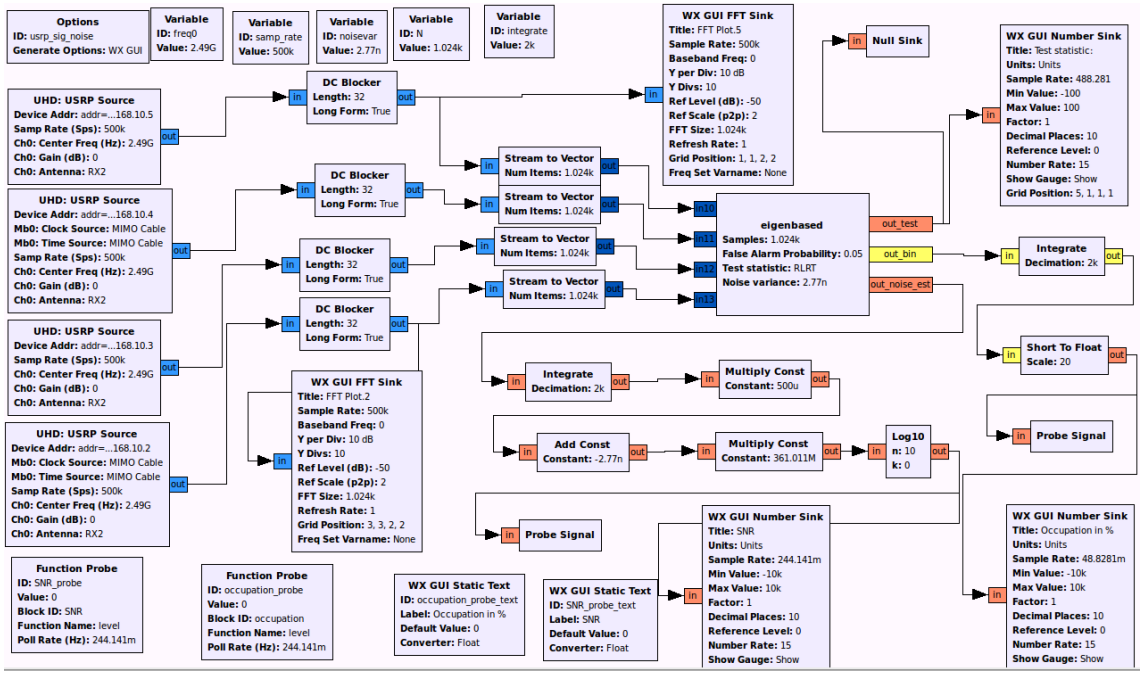


Figure 11.4: Flowgraph at the RX host with 4 USRP sources.

- phase offset.

Also time delay has to be taken into account, but it is negligible with respect to the 2 mentioned offsets.

Please note that based on the performance results shown in Fig. 3.15 and 3.16, a possible configuration with only 2 receiving devices was not taken into account, due to the poor performance of EBD algorithms when $K = 2$.

For USRP-based systems, external timing reference and distribution systems, as e.g., Ettus Research OctoClock-G [133], able to synchronize up to 16 USRP devices in combination with MIMO cables, are the only solutions that would allow to build a fully synchronized testbed. Another possible non USRP-based solution is presented in the next section.

11.2 Nutaq PicoSDR 4x4

The Nutaq PicoSDR 4x4 is a MIMO prototyping platform with a model-based design environment and GNU Radio support. It incorporates two multimode SDR dual-channel RF transceiver modules, FPGA logic and memory that can be stacked together to form a 4x4 MIMO solution from baseband processing to the air interface.

The PicoSDR uplinks and downlinks data streams to a remote computer through high-speed Gigabit Ethernet (GigE) interface and optional external x4 PCI Express (PCIe) port.

The FPGA section of the PicoSDR is based on the Virtex-6 family The PicoSDR used in CITI Lab is equipped with a Virtex-6 SX315T FPGA device and has the following specifications:

- 4 GB SODIMM DDR3,
- 18 MB QDR2 SRAM,
- 64 MB NOR Flash,
- 128 MB DDR3 SRAM Dedicated for the Nutaq Central Communication Engine (CCE) Linux application on MicroBlaze.

The radio section is based on the Nutaq Radio420x FMC module and is equipped with four multimode, multiband RF transceivers that support operation between 0.3 GHz and 3.8 GHz, time division duplex (TDD) or frequency division duplex (FDD). The selectable bandwidth ranges from 1.5 to 28 MHz.

PicoSDR 4x4 specifications are summarized in Tab. 11.1. Fig. 11.5 shows the PicoSDR 4x4 schematic [134], while Fig. 11.6 shows 2 images of the PicoSDR 4x4 used in CITI Laboratory with 4 Aaronia OmniLOG 70600 omni-directional antennas.

RF channels	**MIMO**	**RF coverage**	**RF bandwidth**	**Host interface**	**Host throughput**	**FPGA**
4x TRX	4x4	0.3 - 3.8 GHz	1.5 - 28 MHz	1x GigE 6.4 Gbps	900 Mbps (opt. 4x PCIe)	2x Virtex-6

Table 11.1: Nutaq PicoSDR 4x4 specifications.

11.2.1 PicoSDR and GNU Radio

During the research activity at the CITI Laboratory in France, a large amount of time has been dedicated on the utilization of the Nutaq PicoSDR with the GNU Radio platform and the exploitation of its MIMO 4x4 features.

The Nutaq's Advanced Development Platform (ADP) 6.6 Software Suite includes a GNU Radio plugin. The plugin is basically a GNU Radio module called *gr-nutaq* and includes the following blocks:

- *Carrier Perseus Board*: The carrier board specifies the platform's IP address (ID) and must be instantiated as first thing in the GNU Radio environment. Every other *gr-nutaq* block must link to its carrier ID.

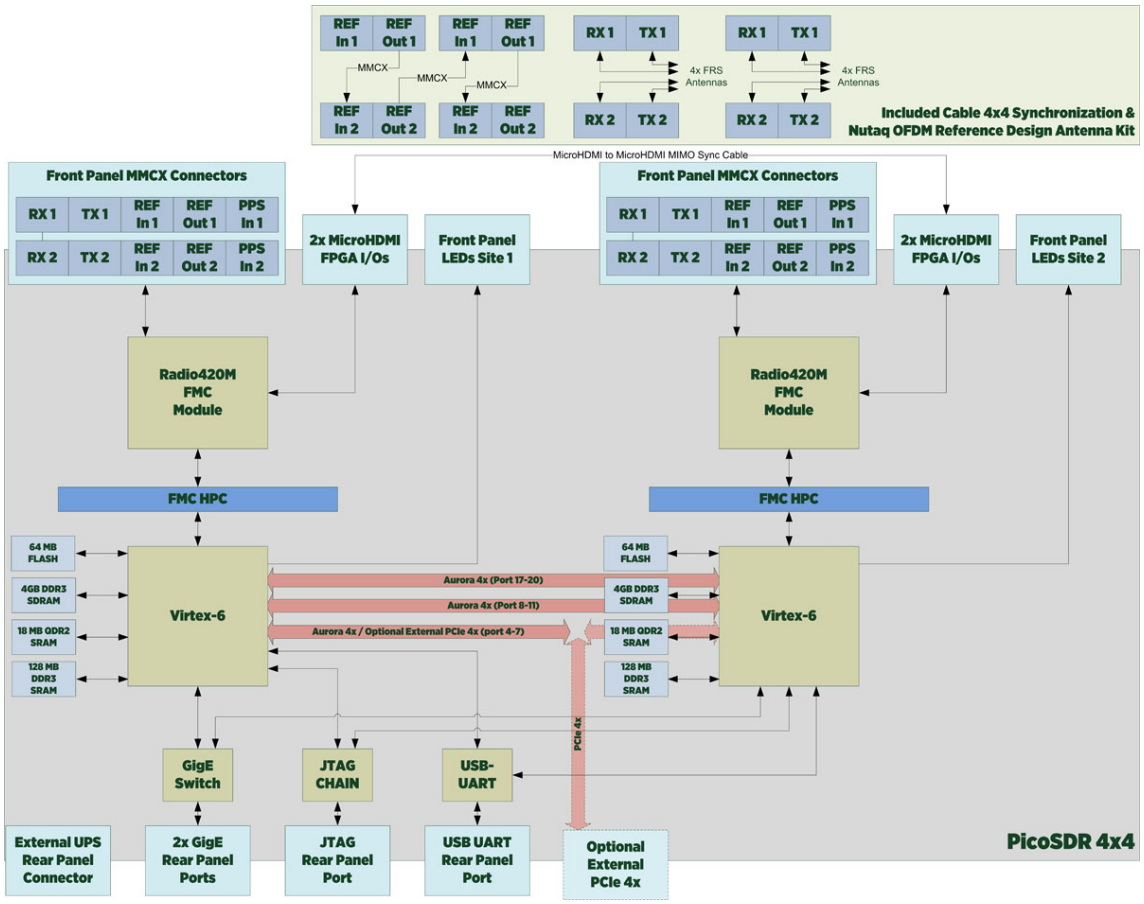


Figure 11.5: Nutaq PicoSDR 4x4 schematic.

Figure 11.6: Nutaq PicoSDR 4x4 with 4 Aaronia OmniLOG 70600 antennas.

- *Radio420 TX/RX*: The TX block configures the transmission path, while the RX the reception path. The blocks are used to configure the different

parameters of the Radio420, like RF frequencies, data rate, analog gains, and filters. In MIMO 2x2 configuration, 2 Radio420 TX and 2 RX blocks must be instantiated.

- *RTDEx Sink/Source*: These blocks allow direct data transfer between the host application and the FPGA design. Each RTDEx channel implementation with its counterpart in the FPGA can be seen as a data pipe with a flow control mechanism. Ethernet or PCIe interface can be selected.

- *Custom register*: These blocks can be used to read and write data within the FPGA design. The write operation can be triggered by a variable or by a data stream. These blocks enable the host application to dynamically control the FPGA.

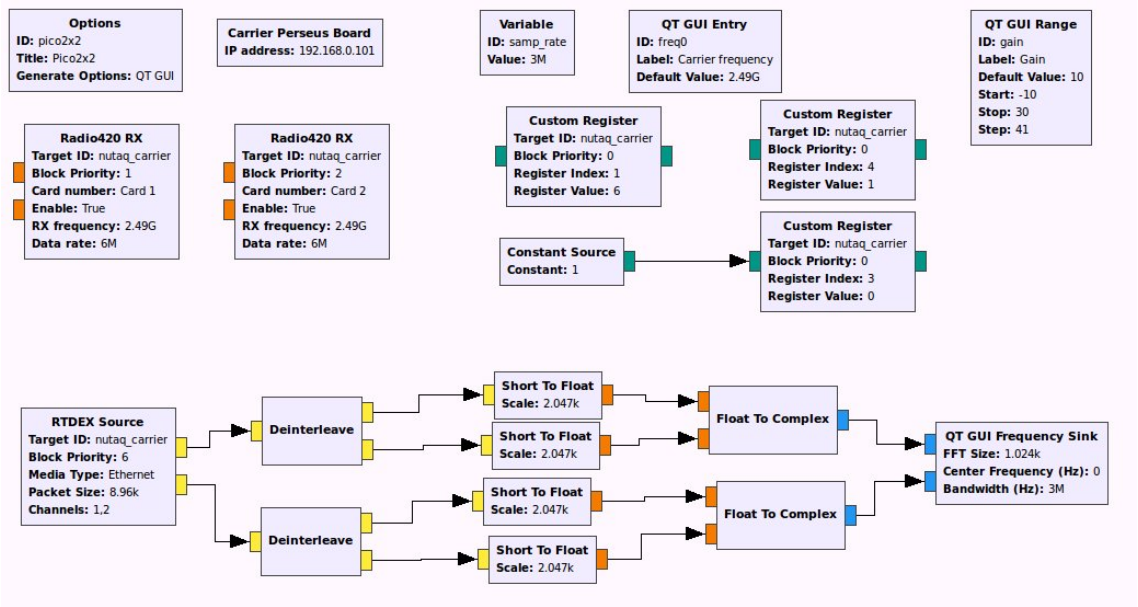


Figure 11.7: Flowgraph for RX section of a synchronized 1x2 SIMO scheme.

In order to enable 2x2 synchronization, the following connections through MMCX cables apply:

1. Rout A1 to Rin A2.

2. Rout A2 to Rin B1.

3. Rout B1 to Rin B2.

First, we show how to set up the blocks in order to implement the RX section of a synchronized 1x2 SIMO scheme. Fig. 11.7 shows the corresponding flowgraph. Two *Radio420 RX* blocks have been instantiated, each one with a different card number, the clock reference of Radio 1 is set to internal and the clock reference of Radio 2 to external. In the FPGA, the RTDEx port width is always 32-bit wide, each I/Q sample is composed of a short integer located in the least significant bit (LSB) for the real part of the signal and a short integer in the most significant bit (MSB) for the imaginary part, hence, deinterliving, short to float and float to complex conversions are needed to correctly represent I/Q samples in GNU Radio, which uses 64 bits (32 for real and 32 for imaginary part) for complex data types.

The PicoSDR 4x4 of CITI Lab uses the `Radio420_GigE_6_6_0_sx315.bit` bitstream (SX315T FPGA device) and in order to enable 2x2 synchronization, the following configuration of the *Custom Register* blocks applies:

- Register index 4 is set with value 1.
- Register index 3 is set with value 0 and has a constant value at its input of 1 (*Constant Source* block connected to custom register 3).
- Register index 1 is set with value 6.

The last setting is about priority, which is a parameter of every *gr-nutaq* block. Priorities must be set such that Radio 1 is initialized before Radio 2 and that TX is initialized before RX for each Radio420.

4x4 synchronization

From the previous configuration, the 4x4 synchronization setup is pretty straightforward. Fig. 11.8 shows the flowgraph for RX section of a synchronized 1x4 SIMO scheme and it can be noticed that all *gr-nutaq* blocks have been duplicated. In addition, the following changes are needed:

1. In *Radio420 RX* blocks, Radio 1 is set to have the reference parameter to internal, while all other radios must be set to external.
2. In *Custom Register* blocks, register index 4 is set with value 2 for both instances, instead of 1.

Please note that both configurations will make sure that the phase will not drift between all signals, but there still may be some constant phase difference between signals because each of the Radio PLLs does not lock exactly on the same phase.

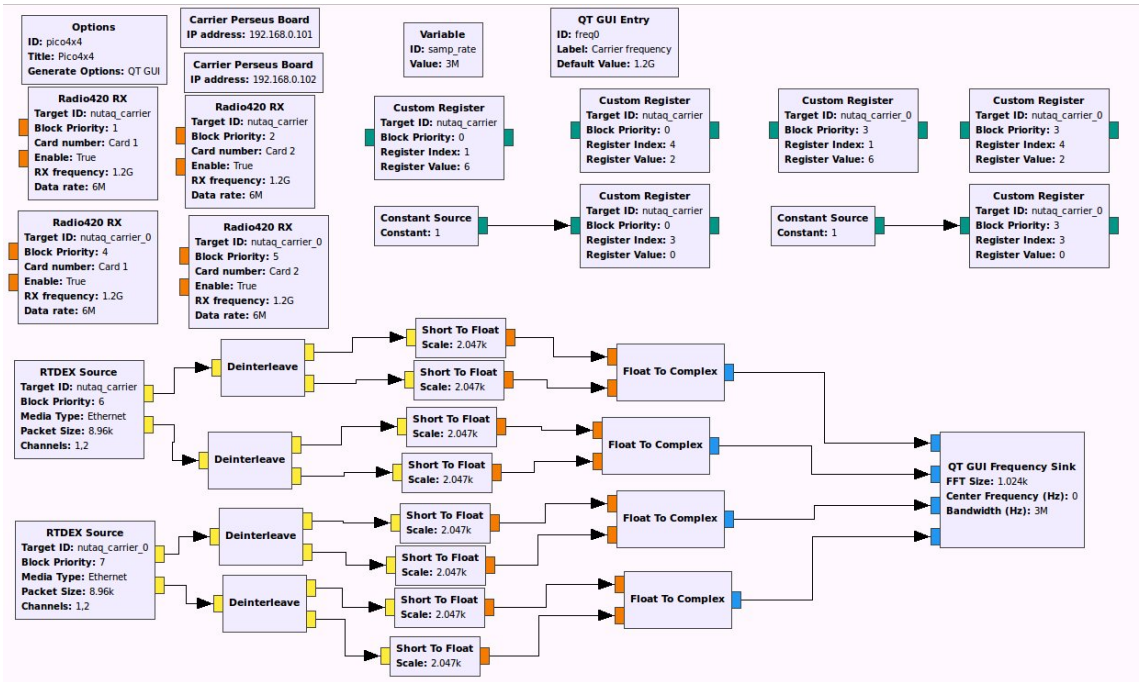


Figure 11.8: Flowgraph for RX section of a synchronized 1x4 SIMO scheme.

11.2.2 PicoSDR 4x4 and eigenvalue-based detection

Considering the MIMO synchronization capability, the PicoSDR 4x4 appears as a promising radio peripheral to build multi-antenna testbeds.

The integration of the PicoSDR with the *eigenbased* block is still at its preliminary stages. The flowgraph in Fig. 11.9 shows the starting point of the realization of an eigenvalue-based spectrum sensing testbed. The results of GLRT testing under $\mathcal{H}_0$ assumption shown in Fig. 11.10, in which "Occupation in %" denotes the false alarm probability. It is evident that the obtained spectrum is not flat when there is no primary transmitted signal and a 50 dB DC component above noise floor is clearly visible. In Fig. 11.11, 4 DC blocking filters were added in the flowgraph, but the false alarm probability of GLRT above 0.5 suggests that the received samples contain several spurious signals due to electronic component imperfections.

Hence, the received signal will need a preprocessing stage or calibration before being examined by spectrum sensing algorithms. Future research will be devoted on the calibration of the SDR peripheral by following the methodology applied in [135], based on [136].

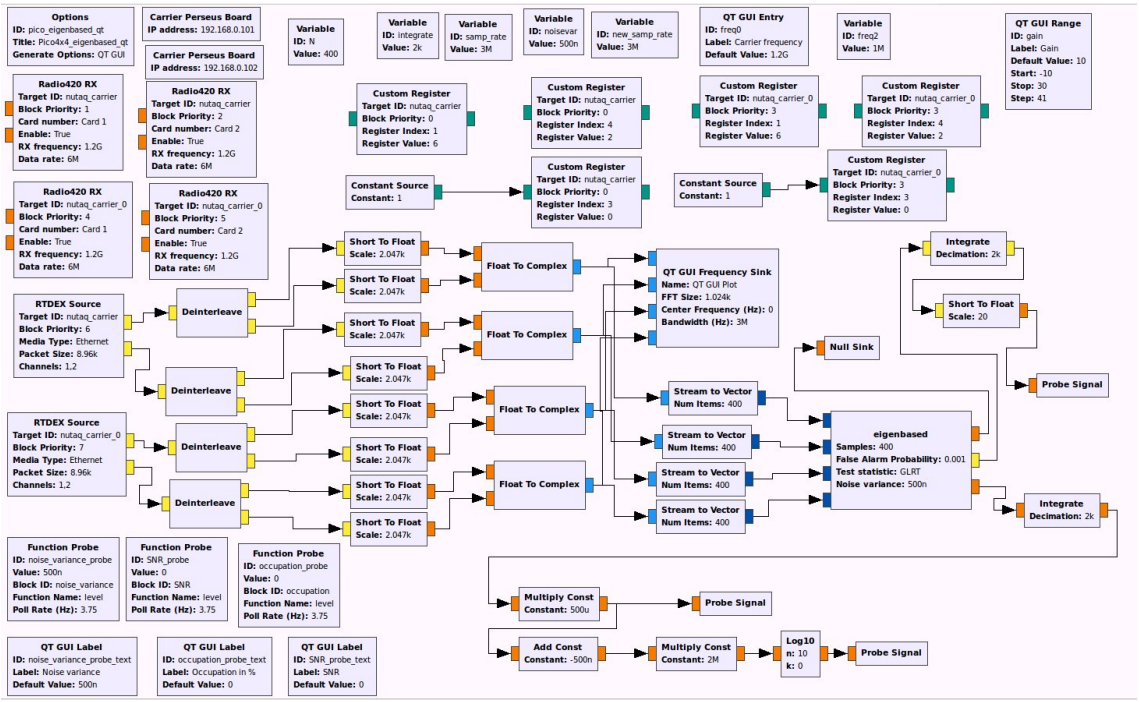


Figure 11.9: Flowgraph with PicoSDR 4x4 for EBD testbed.

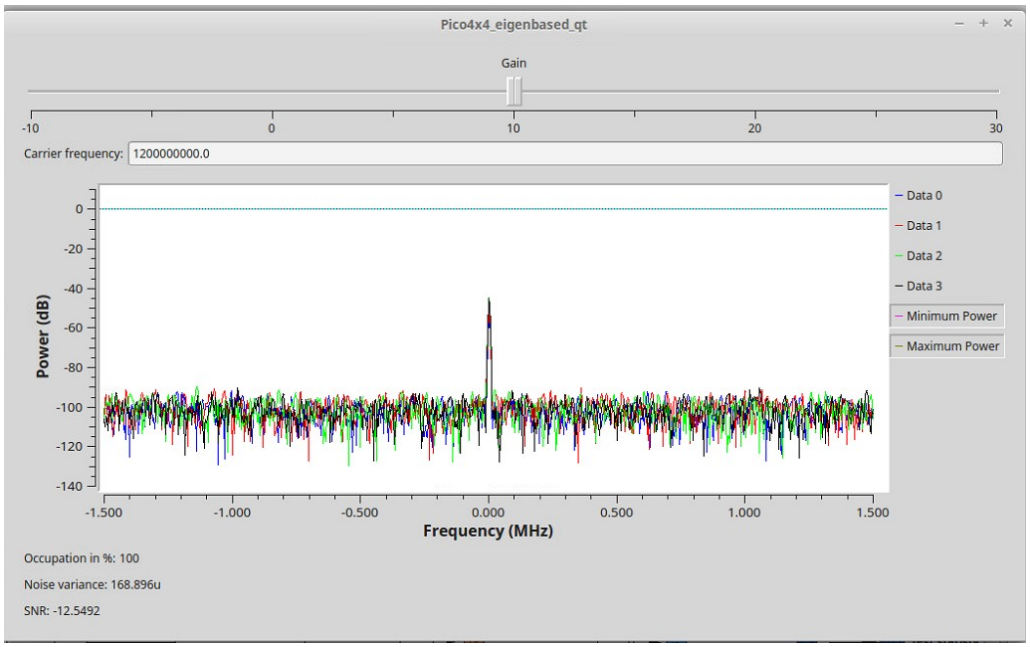


Figure 11.10: PicoSDR 4x4 spectrum and GLRT P_{fa} with no PU signal.

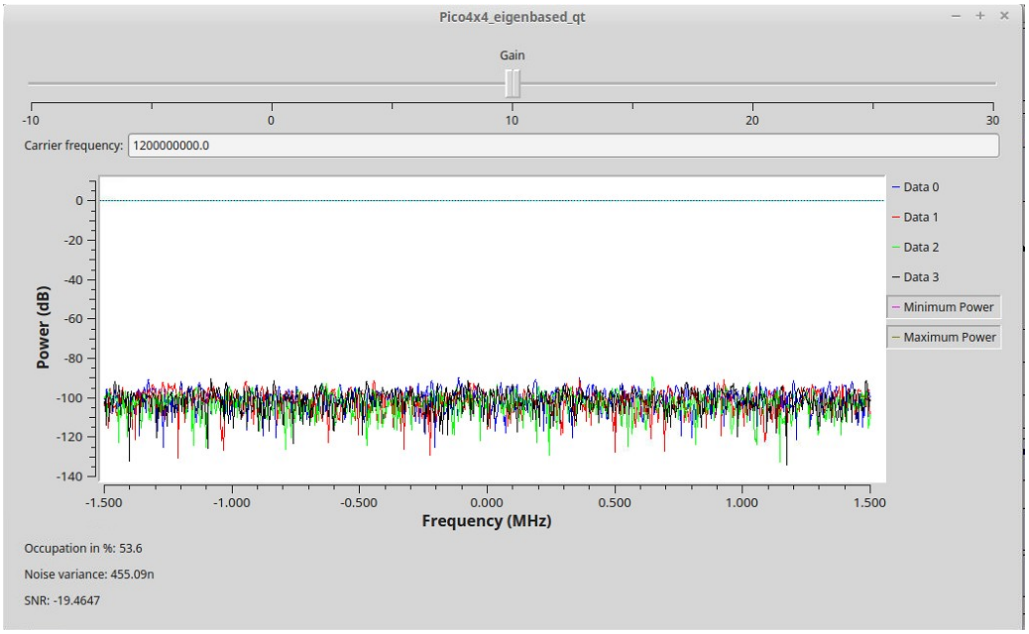


Figure 11.11: PicoSDR 4x4 spectrum and GLRT P_{fa} with no PU signal and DC blocker.

Chapter 12

Conclusion and future work

In this thesis, several spectrum sensing techniques for cognitive radio applications have been presented. Their performance have been evaluated under different scenarios and the SDR implementation in the GNU Radio platform of a selection of these techniques has been illustrated.

In Part I, we have presented the spectrum sensing problem and the used methodologies. Analytical performance evaluation has been carried out for Energy Detection (ED) and for Eigenvalue-Based Detection (EBD) algorithms, in particular for the Roy's Largest Root Test (RLRT) and the Generalized Likelihood Ratio Test (GLRT). Performance results have been provided in terms of false alarm probability (P_{fa}) and detection probability (P_d).

We have first assessed the performance of single-antenna parametric and multi-antenna ED and EBD non parametric sensing algorithms applied to real DVB-T signals under different channel profiles. The flat fading channel analysis confirms the results obtained by using linear mixture models of Gaussian signals, but under a more realistic multipath channel model, the performance and hierarchy of the algorithms completely changes with respect to the flat-fading case signals.

The impact of noise estimation on semi-blind sensing techniques has been studied. Two different noise estimation models have been provided and the analysis of ED and RLRT has been extended to their hybrid approaches. Analytical expressions for P_d and P_{fa} have been derived for each hybrid detector. The results have shown that the fluctuation of noise variance estimation from nominal value is severe in case of small number of auxiliary slots used for the estimation of noise variance.

We have introduced a novel EBD algorithm, based on the eigenvector of the sample covariance matrix, named the EigenVEctor (EVE) Test. It has been shown that EVE and its hybrid and blind variants are able to outperform every every proposed algorithm and can significantly reduce the gap with the Neyman Pearson (NP) test.

We have presented the SNR Wall problem for ED and extended the analysis to

the multi-antennna case. The noise variance uncertainty bound has been quantified with the analytical derivation for hybrid ED. It has been shown that the noise uncertainty can be reduced by increasing the number of samples used for noise variance estimation, but as the SNR Wall condition becomes more stringent, the number of samples or slots used for noise estimation exponentially increases.

The effect of Primary User (PU) traffic on the performance of RLRT has been studied. A realistic and simple PU traffic model has been considered, based only on the discrete time distribution of PU free and busy periods. An analytical evaluation of the spectrum sensing performance under the considered scenario has been carried out. It has been observed that the performance gain due to multiple antennas in the sensing unit is significantly suppressed by the effect of the PU traffic when PU traffic transitions occur more frequently.

In Part II, we have presented the implementation of spectrum sensing algorithm in a *software-defined radio* platform. The SDR concept has been introduced and a general overview on SDR testbeds has been given. We have focused on the GNU Radio software platform and details on how to design and write custom application in GNU Radio have been given.

The implementation of the parallel Energy Detector has been presented. We have illustrated the used technology and details on the necessary custom applications in GNU Radio. The whole project was realized in collaboration with CSP - ICT Innovation,

Finally, we have given details on the GNU Radio implementation of the eigenvalue-based (RLRT and GLRT) detector. Preliminary multi-antenna testbeds with SDR peripherals have been presented.

In conclusion, this work has produced a significant amount of theoretical and algorithmic results, moreover, the SDR implementation offers a set of tools that allow the creation of a realistic cognitive radio system with real-time spectrum sensing capabilities.

Future work will focus on the GNU Radio implementation of the EVE test and on the application of a spur cancellation model to SDR peripherals in order to validate the simulated performance of multi-antenna detection algorithms.

We will also focus our research on the mathematical modelling of a multi-antenna detector which takes into account non-uniform noise variances at different antennas and the performance of EBD algorithms will be also evaluated with correlated noise variance.

Bibliography

[1] Roke Manor Research Ltd, "The UK Frequency Allocations", 2007. Online: http://www.roke.co.uk/resources/datasheets/UK-Frequency-Allocations.pdf. [Accessed April 1, 2016]

[2] Electronic Communications Committee (ECC), 'The European table of frequency allocations and applications in the frequency range 8.3 kHz to 3000 GHz", ERC Report 25, 2015. Online: http://www.erodocdb.dk/Docs/doc98/official/pdf/ercrep025.pdf. [Accessed April 1, 2016]

[3] Dominique Noguet et al. ,"Sensing techniques for Cognitive Radio - State of the art and trends - A White Paper -", *IEEE SCC 41*, Apr. 2009.

[4] M. A. McHenry, "NSF Spectrum Occupancy Measurements Project Summary", shared spectrum co. report, Aug. 2005.

[5] J. Mitola, III and G. Q. Maguire, Jr., "Cognitive radio: Making software radios more personal", *IEEE Personal Communications*, 1999.

[6] Federal Communications Commission, "Notice of proposed rule making and order: Facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technology", ET Docket No. 03-108, Feb. 2005.

[7] IEEE-USA "Improving Spectrum Usage through Cognitve Radio Technology", IEEE USA Position, Nov. 13, 2003.

[8] Tevfik Yücek, Hüseyin Arslan, "A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications", *IEEE Communications Surveys and Tutorials*, vol. 11, n. 1, 2009.

[9] Visa Koivunen, Sachin Chaudhari, Jussi Ryynänen, Marko Kosunen, Erik Larsson, Danyo Danev, "Spectrum Sensing Algorithm Evaluation", *FP7 ICT SENDORA (SEnsor Network for Dynamic and cOgnitive Radio Access) project*", 2010.

[10] Urkowitz, Harry, "Energy detection of unknown deterministic signals", *Proceedings of the IEEE*, vol. 55, no. 4, pp. 523-531, Apr. 1967.

[11] Cabric, D.; Tkachenko, A.; Brodersen, R.W., "Spectrum Sensing Measurements of Pilot, Energy, and Collaborative Detection", *IEEE Military Communications Conference (MILCOM) 2006*, pp. 1-7, Oct. 2006.

[12] Jian Chen; Gibson, A.; Zafar, J., "Cyclostationary spectrum detection in

cognitive radios", *Cognitive Radio and Software Defined Radios: Technologies and Techniques, 2008 IET Seminar on*, pp. 1-5, Sep. 2008.

[13] J. Mitola, "Software radios-survey, critical evaluation and future directions",*Telesystems Conference, 1992. NTC-92., National*, pp. 13/15-13/23, May 1992.

[14] GNU Radio, the free & open software radio ecosystem. Online: http://gnuradio.org. [Accessed April 1, 2016]

[15] A. Nafkha, M. Naoues, K. Cichon,A. Kliks, "Experimental spectrum sensing measurements using USRP Software Radio platform and GNU-radio", *Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM), 2014 9th International Conference on*, pp. 429-434, Jun. 2014.

[16] L. Sanabria-Russo, J. Barcelo, A. Domingo, B. Bellalta, "Spectrum Sensing with USRP-E110", *5th International Workshop on Multiple Access Communications*, Nov. 2012.

[17] S. Haykin, D. J. Thomson, and J. H. Reed, "Spectrum sensing for cognitive radio", *Proceedings of the IEEE*, Vol. 97, No. 5, pp. 849-877, May 2009.

[18] IEEE Standard for Information technology - Local and metropolitan area networks – Specific requirements – Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Policies and procedures for operation in the TV Bands, *IEEE Std 802.22-2011*, pp. 1-680, Jul. 2011.

[19] Yonghong Zeng; Ying-Chang Liang, "Spectrum-Sensing Algorithms for Cognitive Radio Based on Statistical Covariances", *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 4, pp. 1804-1815, May 2009.

[20] Ariananda, D.D.; Lakshmanan, M. K.; Nikookar, H., "A survey on spectrum sensing techniques for cognitive radio", *Cognitive Radio and Advanced Spectrum Management, 2009. CogART 2009. Second International Workshop on*, pp. 74-79, May 2009.

[21] J. G. Proakis, *Digital Communications 4th Edition*, McGraw-Hill, 2001.

[22] Digham, F.F.; Alouini, M.-S.; Simon, Marvin K., "On the energy detection of unknown signals over fading channels", *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 5, pp. 3575-3579, May 2003.

[23] Digham, F.F.; Alouini, M.-S.; Simon, Marvin K., "On the Energy Detection of Unknown Signals Over Fading Channels", *Communications, IEEE Transactions on*, vol. 55, no. 1, pp. 21-24, Jan. 2007.

[24] Kostylev, V.I., "Energy detection of a signal with random amplitude", *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 3, pp. 1606-1610, 2002.

[25] Tandra, R.; Sahai, A., "SNR Walls for Signal Detection", *Selected Topics in Signal Processing, IEEE Journal of*, vol. 2, no. 1, pp. 4-17, Feb. 2008.

[26] Tandra, R.; Sahai, A., "Noise Calibration, Delay Coherence and SNR Walls for Signal Detection", *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, pp. 1-11, Oct. 2008.

[27] Mariani, A.; Giorgetti, A.; Chiani, M., "Effects of Noise Power Estimation on Energy Detection for Cognitive Radio Applications", *Communications, IEEE Transactions on*, vol. 59, no. 12, pp. 3410-3420, Dec. 2011.

[28] Mariani, A.; Giorgetti, A.; Chiani, M., "SNR Wall for Energy Detection with Noise Power Estimation", *Communications (ICC), 2011 IEEE International Conference on*, pp. 1-6, Jun. 2011.

[29] Khalaf, Z.; Nafkha, A.; Palicot, J., "Enhanced hybrid spectrum sensing architecture for cognitive radio equipment", *General Assembly and Scientific Symposium, 2011 XXXth URSI*, pp. 1-4, Aug. 2011.

[30] Moghimi, F.; Schober, R.; Mallik, R.K., "Hybrid Coherent/Energy Detection for Cognitive Radio Networks", *Wireless Communications, IEEE Transactions on*, vol. 10, no. 5, pp. 1594-1605, May 2011.

[31] Badrinath, S.; Reddy, V.U., "A hybrid energy detection approach to spectrum sensing", *Cognitive Wireless Systems (UKIWCWS), 2009 First UK-India International Workshop on*, pp. 1-6, Dec. 2009.

[32] Sequeira, S.; Mahajan, R.R.; Spasojevic, P., "On the noise power estimation in the presence of the signal for energy-based sensing", *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pp. 1-5, May 2012.

[33] Yonghong Zeng; Ying-Chang Liang; Anh Tuan Hoang; Peh, E.C.Y., "Reliability of Spectrum Sensing Under Noise and Interference Uncertainty", *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pp. 1-5, Jun. 2009.

[34] Wei Lin; Qinyu Zhang, "A design of energy detector in cognitive radio under noise uncertainty",, *Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on*, pp. 213-217, Nov. 2008.

[35] Gismalla, E.H.; Alsusa, E., "On the Performance of Energy Detection Using Bartlett's Estimate for Spectrum Sensing in Cognitive Radio Systems", *Signal Processing, IEEE Transactions on*, vol. 60, no. 7, pp. 3394-3404, Jul. 2012.

[36] Zhang, Q.T., "Advanced Detection Techniques for Cognitive Radio", *Communications, 2009. ICC '09. IEEE International Conference on*, pp. 1-5, Jun. 2009.

[37] Ramirez, D.; Vazquez-Vilar, G.; Lopez-Valcarce, R.; Via, J.; Santamaria, I., "Multiantenna detection under noise uncertainty and primary user's spatial structure", *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pp. 2948-2951, May 2011.

[38] Tugnait, J.K., "On Multiple Antenna Spectrum Sensing Under Noise Variance Uncertainty and Flat Fading", *Signal Processing, IEEE Transactions on*, vol. 60, no. 4, pp. 1823-1832, Apr. 2012.

[39] Yonghong Zeng; Ying-Chang Liang, "Eigenvalue-based spectrum sensing algorithms for cognitive radio", *Communications, IEEE Transactions on*, vol. 57, no. 6, pp. 1784-1793, Jun. 2009.

[40] Mahram, A.; Amirani, M.C., "The Complexity as a Criterion for Blind Spectrum Sensing in Cognitive Radio", *Developments in E-systems Engineering (DeSE)*, pp. 444-447, Dec. 2011.

[41] Rui Wang; Meixia Tao, "Blind Spectrum Sensing by Information Theoretic Criteria for Cognitive Radios", *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 8, pp. 3806-3817, Oct. 2010.

[42] S. N. Roy, "On a Heuristic Method of Test Construction and its use in Multivariate Analysis", *The Annals of Mathematical Statistics*, vol. 24, no. 2, pp. 220-238, 1953.

[43] Yonghong Zeng; Ying-Chang Liang, "Maximum-Minimum Eigenvalue Detection for Cognitive Radio", *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pp. 1-5, Sept. 2007.

[44] Penna, F.; Garello, R.; Spirito, M.A., "Probability of Missed Detection in Eigenvalue Ratio Spectrum Sensing", *Wireless and Mobile Computing, Networking and Communications, 2009. WIMOB 2009. IEEE International Conference on*, pp. 117-122, Oct. 2009.

[45] Bianchi, P.; Debbah, M.; Maida, M.; Najim, J., "Performance of Statistical Tests for Single-Source Detection Using Random Matrix Theory", *Information Theory, IEEE Transactions on*, vol. 57, no. 4, pp. 2400-2419, Apr. 2011.

[46] Nadler, B.; Penna, F.; Garello, R., "Performance of Eigenvalue-Based Signal Detectors with Known and Unknown Noise Level", *Communications (ICC), 2011 IEEE International Conference on*, pp. 1-5, Jun. 2011.

[47] Kortun, A.; Ratnarajah, T.; Sellathurai, M.; Ying-Chang Liang; Yonghong Zeng, "Throughput analysis using eigenvalue based spectrum sensing under noise uncertainty", *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pp. 395-400, Aug. 2012.

[48] H. Khaleel, F. Penna, C. Pastrone, R. Tomasi, M. A. Spirito, "Frequency Agile Wireless Sensor Networks: Design and Implementation", *IEEE Sensors Journal*, vol. 12, no. 5, pp. 1599-1608, May 2012.

[49] H. Khaleel, C. Pastrone, F. Penna, M. A. Spirito and R. Garello, "Impact of Wi-Fi traffic on the IEEE 802.15.4 channels occupation in indoor environments", *Electromagnetics in Advanced Applications, 2009. ICEAA '09. International Conference on*, pp. 1042-1045, Sep. 2009.

[50] F. Penna, R. Garello, "Theoretical Performance Analysis of Eigenvalue-based Detection", arXiv:0907.1523v2 [cs.IT], 2009. Online: http://arxiv.org/pdf/0907.1523v2.pdf. [Accessed April 1, 2016]

[51] J. Neyman, E. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses", *Philosophical Transactions of the Royal Society of London*, vol. 231, pp. 289-337, 1933.

[52] Edell, John D. "Wideband, noncoherent, frequency-hopped waveforms and their hybrids in low-probability-of-intercept communications", *No. NRL-8025. NAVAL RESEARCH LAB WASHINGTON DC*, 1976.

[53] D.J. Torrieri, *Principles of Secure Communication Systems*, Artech House, 1992.

[54] Danijela Cabric, Artem Tkachenko, and Robert W. Brodersen, "Experimental study of spectrum sensing based on energy detection and network cooperation", *Proceedings of the first international workshop on Technology and policy for accessing spectrum (TAPAS '06)*, n. 12, 2006.

[55] S. Ciftci and M. Torlak, "A Comparison of Energy Detectability Models for Spectrum Sensing", *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008*, pp. 1-5, 2008.

[56] J. Wishart "The generalized product moment distribution in samples from a normal multivariate population", *Biometrika*, vol. 20A, pp. 32-52, 1928.

[57] V. A. Marchenko and L. A. Pastur, "Distribution of eigenvalues for some sets of random matrices", *Math USSR-Sbornik*, vol .1, pp. 457-483, 1967.

[58] Z. D. Bai, "Methodologies in spectral analysis of large-dimensional random matrices, a review", *Statistica Sinica*, vol. 9, pp. 611-677, 1999.

[59] K. Johansson, "Shape fluctuations and random matrices", *Communications in Mathematical Physics*, vol. 209, no. 12, pp. 437–476, Feb. 2000.

[60] Iain M. Johnstone, "On the distribution of the largest eigenvalue in principal components analysis", *The Annals of statistics*, vol. 29, no. 2, pp. 295-327, 2001.

[61] A. Soshnikov, "A note on universality of the distribution of the largest eigenvalues in certain sample covariance matrices", *Journal of Statistical Physics*, vol. 108, no. 5-6, pp. 1033–1056, Sep. 2002.

[62] S. Péché, "Universality results for largest eigenvalues of some sample covariance matrix ensembles", *Probability Theory and Related Fields*, vol. 143, no. 3, pp. 481-516, Mar. 2009.

[63] C. Tracy and H. Widom, "On orthogonal and symplectic matrix ensembles", *Communications in Mathematical Physics*, vol. 177, no. 3, pp. 727-754, Apr. 1996.

[64] Jinho Baik and Jack W. Silverstein, "Eigenvalues of large sample covariance matrices of spiked population models", *Journal of Multivariate Analysis*, vol. 97, no. 6, pp. 1382-1408, Jul. 2006.

[65] J. Baik, G. Ben Arous, G. and S. Péché, "Phase transition of the largest eigenvalue for non-null complex sample covariance matrices", *The Annals of Probability*, vol. 33 no. 5, pp. 1643-1697, 2005.

[66] D. Féral, S. Péché, "The largest eigenvalues of sample covariance matrices for a spiked population: diagonal case", *Journal of Mathematical Physics*, vol. 50, no. 7, 2009.

[67] L. Wei, O. Tirkkonen, "Cooperative spectrum sensing of OFDM signals using largest eigenvalue distributions", *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, pp. 2295-2299, Sep. 2009.

[68] S. Kritchman, B. Nadler, "Non-Parametric Detections of the Number of Signals: Hypothesis Testing and Random Matrix Theory", *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3930-3941, 2009.

[69] B. Nadler, "On the distribution of the ratio of the largest eigenvalue to the trace of a Wishart matrix", *Journal of Multivariate Analysis*, vol. 102, no. 2, pp. 363-371, Feb. 2011.

[70] P. Bianchi, J. Najim, G. Alfano and M. Debbah, "Asymptotics of eigenbased collaborative sensing", *Information Theory Workshop, 2009. ITW 2009. IEEE*, pp. 515-519, 2009.

[71] European Telecommunications Standards Institute, "ETSI EN 300 744, Digital Video Broadcasting (DVB); framing structure, channel coding and modulation for digital terrestrial television", January 2009.

[72] Floriana Loredana Crespi, Matteo Maglioli, Sergio Benco, Alberto Perotti, "A real-time video broadcasting system based on the GNU Radio-USRP2 platform", *Karlsruhe workshop on software radios*, 2012.

[73] S. Benco, F. Crespi, A. Ghittino, A. Perotti, "Software-defined white space cognitive systems: implementation of the spectrum sensing unit", *The* 2^{nd} *Workshop of COST Action IC0902*, Oct. 2001.

[74] D. Danev, E. Axell, and E. G. Larsson, "Spectrum sensing methods for detection of DVB-T signals in AWGN and fading channels", *Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on*, pp. 2721-2726, Sep. 2010.

[75] Lopez-Valcarce, Vazquez-Villar, Sala, "Multiantenna spectrum sensing for Cognitive Radio: overcoming noise uncertainty", *Cognitive Information Processing (CIP), 2010 2nd International Workshop on*, pp. 310-315, Jun. 2010.

[76] A.M. Tulino and S. Verdú, "Random Matrix Theory and Wireless Communications", *Foundations and Trends in Communications and Information Theory*, vol. 1, no. 1, pp. 1-182, 2004.

[77] G. W. Anderson, A. Guionnet, O. Zeitouni, *An Introduction to Random Matrices*, Cambridge Studies in Advanced Mathematics, Dec. 2009.

[78] European Telecommunications Standards Institute, "ETSI TR 102 377, Digital Video Broadcasting (DVB); DVB-H Implementation Guidelines", Jun. 2009.

[79] W. J. Krzanowski, *Principles of Multivariate Analysis: A User's Perspective*, Oxford University Press, USA, 2000.

[80] Z. Ye, G. Memik, J. Grosspietsch, "Energy Detection Using Estimated Noise Variance for Spectrum Sensing in Cognitive Radio Networks", *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pp. 711-716, Apr. 2008.

[81] R. Garello and Y. Jia, "Improving spectrum sensing performance by using eigenvectors", *COCORA 2011, The First International Conference on Advances in Cognitive Radio*, pp. 26-30, Apr. 2011.

[82] Consultative Committee for Space Data Systems, "TC synchronization and channel coding", *CCSDS 231.0-B-2 Blue Book*, Sep. 2010.

[83] M. Zorzi, A. Gluhak, S. Lange, A. Bassi, "From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view", *IEEE Wireless Communications*, vol. 17, no. 6, pp. 44-51, Dec. 2010.

[84] F. Penna and R. Garello, "Detection of Discontinuous Signals for Cognitive Radio Applications", *IET Communications*, vol. 5, no. 10, pp. 1453-1461, Jan. 2011.

[85] R. Palit, K. Naik, A. Singh, "Anatomy of WiFi access traffic of smart-phones and implications for energy saving techniques", *International Journal of Energy, Information and Communications*, vol. 3, no. 1, Feb. 2012.

[86] A. Ghosh, R. Jana, V. Ramaswami, J. Rowland, N. K. Shankaranarayanan, "Modeling and characterization of large-scale Wi-Fi traffic in public hot-spots", *INFOCOM, 2011 Proceedings IEEE*, pp. 2921-2929, Apr. 2011.

[87] S. L. MacDonald and D. C. Popescu, "Impact of primary user activity on the performance of energy-based spectrum sensing in cognitive radio systems", *Global Communications Conference (GLOBECOM), 2013 IEEE*, pp. 3224-3228, Dec. 2013.

[88] T. Wang, Y. Chen, E. Hines, and B. Zhao, "Analysis of effect of primary user traffic on spectrum sensing performance", *Communications and Networking in China, 2009. ChinaCOM 2009. Fourth International Conference on*, pp. 1-5, Aug. 2005.

[89] J. Y. Wu, P. H. Huang, T. Y. Wang and V. W. S. Wong, "Energy detection based spectrum sensing with random arrival and departure of primary user's signal", *Globecom Workshops (GC Wkshps), 2013 IEEE*, pp. 380-384, Dec. 2013.

[90] L. Tang, Y. Chen, E. L. Hines and M. S. Alouini, "Effect of primary user traffic on sensing-throughput tradeoff for cognitive radios", *IEEE Transactions on Wireless Communications*, vol. 10, no. 4, pp. 1063-1068, Apr. 2011.

[91] H. Pradhan, A. S. Kalamkar and A. Banerjee, "Sensing-throughput tradeoff in cognitive radio with random arrivals and departures of multiple primary users", *IEEE Communications Letters*, vol. 19, no. 3, pp. 415-418, Mar. 2015.

[92] Hwei P. Hsu, *Schaum's outline of theory and problems of probability, random variables, and random processes*, McGraw-Hill, 1997.

[93] Ettus Research. Online: http://www.ettus.com. [Accessed April 1, 2016]

[94] Nutaq (formery Lyrtech). Online: http://www.nutaq.com. [Accessed April 1, 2016]

[95] Per Vices Corporation. Online: http://www.pervices.com/. [Accessed April 1, 2016]

[96] S. Caban, C. Mehlführer, R. Langwieser, A.L. Scholtz, and M. Rupp, "Vienna MIMO testbed", *EURASIP Journal on Advances in Signal Processing*, vol. 2006, pp. 1-13, 2006.

[97] D. Borkowski, L. Brühl, C. Degen, W. Keusgen, G. Alirezaei, F. Geschewski, C. Oikonomopoulos, and B. Rembold, "SABA: A testbed for a real-time MIMO system", *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. 1-15, 2006.

[98] X. Nieto, L.M. Ventura, and A. Mollfulleda, "GEDOMIS: a broadband wireless MIMO-OFDM testbed, design and implementation", *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, Mar. 2006.

[99] Mishra, S.M.; Cabric, D.; Chang, C.; Willkomm, D.; van Schewick, B.; Wolisz, A.; Brodersen, R.W., "A real time cognitive radio testbed for physical and link layer experiments", *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 562-567, Nov. 2005.

[100] ASGARD, a software framework for the development of SDR and Cognitive test beds. Online: http://asgard.lab.es.aau.dk. [Accessed April 1, 2016]

[101] Tavares, F., Tonelli, O., Berardinelli, G., Cattoni, A.F., Mogensen, P.E. (2012). "ASGARD: the Aalborg University Cognitive Radio Software Platform for DSA experimentation", *Proceedings of the 3rd International workshop of COST Action IC0902*, Sep. 2012.

[102] FIT/CorteXlab, Cognitive Radio Testbed. Online: http://www.cortexlab.fr/. [Accessed April 1, 2016]

[103] M. Dardaillon, K. Marquet, T. Risset, and A. Scherrer, "Software Defined Radio architecture survey for cognitive testbeds", *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pp. 189-194, Aug. 2012.

[104] The CREW testbed federation. Online: http://www.crew-project.eu/crewtestbed. [Accessed April 1, 2016]

[105] WARP project: Wireless Open Access Research Platform. Online: http://warpproject.org/trac. [Accessed April 1, 2016]

[106] Microsoft Research Software Radio (Sora). Online: http://research.microsoft.com/en-us/projects/sora/. [Accessed April 1, 2016]

[107] Nutaq PicoSDR Series for Wireless Multi-Standard Prototyping, Online: http://www.nutaq.com/products/picosdr. [Accessed April 1, 2016]

[108] Simplified Wrapper and Interface Generator. Online: http://www.swig.org. [Accessed April 1, 2016]

[109] Python. Online: http://www.python.org. [Accessed April 1, 2016]

[110] GNU Radio, Out-of-tree modules. Online: https://gnuradio.org/redmine/projects/gnuradio/wiki/OutOfTreeModules. [Accessed April 1, 2016]

[111] Gunjan Verma, Paul Yu, "Developing signal processing blocks for software-defined radios", *Adelphi, MD : Army Research Laboratory*, 2012.

[112] CSP - ICT Innovation. Online: http://www.csp.it/. [Accessed April 1, 2016]

[113] ETSI, DMR standard overview. Onlinehttp://www.etsi.org/website/document/technologies/leafl [Accessed April 1, 2016]

[114] Ettus Research, "07495 Ettus USRP1 DS Flyer". Online:https://www.ettus.com/content/files/07495_Ettus_USRP1_DS_Flyer_HR.pdf. [Accessed April 1, 2016]

[115] wxWidgets, Cross-Platform GUI Library. Online: http://www.wxwidgets.org/. [Accessed April 1, 2016]

[116] ZeroMQ, Distributed Messaging. Online: http://zeromq.org/. [Accessed April 1, 2016]

[117] JSON, JavaScript Object Notation. Online: http://json.org/. [Accessed April 1, 2016]

[118] Socket.IO. Online:http://socket.io/. [Accessed April 1, 2016]

[119] Canvas Tutorial. Online: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial. [Accessed April 1, 2016]

[120] Momar Dieng, *Distribution Functions for Edge Eigenvalues in Orthogonal and Symplectic Ensembles: Painlev´e Representations*, PhD dissertation, University of California, Davis, 2005. Online:http://arxiv.org/abs/math/0506586 [Accessed April 1, 2016]

[121] Boaz Nadler, Ratio of Largest Eigenvalue to Trace of a Wishart Matrix, with Matlab Code. Online: http://www.wisdom.weizmann.ac.il/~nadler/Wishart_Ratio_Trace/TW_ratio.html. [Accessed April 1, 2016]

[122] Folkmar Bornemann, "On the numerical evaluation of distributions in random matrix theory", *Markov Processes and Related Fields*, 2010.

[123] W. E. Arnoldi, "The principle of minimized iterations in the solution of the matrix eigenvalue problem", *Quarterly of Applied Mathematics*, vol. 9 , pp. 17-29, 1951.

[124] Mark T. Jones, Merrell L. Patrick, *The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem*, Defense Technical Information Center, 1989.

[125] Jane K. Cullum, Ralph A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol. I: Theory*, SIAM, 2002.

[126] Chengshu Guo, Sanzheng Qiao, "A Stable Lanczos Tridiagonalization of Complex Symmetric Matrices", *Advanced Signal Processing Algorithms, Architectures, and Implementations XV, Proceedings of SPIE*, pp. 1-12, 2005.

[127] J. Kuczynski, H. Wozniakowski, "Estimating the Largest Eigenvalue by the Power and Lanczos Algorithms with a Random Start", *SIAM Journal on Matrix Analysis and Applications*, 1992.

[128] W. Barth, R. S. Martin, J. H. Wilkinson, "Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection", *Numerische Mathematik*, vol. 9, pp. 386-393, 1967.

[129] D. J. Evans, J. Shanehchi, C. C. Rick, "A modified bisection algorithm for the determination of the eigenvalues of a symmetric tridiagonal matrix", *Numerische Mathematik*, vol. 38, pp. 417-419, 1982.

[130] CITI Laboratory, Centre of Innovation in Telecommunications and Integration of service, INSA-Lyon. Online: http://www.citi-lab.fr. [Accessed April 1, 2016]

[131] Ettus Research, "07495 Ettus N200-210 DS Flyer". Online:https://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer_HR_1.pdf. [Accessed April 1, 2016]

[132] Aaronia AG, Radial-Isotropic Ultra Broadband Antenna OmniLOG 70600. Online: http://www.aaronia.com/Datasheets/Antennas/Ultra-Broadband-Antenna-OmniLOG-70600.pdf. [Accessed April 1, 2016]

[133] Ettus Research, "OctoClock-G datasheet". Online:https://www.ettus.com/content/files/Octoclock_Spec_Sheet.pdf. [Accessed April 1, 2016]

[134] Nutaq, "PicoSDR Product Sheet". Online:http://www.nutaq.com/wp-content/uploads/2015/07/PicoSDR_V1.4_02_14_2014_web.pdf. [Accessed April 1, 2016]

[135] J. C. Merlano-Duncan ; T. E. Bogale ; L. B. Le, "SDR Implementation of Spectrum Sensing for Wideband Cognitive Radio", *Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd*, pp. 1-5, Sep. 2015.

[136] Narasimhan, R. and Goldsmith, A., "Spur cancellation", *Google Patents*, US Patent 8,031,101. Oct. 2011. Online: http://www.google.ch/patents/US8031101. [Accessed April 1, 2016]