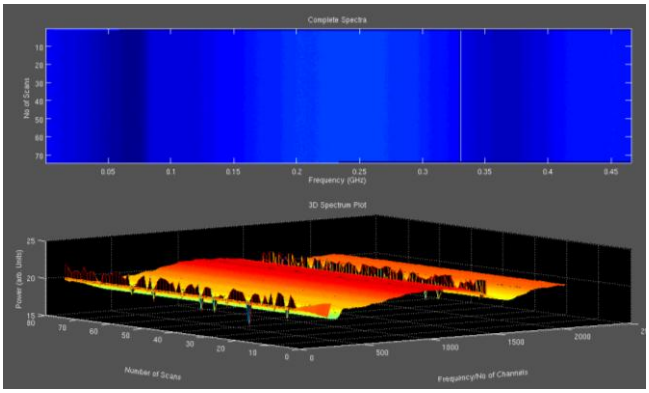
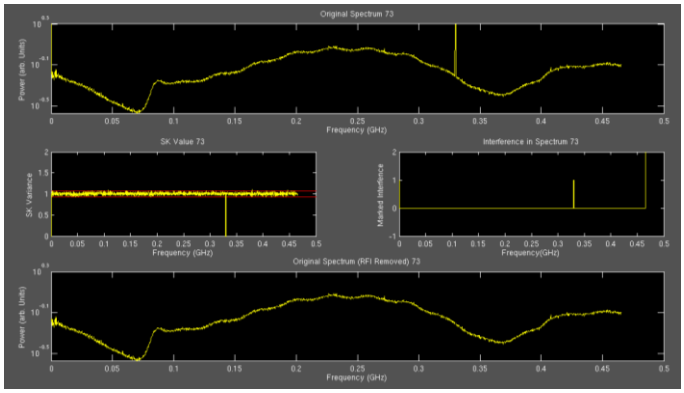
CHALMERS




# FFTS Based RFI Monitoring for Onsala Space Observatory

*Master of Science Thesis (Communication Engineering)*

**Rayyan Ali Qureshi**

Earth and Space Sciences, Onsala Space Observatory,
CHALMERS UNIVERSITY OF TECHNOLOGY,
Göteborg, Sweden, 2010.

**FFTS Based RFI Monitoring for Onsala Space Observatory**
Rayyan Ali Qureshi



Earth and Space Sciences, Onsala Space Observatory
CHALMERS UNIVERSITY OF TECHNOLOGY
SE – 412 96, Göteborg
Sweden
Telephone: +46 (0) 31-772 1000

Cover: Design and implementation of RFI monitoring system based on FFTS and Spectral Kurtosis. See chapter 5 for results and discussion.

Onsala Space Observatory
Göteborg, Sweden

# Abstract

The quality of radio astronomical scientific data can be greatly affected by radio frequency interference (RFI). With the increasing demand for wireless communications, the RFI environment is becoming more and more hostile and as a result, radio astronomical signals are becoming more susceptible to man-made disturbances. It is therefore necessary to find solutions by which RFI in an astronomical signal can be determined and successfully mitigated so that the underlying information can be preserved. For this purpose a software application has been developed in this thesis which can be used to successfully identify RFI in an astronomical signal. This application utilizes a signal processing device called Interconnect Break-out-Board (iBOB) for providing real time power spectral density (PSD) estimates with 2048 (4096 real samples per spectrum) frequency channels of the captured signal. The Spectral Kurtosis Algorithm has been implemented in MATLAB for the analysis of the data captured by the iBOB. This algorithm utilizes the PSD estimates for the calculation and flagging of RFI.

**Keywords:** Radio Frequency Interference, Spectral Kurtosis, Interconnect Break-out-Board

# Acknowledgements

My foremost thanks to Allah, whose kindness, mercifulness and help made it all happen, and without which nothing would have been possible.

I am grateful to my family for their support during my entire study especially in my thesis days. I am very much thankful to my parents whose prayers and support have bought me up till this point of my life.

I am also thankful to my supervisor **Miroslav Pantaleev** and examiner **Rüdiger Haas** for their support during my entire thesis. Special thanks to my second supervisor **Simon Casey** for providing me support and guidance in every single step of software related issues. Thanks to all the people at Onsala Space Observatory for their moral support and for providing good working environment.

# List of Abbreviations

| | |
|---|---|
| ADC | Analog-to-Digital Converter |
| AM | Amplitude Modulation |
| BEE2 | Berkeley Emulation Engine 2 |
| CASPER | Centre for Astronomical Signal Processing and Electronics Research |
| DAC | Digital-to-Analog Converter |
| dB | Decibels |
| DFT | Discrete Fourier Transform |
| FFT | Fast Fourier Transform |
| FM | Frequency Modulation |
| FPGA | Field-programmable Gate Array |
| Gb/s | Giga bits per second |
| GHz | Giga Hertz |
| GS/s | Giga samples per second |
| GUI | Graphical User Interface |
| IAU | International Astronomical Union |
| iBOB | Interconnect Break-out-Board |
| LNA | Low Noise Amplifier |
| LO | Local Oscillator |
| LOFAR | Low Frequency Array |
| MB/s | Megabytes per Second |
| MHz | Mega Hertz |
| MSSGE | MATLAB/Simulink/System Generator/EDK |
| mW | Milli Watt |
| OSO | Onsala Space Observatory |
| PDF | Probability Density Function |
| PFB | Poly Phase Filter Bank |
| PSD | Power Spectral Density |
| RFI | Radio Frequency Interference |
| SK | Spectral Kurtosis |
| SKA | Square Kilometer Array |
| THz | Tera Hertz |
| VLBI | Very Long Baseline Interferometry |
| Wi-MAX | Worldwide Interoperability for Microwave Access |

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1: Radio Astronomy & RFI

## 1.1. Radio Astronomy Receivers

The main subject of radio astronomy is to study the emission of celestial objects within a wide band of radio frequencies. There are basically two types of observations performed, namely total power and spectral line detection (Radio Astronomy n.d.). The later type is of main interest for the current thesis and it is done with heterodyne receivers (Heterodyne n.d.). Figure 1.1.1 shows the basic block diagram of a typical radio astronomical receiver.

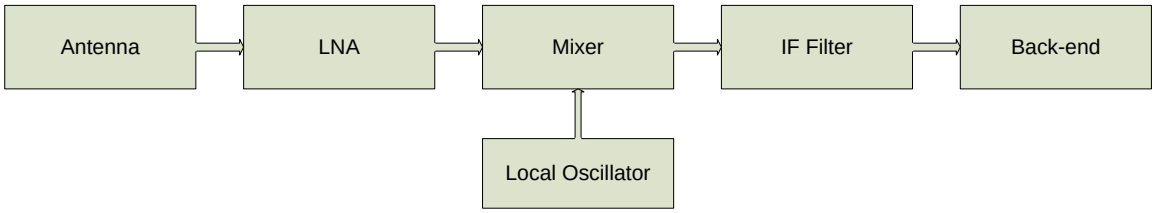



Figure 1.1.1: Basic Block Diagram of a Radio Astronomical Receiver

The antennas are usually very large and the Front-end, consisting of an LNA and mixer, is mounted at the antenna and it is usually cryogenically cooled to increase sensitivity. After passing the signal from the LNA, it is mixed with a frequency tuned by a Local Oscillator. Later on it is passed through different filters, amplified again and then processed by the Back-end to obtain the spectra.

With the advancement of science and technology, radio astronomy is becoming a vast field. The frequency ranges of interest totally depend upon the purpose for which radio astronomy is being used and spans the range from tens of MHz to several THz. The following table shows some of the low-frequency bands of interest for radio astronomical observation.

| Frequencies | Objects of Interest |
|---|---|
| **13.36 - 13.41 MHz** | Solar Observations |
| **25.55 - 25.67 MHz** | Jupiter Observations |
| **73.00 - 74.60 MHz** | Pulsar Detection and Observations |
| **150.05 - 153.00 MHz** | |
| **406.10 - 410.00 MHz** | |
| **1400.0 - 1427.0 MHz** | Hydrogen Line Measurements |

Table 1: Example of Low-Frequency Bands of Interest in Radio Astronomy

The International Astronomical Union (IAU) has defined protected segments of spectra, the complete list of which can be found in (Significant Radio Astronomy Frequencies n.d.). The Low Frequency Array (LOFAR) (About LOFAR n.d.) for example, aims to study the universe in the above mentioned frequency bands. This LOFAR project aims for signal measurements below 250MHz with high sensitivity

(About LOFAR n.d.) (Sensor Fields n.d.). A future project, currently in the design stage, the Square Kilometer Array (SKA) is a radio telescope which will observe the sky covering 70-MHz to 10-GHz band with 50 times more sensitivity than any other radio instrument. The total coverage area of this telescope will be approximately 1 $km^2$ (SKA Home Page n.d.).

## 1.2. RFI Interference

The quality of radio astronomical scientific data, data related to cosmic emissions and celestial objects can be greatly affected by Radio Frequency Interference (RFI). With the increasing demand for wireless communications, the RFI environment is becoming more and more hostile and as a result radio astronomical signals are becoming more susceptible to noise. Typically the bands most affected by RFI are the ones including radio services like FM, mobile phones Wi-Fi and Wi-MAX. A typical example is the Very Long Baseline Interferometry (VLBI) observations in the S-band, which is almost completely occupied by mobile phone bands.

A passive service is a kind of radio service which has naturally occurring, pre-defined spectral lines. Humans have no control over the transmission of these signals and furthermore, these signals have specific power. The received power of a signal is directly proportional to the distance covered and usually it is reduced from 30 to 60 dB below the noise floor of a typical radio astronomical receiver (Jim Cohen 2005). Low noise amplification is done on these received signals before any further processing can be done.

This situation brings the need to find solutions by which RFI can be distinguished from astronomical signals and successfully mitigated so that the underlying information can be preserved. Information on RFI monitoring can be found in detail in chapter 2.

## 1.3. Thesis Motivation

One of the major focuses of radio astronomy is on RFI detection. Every day new research is being done for interference detection in radio astronomical signal. Based on this thought, following are the motivation factors for this thesis.

- Get the knowledge about RFI at OSO.
- Develop a flexible and sensitive FPGA based spectrometer (FFTS).
- Implement an algorithm which can be used on data received from FPGA based spectrometer for RFI detection.

The completion of this thesis will enable astronomers at OSO to easily and effectively identify RFI underlying in a radio astronomical signal. Rather than physically observing each and every signal, the designed application will automatically detect RFI and will mitigate the fractions of band, associated with interference signals.

## 1.4. Thesis Organization

The thesis work consists of two major parts. The first part involves the implementation of an FPGA-based spectrometer. The hardware used for running this spectrometer is known as the Interconnect Break-out-Board (**iBOB)**, which is developed by the Centre for Astronomical Signal Processing and Electronics Research **(CASPER)** (Main Page u.d.)**.** The software of this spectrometer has been implemented in block based programming of MATLAB/Simulink using software libraries from Xilinx and CASPER. This spectrometer provides real time captured data in the form of power and power squared.

The second part of this thesis involves the implementation of the Spectral Kurtosis (**SK)** Algorithm (Gelu M. Nita 2007) (Gary 2010)**.** This algorithm has been implemented in MATLAB and is used for the offline excision of RFI. Multiple plots have been drawn showing original captured signal and signal with marked threshold levels for RFI detection.

Chapter 2 provides an overview to RFI monitoring with a brief introduction to different types of spectrometers along with a detailed explanation of FPGA-based spectrometers, especially the iBOB. Chapter 3 discusses the complete design and implementation of FPGA-based spectrometer based on iBOB. Chapter 4 shows the mathematical implementation of SK algorithm. Chapter 5 provides the results and discussion on the implementation of SK algorithm. Chapter 6 concludes the thesis with some recommendations and future work.

# Chapter 2: RFI Monitoring

## 2.1. Overview

RFI present in an astronomical signal can lead to incorrect interpretation of data. According to the ITU, RFI is defined as "*the effect of unwanted energy due to one or a combination of emissions, radiations, or inductions upon reception in a radio communication system, manifested by any performance degradation, misinterpretation, or loss of information which could be extracted in the absence of such unwanted energy.*" There are several sources of RFI which are categorized as follows.

| Category | RFI Source |
|---|---|
| **Authorized Transmissions** | Licensed Transmissions (FM, AM, TV) |
| **Unauthorized Transmissions** | Wireless Video Camera<br>High Power Cordless Telephones<br>Unlicensed FM Stations Spurious |
| **Intentional** | Unauthorized Jamming |
| **Unintentional** | Spurious Transmissions |

Table 2: Different Sources of RFI

## 2.2. RFI Monitoring Techniques

Over the years, different techniques have been presented to detect the underlying RFI in a radio astronomical signal. These techniques either involve time domain analysis, frequency domain analysis or statistical analysis of the received signal.

### 2.2.1. Time Domain Analysis

This type of analysis consists of the detection and elimination of power peaks in the received signal that are larger than the variance of the measured signal in the absence of RFI (Ellingson, Hampson och Johnson 2003) (Guner, Johnson och Niamsuwan 2007). But since the detected power is the averaged version of the instantaneous one, a peak may pass undetected if the duration of RFI peaks is shorter than the integration time.

### 2.2.2. Frequency Domain Analysis

This type of analysis consists of discarding sub-bands from the received signal spectrum with power higher than the variance of received signal (Johnson 2006).

### 2.2.3.Statistical Analysis

This type of analysis is based on the assumption that an astronomical signal follows the Gaussian distribution with zero-mean in the absence of RFI (Camps 2009). Since it is following a known distribution, the probability density function (PDF) and moments are perfectly known. The Spectral Kurtosis algorithm, which is the main focus of this thesis, is also based on such a process.

## 2.3. RFI Monitoring Instruments

Spectrometers or Spectrum Analyzers are devices used to examine the spectral composition of a received signal. Nowadays spectrum analyzers can measure signals up to 100-GHz and above. Spectrometers can be divided into two categories namely analog or digital depending upon whether they imply band-pass filter/super heterodyne receiver or Digital Fourier Transform (DFT)/Fast Fourier Transform (FFT) phenomenon.

### 2.3.1.Analog Spectrometers

An analog spectrometer uses either a variable band-pass filter whose mid-frequency is automatically tuned through the range of frequencies of which the spectrum is to be measured or a super-heterodyne receiver where the local oscillator is swept through a range of frequencies ( Spectrum Analyzer Fundamentals u.d.). Different types of analog spectrometers available at OSO are Anritsu MS2602A, Agilent E4407B, and Rohde & Schwarz ESMD.

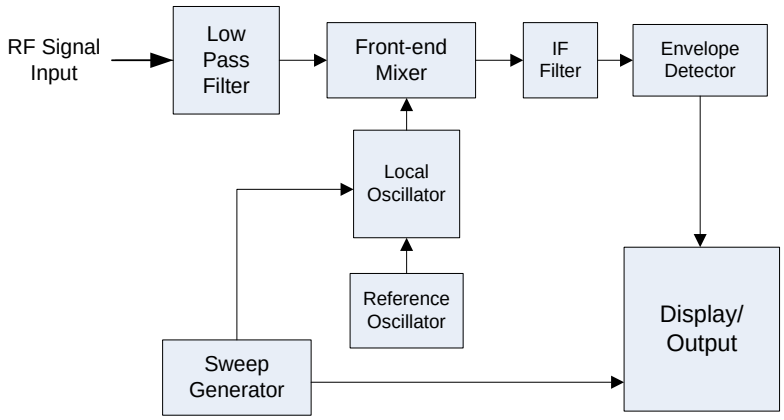



**Figure 2.3.1: Block Diagram of a Classic Superheterodyne Spectrum Analyzer ( Spectrum Analyzer Fundamentals n.d.)**

### 2.3.2.Digital Spectrometers

A digital spectrometer converts the time-domain signal to frequency domain using discrete Fourier transform. The main advantage of these types of spectrometers is acquiring and converting the time domain signal to frequency domain in a single-shot rather than repetitive acquisition though it can also be done. Figure 2.3.2 shows the block diagram of a basic digital spectrometer.


Figure 2.3.2: Block Diagram of a Basic Digital Spectrometer

The first block shows the antenna/telescope that is receiving the astronomical signal. The output of this antenna is connected to an input of the FFTS based device. The signal in this device is first converted to digital form using an analog to digital converter as shown in the second block. Afterwards it is passed from a digital filter and then an FFT is performed. The accumulator block takes care of all the scaling and resolution issues so that relevant spectral information can be maintained within the digital data. The data from the accumulator can then be outputted over Ethernet or any other cable depending upon which one is supported by the spectrometer design.

Digital spectrometers are generally difficult to implement due to the complications in data conversion from the analog to digital domain. Also several others factors are to be considered like data accuracy, resolution and hardware speed. But these spectrometers are very versatile as different functions can be performed just by changing the software of that spectrometer rather than changing the hardware as in case of analog ones. A digital spectrometer has been built by utilizing the FPGA-based processing device available at OSO. This device is known as Interconnect Break-out Board (iBOB)

However, these analyzers do have some limitations in comparison to a super heterodyne spectrum analyzer, particularly in the areas of frequency range, sensitivity, and dynamic range. But the main advantage of digital spectrometers over analog ones is due to the flexibility in their designs. Section 2.3.3 will discuss the advantages and disadvantages of analog and digital spectrometers in detail.

### 2.3.3. Comparison between Analog and Digital Spectrometers

There are several differences on the basis of hardware, processing etc. in analog and digital spectrometers. Table 3 summarizes major differences and gives a comparison between both types of spectrometers.

| Comparison | Analog Spectrometers | Digital Spectrometers |
|---|---|---|
| **Processing Technique** | Uses either variable band-pass filter or super-heterodyne receiver | Uses ADC to convert the signal to digital form and then perform FFT |
| **Hardware Limitations** | Hardware limitations in term of specifications like sweep time, resolution bandwidth, and trace points | With a good resolution ADC, sweep time, resolution bandwidth and trace points can be controlled easily using software |
| **Measurement of Parameters** | Impulse response and interference can be difficult to measure. Different offline techniques are required | Impulse response and interference can be measured straight out-of-box |

| | | |
|---|---|---|
| **Processing Speed** | Extremely fast in the detection of radio astronomical signal | Can experience some latency due to sampling process in ADC and DSP afterwards |
| **Non-linearities and unpredictability** | Different electronic component values are specified with some tolerance for example in resistors, capacitors etc. which could vary with temperature, humidity and time. This could bring non-linearities in the behavior of spectrometer | Since all the calculations are done digitally, most of the coefficients can be stored and controlled from the memory of processing device making it more stable and predictable. Furthermore this reduces the work of calibration |
| **Filters** | Analog filters in these kind of spectrometers are expensive | Digital filter are cheap and versatile. Though they can be very complex to implement, they make it possible to implement designs which are almost impractical using analog filters |

Table 3: Comparison between Analog and Digital Spectrometer

## 2.4. Interconnect Break-out Board (iBOB)

The **Interconnect Break-out Board (iBOB)** is an FPGA-based processing board for digital signal processing. It features two Z-DOK connectors which can provide data rates up to 6.25 Gb/s. These connectors can be attached to a variety of other input/output boards like ADCs and DACs. Despite the high data rate, these connectors have very low cross talk rate about 100 ohms (Z-DOK Overview n.d.).

The iBOB has a Virtex-II Pro 2VP50 FPGA which can support operations up to 10 Gb/s (Xilinx u.d.). The processed data can then be output over 10 Gigabit Ethernet (GbE/XAUI) facilitated by two CX4 connectors on the board. Beside these connectors, iBOB also has a standard 10/100 Mbps Ethernet interface for data output, RS232 port for FPGA control and JTAG connector for FPGA programming. The complete list of peripherals and interfaces can be found at (IBOB n.d.).

Figure 2.4.1 and Figure 2.4.2 shows the upper and front view respectively of the iBOB available at Onsala Space Observatory.

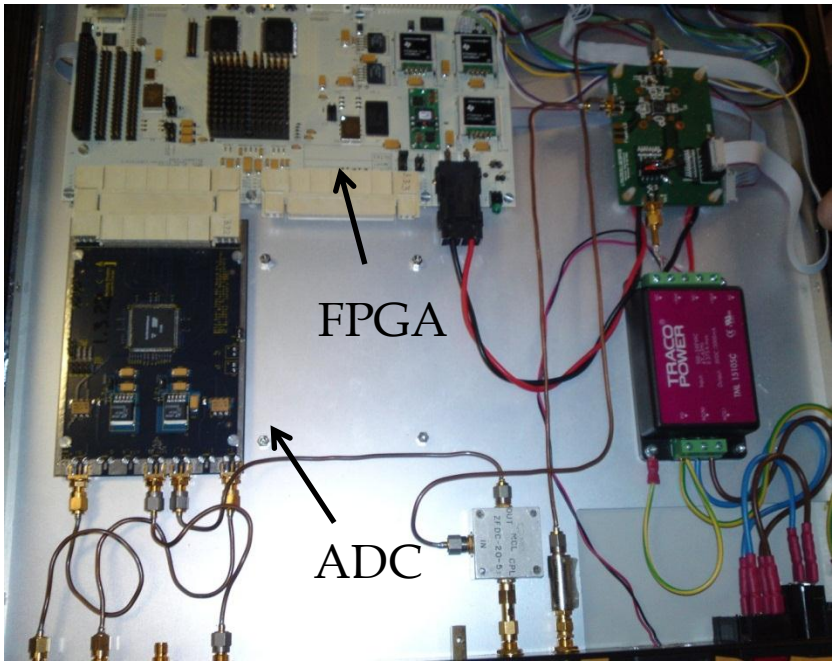



Figure 2.4.1: Upper View of Interconnect Break-out Board (iBOB)


Figure 2.4.2: Front View of Interconnect Break-out Board (iBOB)

# Chapter 3: The iBOB Spectrometer

## 3.1. Introduction

The iBOB spectrometer is based on the design of a Kurtosis Spectrometer (Kurtosis Spectrometer n.d.). The Kurtosis Spectrometer (Spectral Kurtosis Spectrometer) was first built for the Korean Solar Radio Burst Locator (KSRBL) (Dou, et al. 2009). This spectrometer has a bandwidth of 500 MHz at a sampling rate of 1 Giga samples per second (Gs/s). Real time PSD estimates with 2048 (4096 real samples per spectrum) frequency channels along with their power squares are calculated. These power and power squared values are accumulated afterwards and then transmitted over a 100-Mbps Ethernet interface.

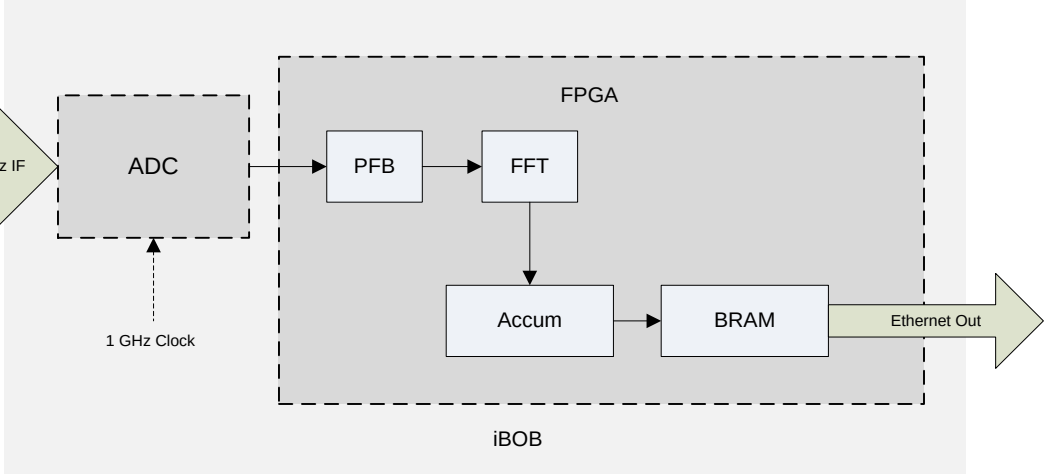



Figure 3.1.1: Basic Block Diagram of iBOB Spectrometer

The complete software design of this spectrometer has been implemented in Simulink using block based programming. Libraries for the software design have been provided by CASPER and Xilinx. These libraries contain different blocks like ADC, FFT, PFB etc. which are used in this design. Later sections in this chapter will provide an explanation on different major blocks of the design. Information only relevant to this particular spectrometer design has been provided. For detailed information of the blocks, references to CASPER and Xilinx documentation have been given where ever needed.

## 3.2. The ADC Block

There are two Analog-to-Digital converters built-in to the iBOB hardware. But this spectrometer design utilizes only one of the two ADCs, which can do sampling at the rate of 1 Gs/s. The clock rate used for 'adc0', the first ADC on the iBOB, is 1-GHz with an output of 8-bit digital data. Figure 3.2.1 shows the block of ADC as used in Simulink design. The masking parameters of this block along with the function values used are shown in Table 4.

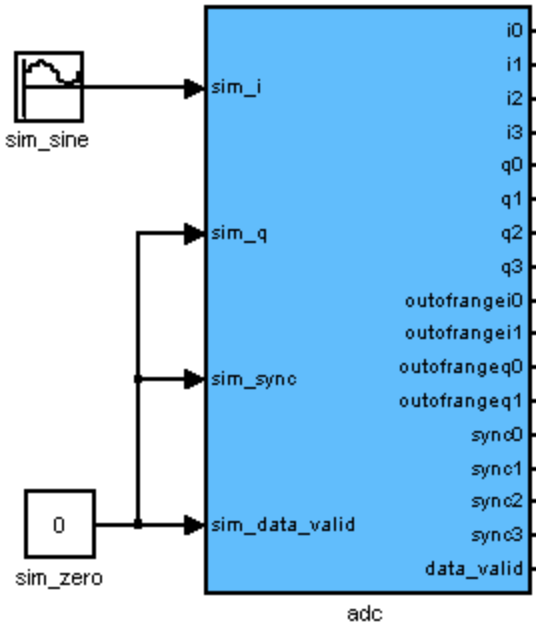



Figure 3.2.1: ADC Block in Simulink

| Parameter | Value (Description) |
|---|---|
| **Sample Period** | 1 |
| **ADC Interleave mode** | 1 (Only one input used) |
| **ADC Clock Rate (MHz)** | 1024 |
| **ADC Board** | adc0 |

Table 4: ADC Block Masking Parameters

The more detailed explanation of input/output ports and hardware related information can be found on the CASPER website at (ADC n.d.).

## 3.3. Poly Phase Filter Bank

The data from the ADC is input to a Poly Phase Filter Bank (PFB) which is followed by a FFT block. The usage of this block along with the FFT block improves the shape of channels within a spectrum by implying longer data windows.

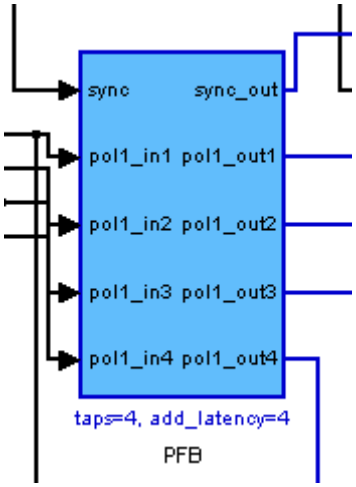



Figure 3.3.1: Polyphase Filter Bank Block in Simulink

The masking parameters of this block along with the function values used are shown in Table 5. More detailed explanation of input/output ports and hardware related information can be found on CASPER website at (Pfb fir real n.d.).

| Parameter | Value (Description) |
|---|---|
| **Size of PFB (2^?)** | 12 |
| **Total Number of Taps** | 4 |
| **Windowing Function** | Hamming |
| **Number of Simultaneous Inputs (2^?)** | 2 (4 inputs to the PFB bank) |
| **Make Biplex** | 0 (No biplex) |
| **Input Bit-width** | 8 (Data coming from ADC) |
| **Output Bit-width** | 18 (Data that is output from PFB) |
| **Coefficient Bit-width** | 8 (Normally equal to input width) |
| **Use Distributed Memory for Coeffs** | 0 (BRAMs are used to hold the coefficients) |
| **Adder Latency** | 4 (Latency through adders in the FFT) |
| **Multiplier Latency** | 6 (Latency through multipliers in the FFT) |
| **BRAM Latency** | 4 (Latency through BRAM in the FFT) |
| **Quantization Behavior** | Truncate (Return the bits specified above) |

Table 5: Polyphase Filter Bank Block Masking Parameters

## 3.4. The FFT Block

This block computes the real-sampled Fast Fourier Transform of the digitized data. The output of this block is half the number of inputs as it only outputs the positive

frequencies and not their mirror images. The data is outputted in an increasing fashion i.e. channel 0, followed by channel 1 and so on, up to channel $2^N - 1$. Figure 3.4.1 shows the FFT block used in Simulink design.

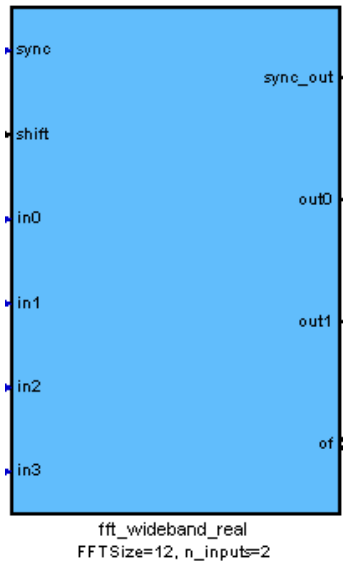



Figure 3.4.1: FFT Block in Simulink

Table 6 shows the values used in masking parameters of this block. Only the values used in design and the variable names are shown in the table. The details of input/output ports along with other information can be found at (Fft wideband real n.d.).

| Variable | Value |
|---|---|
| **Size of FFT** | 12 |
| **Input/output Bit Width** | 18 |
| **Coefficient Bit Width** | 18 |
| **Number of Simultaneous Inputs** | 2 |
| **Quantization** | Truncate |
| **Overflow** | Wrap |
| **Adder Latency** | 4 |
| **Multiplier Latency** | 6 |
| **BRAM Latency** | 4 |

Table 6: FFT Block Masking Parameters

## 3.5. Scaling subsystem

Scaling can be applied to this design as the precision of data needs to be carefully selected in the case of analog to digital conversion. Data that is output from the FFT block has 36-bits of precision. But the value of the scaling register is set to choose the 32-bits that need to be fed to the next block, in order to have ease and convenience in

the calculation and transferring of data. This block has 3 inputs and 2 outputs. 'In1' is the input for the scaling coefficient. 'In2 and 'In3' is the actual data that is to be scaled. Figure 3.5.1 shows the scaling subsystem block in Simulink.

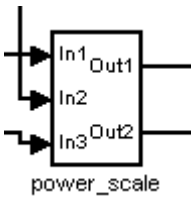



Figure 3.5.1: Scaling subsystem in Simulink

## 3.6. Misc. Blocks

Several other blocks are used in this design in order to provide serialization, concatenation, parallelization etc. The 'cram' block provides the concatenation of input data with a single output. The masking parameter value used for this block is 4. The 'uncram' block takes a concatenated input and slice them up into even pieces. Masking parameters of this block are shown in Table 7. The 'vacc' block is basically a vector accumulator. Masking parameters of this block are shown in Table 8. Figure 3.6.1 shows all the three blocks as connected in Simulink. More details of these blocks can be found at (Block Documentation n.d.).

| Masked Parameter | Value |
|---|---|
| **Slices** | 4 |
| **Slice Width** | 48 |
| **Output Binary Point** | 31 |
| **Output Arithmetic Type** | 0 (Unsigned) |

Table 7: Masked Parameters of 'uncram'

| Masking Parameter | Value |
|---|---|
| **Length of Vector** | 10 |
| **Simultaneous Vectors** | 4 |
| **Maximum Accumulations** | 18 |
| **Arithmetic Type** | 0 |
| **Input Bit Width** | 32 |
| **Input Binary Point** | 31 |
| **Output Bit Width** | 48 |
| **Output Binary Point** | 31 |
| **Add Latency** | 6 |
| **BRAM Latency** | 6 |

Table 8: Masking Parameters of 'vacc'

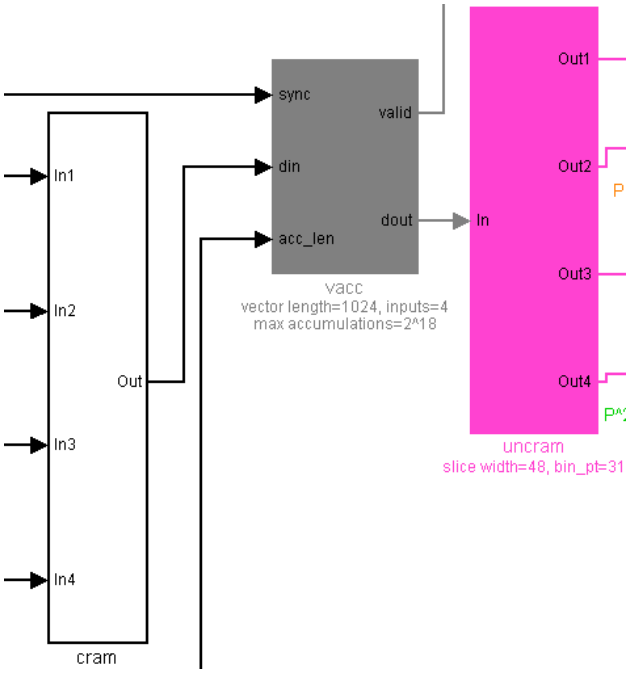



Figure 3.6.1: Misc. Blocks in Simulink

## 3.7. The Complete System

Figure 3.7.1 shows the complete system as implemented in Simulink. Several minor blocks have not been discussed since they are out of the scope of this thesis and their information can be found at (Block Documentation n.d.). Summarizing the design, 1 – 500 MHz RF signal is input to the ADC with a clock rate of 1 GHz. It provides 8-bit samples per output with a total of 4 outputs. These samples are input to the PFB which along with FFT improves the shape of channel within a spectrum. The output of FFT is from N to $2^N - 1$ , where 'N' is the number of channels. The FFTed data is then multiplied with itself to compute the power squares. The input of the multiplier block is 16-bit. Since there are four outputs from the FFT block, a bit select register is required to select which of the 4 outputs with 16-bits is to be inputted to the multiplier. The output of the multiplier is 32-bit. These power squares are accumulated with power values in a vector accumulator, and then transmitted to the BRAM of the device. BRAM transmits this data to fast Ethernet which is connected to the receiving computer.

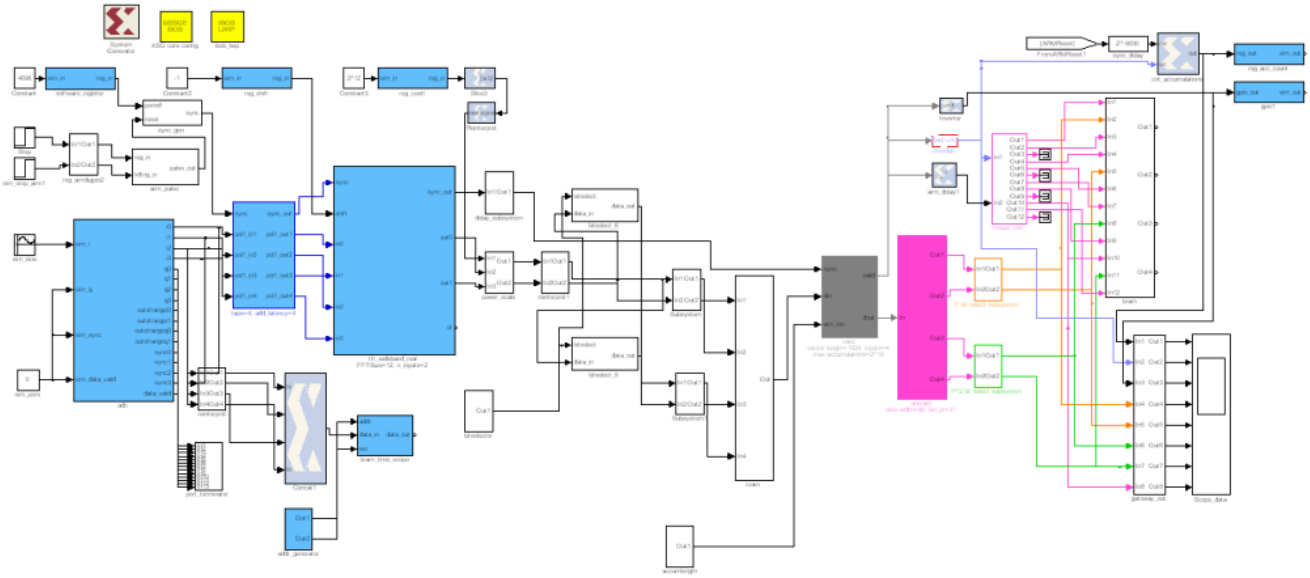


Figure 3.7.1: The Complete System in Simulink[1]

[1] Figure can be viewed at full size in Appendix B

# Chapter 4: The SK Algorithm

## 4.1. Introduction

In the frequency domain, *the spectral kurtosis (SK) of a signal is defined as the kurtosis of its frequency components*. Kurtosis refers to the measure of "peakedness" of the probability distribution of a real valued random variable. In general, spectral kurtosis (SK) is a statistical tool which is used for the detection of non-Gaussian components in a signal. It was first introduced by Dwyer (R. Dwyer 1983) in 1983. This tool not only detects the non-Gaussian component but also their location in the frequency domain. Dwyer initially used it as a complement to power spectral density (PSD) for the detection of randomly occurring transients in (R. F. Dwyer, Use of the kurtosis statistic in the frequency domain as an aid in detecting random signals 1984) and (R. F. Dwyer, A technique for improving detection and estimation of signals contaminated by under ice noise 1983).

## 4.2. Explanation

Discrete Fourier transform is the most common method for obtaining power spectral density (PSD) estimates of a real time signal. Discrete Fourier transform (DFT) is implemented using Fast Fourier transform (FFT) algorithm. The uncertainty of a DFT-based PSD estimates at any given frequency bin is as high as 100% of its computed value (Feng Liu 1989). But this is true only in the case of a stationary Gaussian signal. Time domain signals following other distributions may have a different behavior of their PSD estimates. But this uncertainty, either in frequency or time domain can be significantly reduced using averaging.

Consider a dimensionless quantity $V_k^2$ which is defined as:

$$V_k^2 = \frac{\sigma_k^2}{\mu_k^2} \tag{1.1}$$

The term $V_k^2$ is known as "spectral variability". In Equation (1) $\sigma_k^2$ and $\mu_k$ are the variance and mean respectively and are defined as;

$$\mu_k = \langle \hat{P}_k \rangle \quad \& \quad \sigma_k^2 = \langle \hat{P}_k^2 \rangle - \langle \hat{P}_k \rangle^2 \tag{1.2}$$

Where $\hat{P}_k$ is the non-averaged PSD estimate corresponding to the frequency bin $f_k$. Due to its linearity, the PSD estimate $\hat{P}_k$ is an unbiased estimator of the true spectral power density of the signal. But the parameter defined in Equation (1) is no longer a linear combination of its components. So in order to define the spectral kurtosis estimator, consider the following equation:

$$\hat{V}_k^2 = \frac{\hat{\sigma}_k^2}{\hat{\mu}_k^2} \tag{1.3}$$

Where $\hat{\sigma}_k^2$ and $\hat{\mu}_k$ are unbiased estimators of their associated parameters, $\langle\hat{\sigma}_k^2\rangle = \sigma_k^2$ and $\langle\hat{\mu}_k\rangle = \mu_k$.

In practice the above two estimators will be computed in algorithm from a number of M adjacent blocks of N time-domain samples used to obtain M spectral estimates $\widehat{P}_{ki}(k = 0, \dots, \frac{N}{2}; i = 1, \dots, M)$. Using sums of power and powered squared as;

$$S_1 = \sum_{i=1}^{M} \hat{P}_{ki} \;\; \& \;\; S_2 = \sum_{i=1}^{M} \left(\hat{P}_{ki}\right)^2 \tag{1.4}$$

Now according to (Kendall 1958),

$$\hat{\mu}_k = \frac{1}{M} S_1 \;\; \& \;\; \hat{\sigma}_k^2 = \frac{MS_2 - S_1^2}{M(M-1)} \tag{1.5}$$

This leads us to,

$$\hat{V}_k^2 = \frac{M}{M-1}\left(M\frac{S_2}{S_1^2} - 1\right) \tag{1.6}$$

The above equation completely defines the SK estimator in terms of power and powered square. A more generalized form of equation (1.6) which includes the duty cycle "d" can be given as;

$$\hat{V}_k^2 = \frac{MNd}{M-1}\left(M\frac{S_2}{S_1^2} - 1\right) \tag{1.7}$$

Where, "N" in Equation (1.7) is the number of samples. More detailed explanation of Equations (1.6) and (1.7) can be found in (Gelu M. Nita 2007) and (Gary 2010).

## 4.3. Threshold Calculation

In order to estimate the thresholds to identify the radio frequency interference (RFI) deviations, an infinite number of moments are needed which are calculated analytically. However it was experimentally proved in (Gary 2010) that only the first four statistical moments are needed to accurately estimate the thresholds.

Using the standard notations as described in (Gary 2010), $\mu_1' = E\left(\hat{V}_k\right) \equiv 1$ and $\mu_n = E\left[(\hat{V}_k - \mu_1')^n\right]$ the mean and first central moments of $\hat{V}_k$ are;

$$\begin{aligned} \mu_1' &= 1 \\ \mu_2 &= \frac{2Nd(Nd+1)M^2\Gamma(MNd+2)}{(M-1)\Gamma(MNd+4)} \\ \mu_3 &= \frac{8Nd(Nd+1)M^3\Gamma(MNd+2)}{(M-1)^2\Gamma(MNd+6)} \end{aligned} \tag{1.8}$$

$$\mu_4 = \frac{\times \left((Nd+4)MNd - 5Nd - 2\right)}{\times \left(M^3N^4d^4 + 3M^2N^4d^4 + M^3N^3d^3 + 68M^2N^3d^3 - 93MN^3d^3 + 125M^2N^2d^2 - 245MN^2d^2 + 84N^2d^2 - 32MNd + 48Nd + 24\right)}{(M-1)^3\Gamma(MNd+8)}$$

## 4.4. Graphical User Interface

In order to minimize the user interaction with software related tasks, a graphical user interface (GUI) has been created. This GUI combines the tasks related to the following software commands.

- Capturing data from the iBOB and saving it to the working directory.
- Selecting the saved files or any other iBOB files for the analysis.
- Running the SK algorithm and flagging the corrupted data.
- Doing all the above mentioned tasks in a single click.

### 4.4.1.Working of GUI

Figure 4.4.1 shows a snapshot of the main GUI. It can be seen that there are three 'push' buttons and a check box which can perform the above mentioned tasks easily.

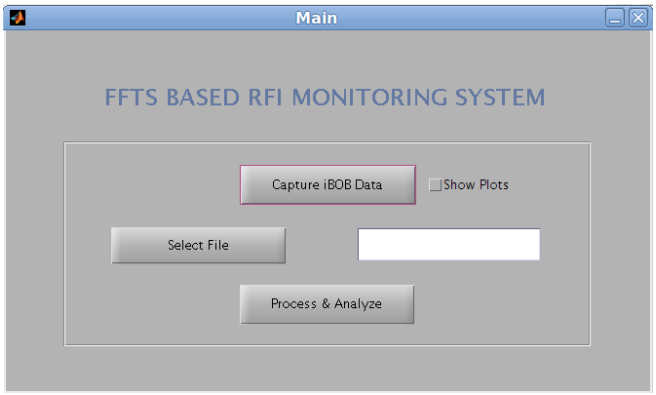



Figure 4.4.1: Main GUI

The working of the GUI is in a systematic way. First the user has to press the "Capture iBOB Data" button in order to start capturing the real time data from the iBOB. A wait bar, with a specified message, will pop-up showing the status of captured data. This can be seen in Figure 4.4.2. Moreover, "Recording, Please wait……!" message is also displayed in the MATLAB command window. The wait bar will vanish with the successful completion of captured data.

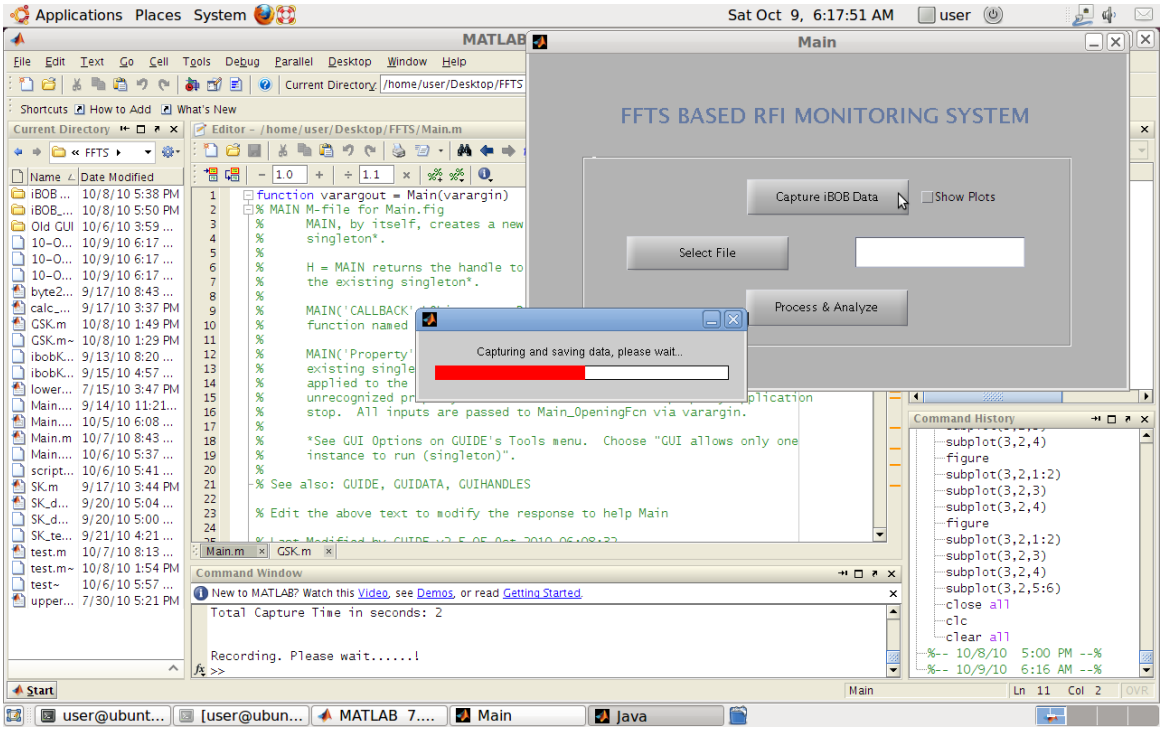



Figure 4.4.2: Wait bar in GUI

The file which is saving the data has an extension of *.out. The file name is chosen in such a way that it will show the current date and time of capture in the name. In this way a user can easily make a distinction between the different captured files.

After the completion of data capture, the user has to press the “Select File” button in order to select the file for analysis. A pop-up window will open with a message “Select File to Open”. The user can select any file for analysis whether it was just captured or saved previously. Figure 4.4.3 shows the process of selecting a file using the GUI.

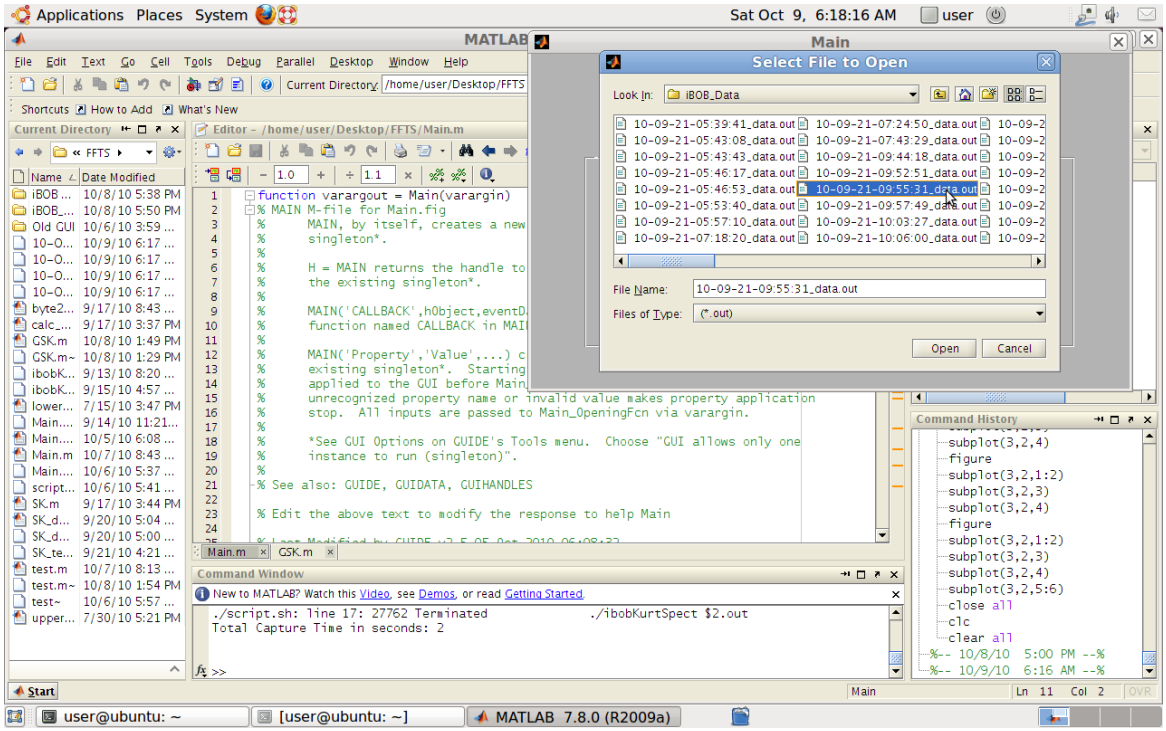



Figure 4.4.3: File Selection in GUI

After having selected the file, the user can press the "Process & Analyze" button to see the working of SK algorithm and other plots. A wait bar will pop-up for a certain time in which the analysis is being done and vanishes with the display of first plot. This can be seen from Figure 4.4.4.

Now in order to perform all the tasks i.e. capturing, saving and analysis of data, a user can "check" the checkbox 'Show Plots' available on the right side of the 'Capture iBOB Data' button. After having "checked" the checkbox, the user has to press the 'Capture iBOB Data' button. The program will automatically capture the current data from iBOB, save it, process it and will show the results after analyzing.

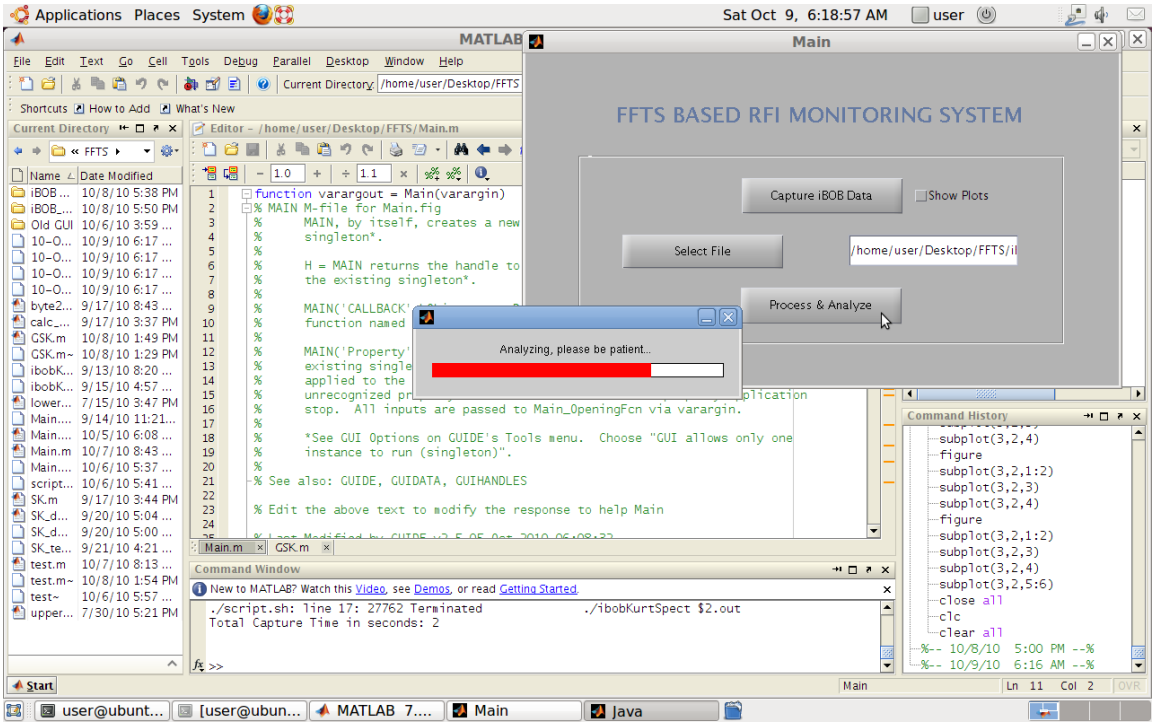



Figure 4.4.4: Analyzing the Data Using GUI

## 4.4.2. The Need for GUI

A question that could come in one's mind that, was there a need of GUI when everything could be run from the command line of MATLAB. The answer to that question is no, but it means creating a lot of difficulties for the person working with RFI detection using this software application. And these difficulties could lead toward erroneous results, errors in the execution of code and so on.

The code for capturing data from iBOB has been written in C. The reason is that C is more hardware oriented than MATLAB. Now this C code simply provides an output file with an extension of *.out which contains all the captured data. A Linux shell script has been written in order to execute that C code with respect to specified time for the data capture. Furthermore, this shell script provides the naming convention for the output file in terms of time and date.

Both the C code and Linux shell script are then interfaced with MATLAB and included in the GUI. In this way all the controlling authority goes to MATLAB. A user just has to run the GUI in MATLAB and all the tasks can then be performed as mentioned in the previous section.

# Chapter 5: Results & Discussion

## 5.1. Overview

The Spectral Kurtosis algorithm has been implemented in MATLAB. In order to check the algorithm, different simulations have been performed. These simulations have been divided into two different sets of categories. The first set involves the simulations performed using the real time data from a spectrum analyzer. The second set involves the simulations performed using real time data from the iBOB which is the actual focus of this thesis. Threshold levels and other parameters for both the set of simulations have been calculated individually within the code.

## 5.2. Simulations Utilizing Spectrum Analyzer

In order to check the SK algorithm with analog spectrometers, different simulations have been performed in MATLAB using real data from an Anritsu MS2602A Spectrum Analyzer which is one of the spectrometers available at OSO. The simulation setup includes a noise diode which is generating a wide band signal and a signal generator which is generating a sine wave at 1.2 GHz. The results presented in this section are calculated using the following parameter values;

| Parameter Name | Notation | Value |
|---|---|---|
| **Number of Samples** | M | 18 |
| **Number of Scans** | K | 1002 |
| **Duty Cycle** | D | ½ |
| **Probability of False Alarm** | PFA | 0.0013499 |

Table 9: Parameter Values

Figure 5.2.1 shows the program flow for the calculation of SK and flagging the data corrupted with RFI. The program starts with loading the file which has been generated by the spectrum analyzer into MATLAB. After loading the file, it calculates the respective frequency points based on the data available in file. The Sum of power and power squares are then calculated afterwards along with N which leads to the calculation of the SK variance. At the end, threshold levels are calculated and data are marked as corrupted or normal based on those levels.

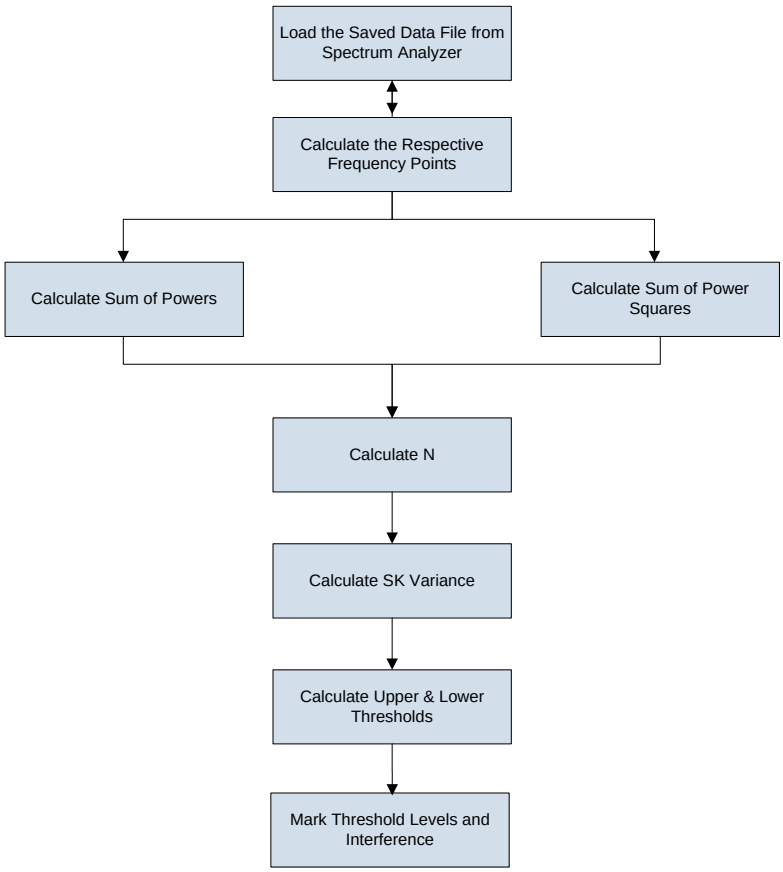



Figure 5.2.1: Spectrum Analyzer Flow Chart for SK Calculation

Figure 5.2.2 shows the spectrum power plot of the real data. The X-axis shows the number of samples "M" taken and the Y-axis shows the number of scans or the number of channels "k".

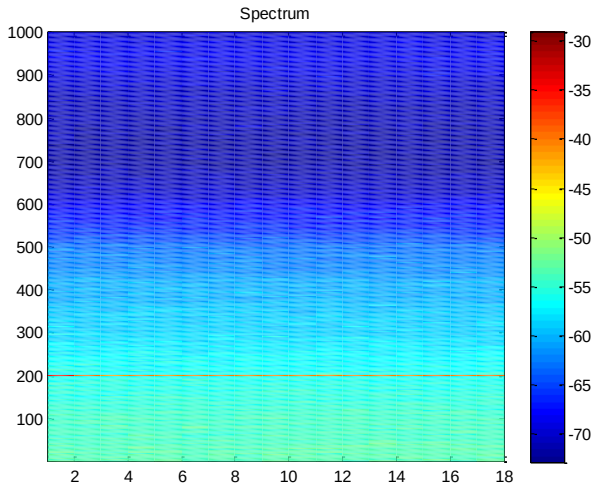



Figure 5.2.2: Power Spectrum Plot

Figure 5.2.3 shows the plot of an accumulated spectra. The signal power levels are shown in "dBm" and frequency is shown in "Hertz (Hz)". It can be clearly seen that there is interference at 1.2-GHz.

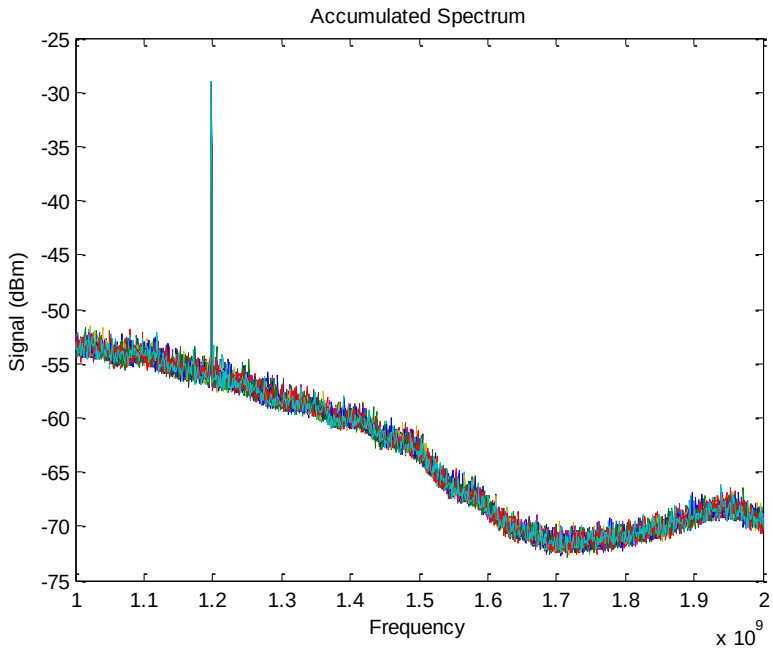



Figure 5.2.3: Accumulated Spectra

Figure 5.2.4 shows a single spectrum of the multiple spectra shown in Figure 5.2.3. Here the signal power is shown in "Milli-Watt (mW)".

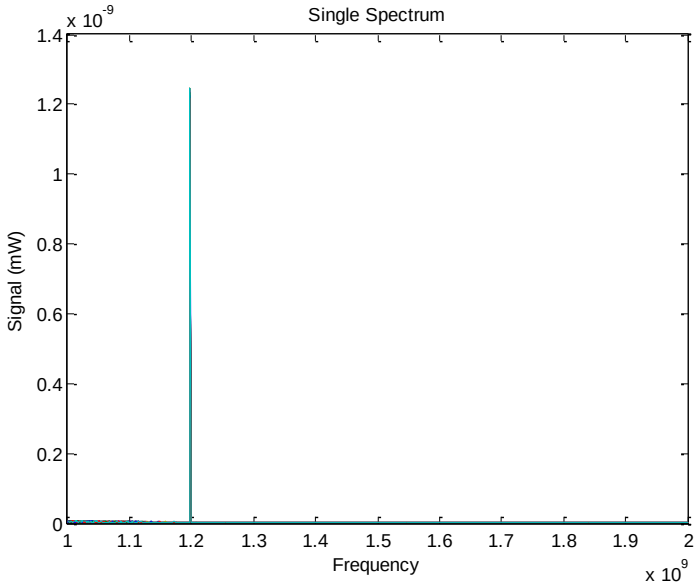



Figure 5.2.4: Single Spectrum

Figure 5.2.5 is a more clear explanation of Figure 5.2.3. In MATLAB these kinds of plots are called "Waterfall" plots. It is basically a 3-dimensional view of all the accumulated spectra. The signal power is again in "Milli-Watt (mW)".

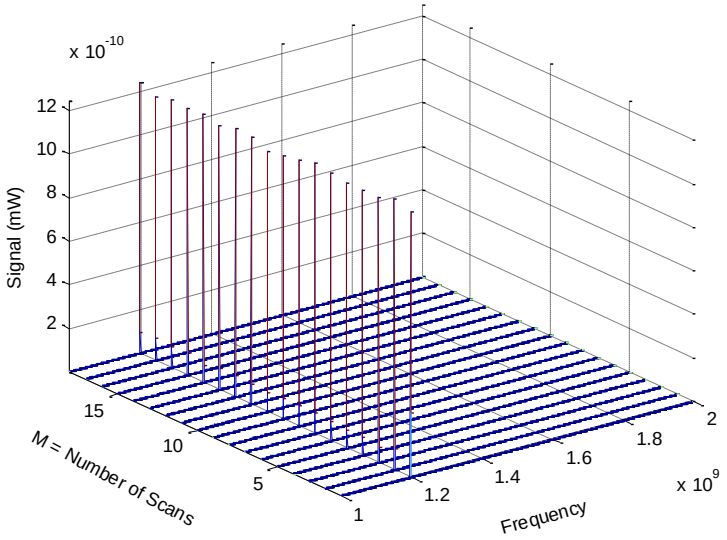



Figure 5.2.5: Waterfall Plot

Figure 5.2.3 and Figure 5.2.5 are the display of intermediate results. The final results are shown in Figure 5.2.6 and Figure 5.2.7. The first one shows the logarithmic plot of the SK estimator. It has been calculated by implying Equation (1.7). It is evident that all the frequencies are centered around "1". Moreover there is a clear interference at 1.2-GHz.

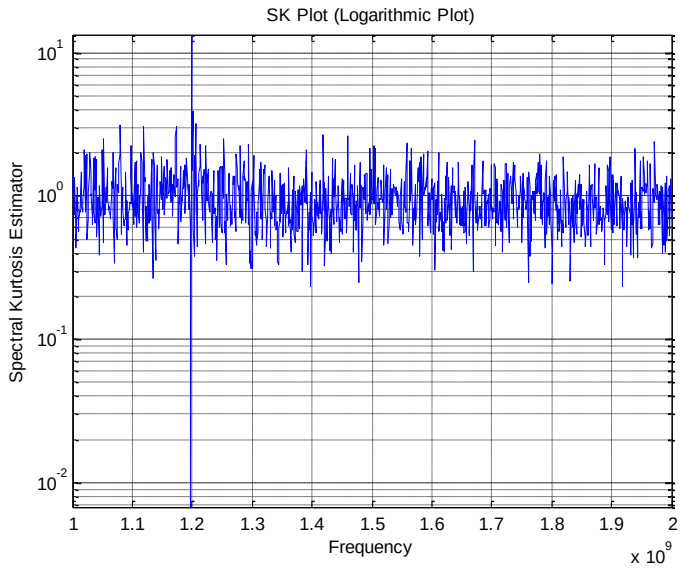



Figure 5.2.6: SK Estimator (Logarithmic Plot)

Figure 5.2.7 shows the plot of SK estimator in linear scale. Here again all the frequencies are centered around "1". Also the thresholds have been marked. The lower and upper threshold values calculated for this particular simulation are 0.2810 and 2.3644 respectively.

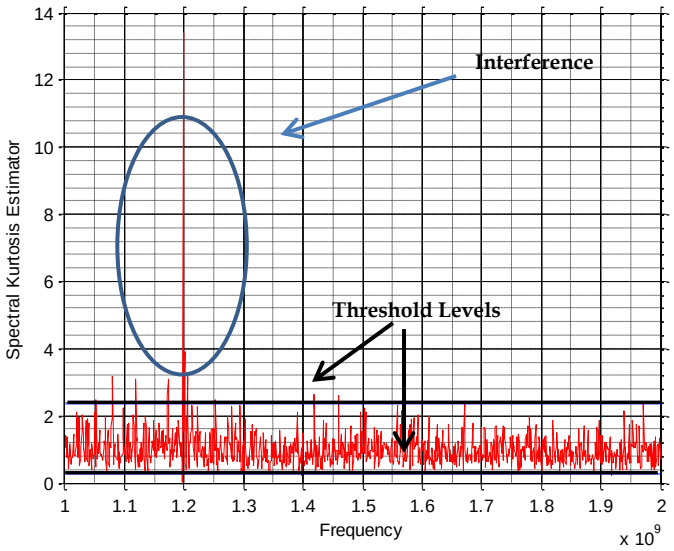



Figure 5.2.7: SK Estimator (Linear Plot)

## 5.3. Simulations utilizing iBOB

In order to check the SK algorithm with the iBOB, several simulations have been performed. The hardware arrangement includes a noise diode which is generating white noise signal over 0-26 GHz band. A signal generator has been used to generate an RF signal at 0.330 GHz with -15 dBm of power. Both the signals are fed to the iBOB input after passing through a mixer.

Figure 5.3.1 shows the program flow for the calculation of SK. After having the data converted to digital form and FFTed, averaging/accumulation of received spectra is done. Afterwards the sum of power and power squares are calculated. All this is done in the iBOB. Later on the data are saved on the disk and loaded into the MATLAB workspace where initially the SK variance is calculated. Then the upper and lower thresholds are calculated and the received data are checked for interference based on those threshold levels. Data fulfilling the threshold criteria is marked normal and vice versa. At the end, RFI free spectrum is generated.

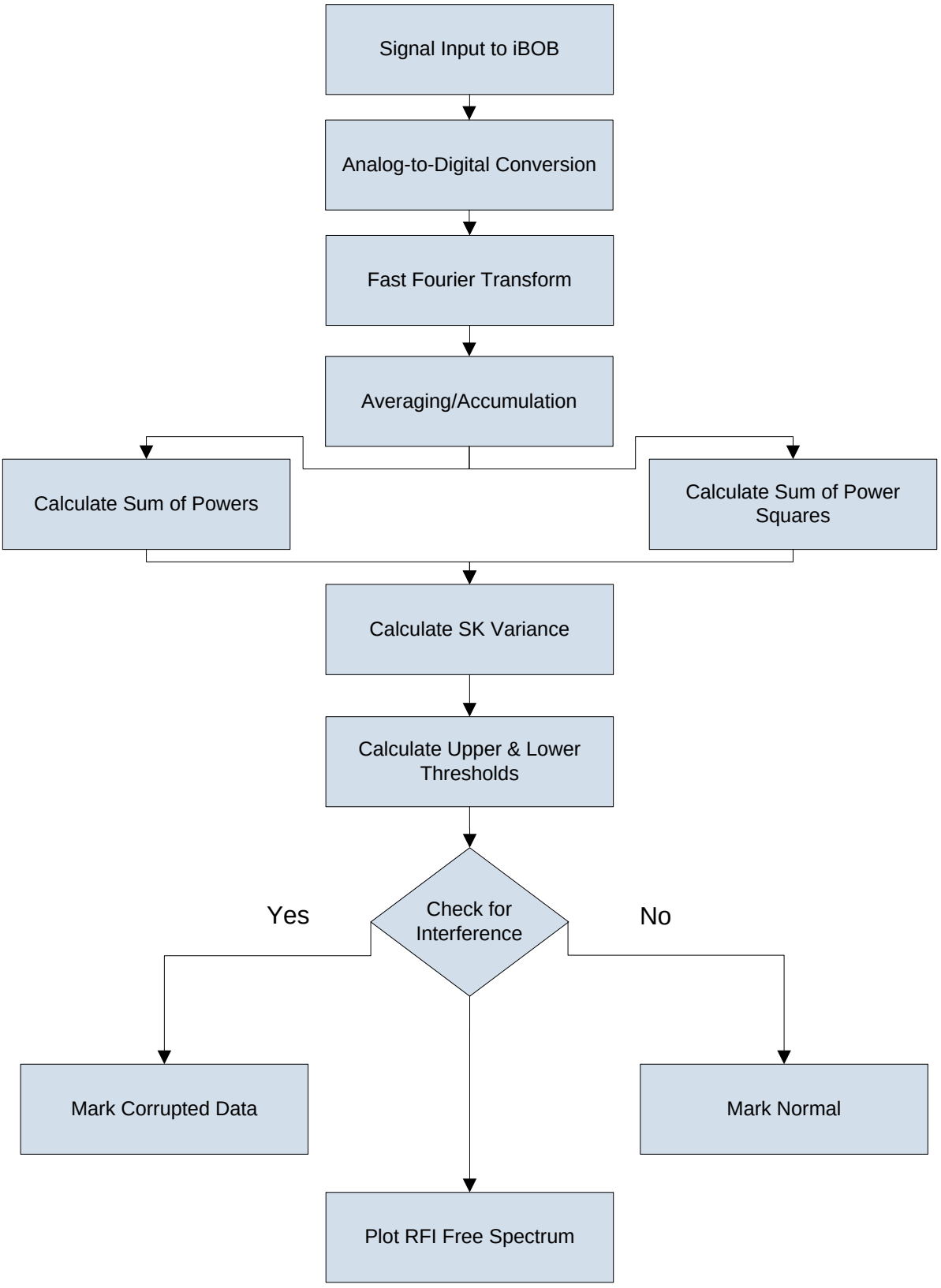



**Figure 5.3.1: iBOB Flow Chart for SK Calculation**

Figure 5.3.2 shows the complete spectrums as scanned by the iBOB in 2 seconds. The first subplot shows the spectrums in 2D. The frequency range is from 0 – 0.5 GHz.

Number of channels are 2048 and the number of scans are 73. The second subplot shows the same spectra in 3D.

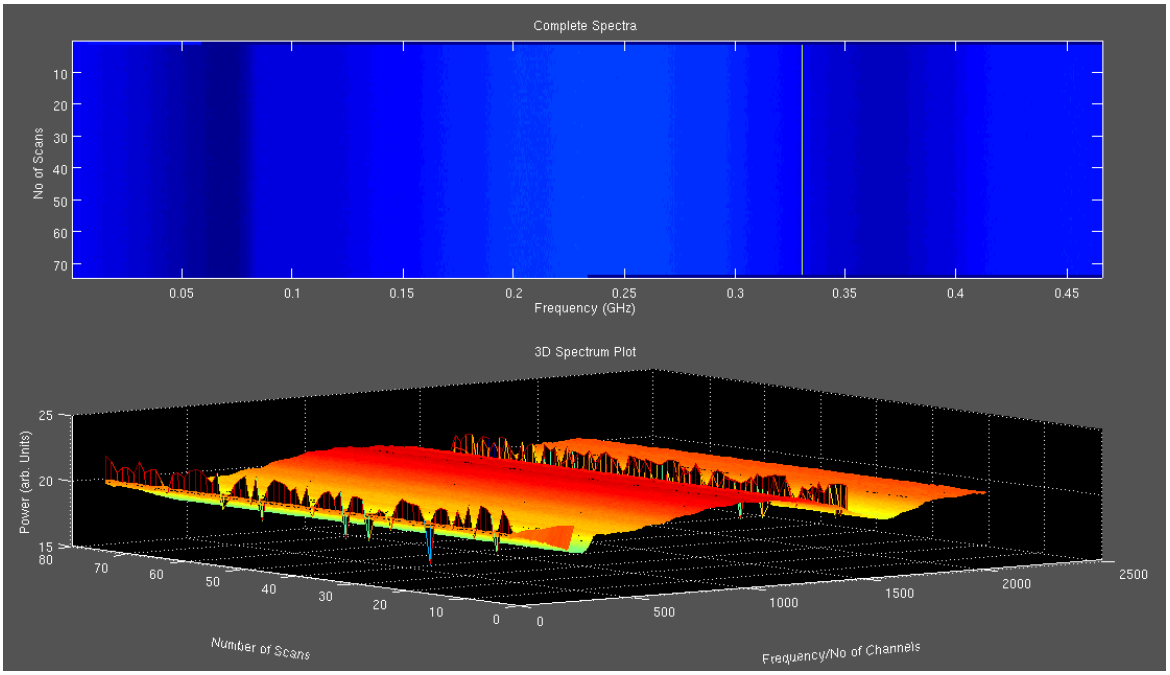



Figure 5.3.2: Power & 3D Spectrum

Figure 5.3.3 shows multiple plots which can be seen as a final result of the simulation. The first subplot shows the original spectrum. It can be seen that there is an RF signal at 0.330 GHz as generated by the function generator.

The second subplot in Figure 5.3.3 shows the result of SK algorithm after it has been applied to the original signal. Upper and lower threshold levels are marked in red lines which have been calculated using Equation (1.8) in Chapter 4. It can be seen that SK algorithm has successfully marked the RFI in the signal in accordance with the threshold levels. Moreover as mentioned in the theory of SK algorithm, all the data is centered around '1' except for the peak at 0.330 GHz.

The third subplot in Figure 5.3.3 shows all the points which have been marked as interference in the complete 500 MHz bandwidth. The interference is marked as '1' and the actual signal is marked as '0'. In this way an astronomer can easily deduce that the data marked with '1' should not be included in the signal analysis.

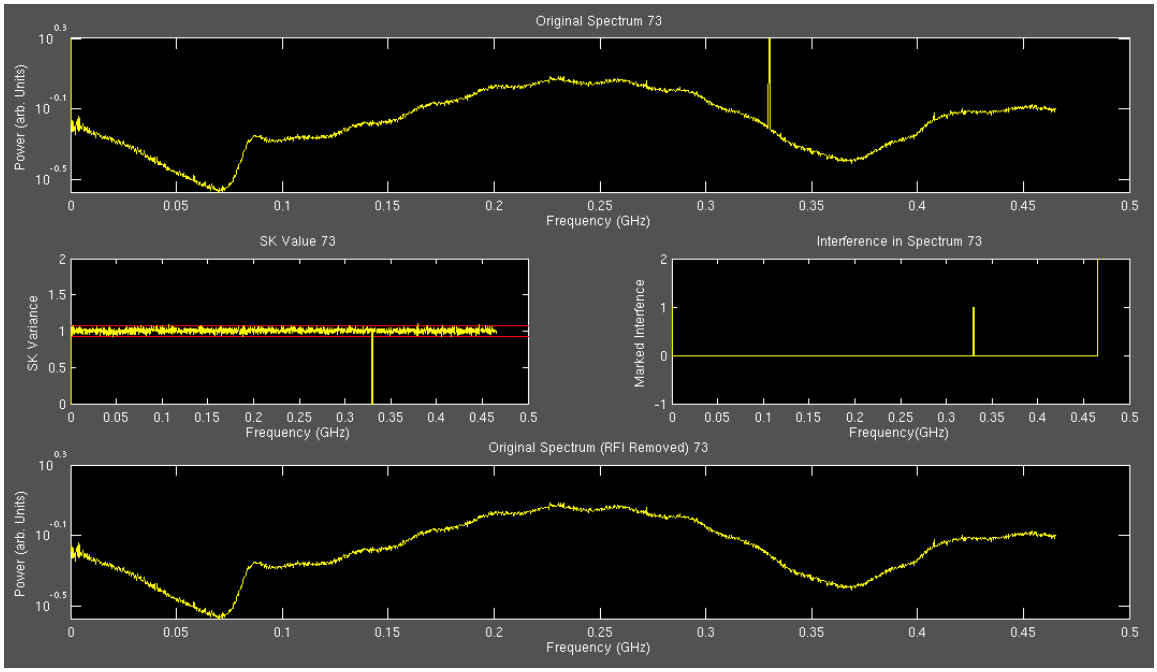



**Figure 5.3.3: RFI Monitoring Using SK**

The fourth subplot in Figure 5.3.3 shows the original spectrum after removing the RFI. It can be clearly seen that the SK algorithm has efficiently removed the RFI present at 0.330 GHz which was present in the first subplot of Figure 5.3.3.

# Chapter 6: Conclusions & Future Work

In this thesis a complete system for identification of RFI in astronomical spectra has been designed and implemented. The real time spectra is sampled and a Fast Fourier Transform is performed. The spectra estimates are passed through SK algorithm to identify RFI. The original spectra from the telescope are then flagged with respective RFI so that only correct information can be used for analysis. The platform used for this system development is Linux and the tools used include MATLAB and Simulink. The firmware for the iBOB has been developed in Simulink and is based on the design of (Kurtosis Spectrometer n.d.). The code for capturing data from the iBOB has been developed in C. Linux shell script has been written in order to execute the code properly and to save the data files from the iBOB with proper naming convention. The implementation of Spectral Kurtosis Algorithm has been done in MATLAB. A Graphical User Interface has also been created in MATLAB in order to provide an interface to all the tasks like capturing data, saving it and using it for analysis.

Since iBOB is capturing around 33 spectra per second, the RFI detection which is done in MATLAB though offline, can be viewed as a real time. It means when the iBOB data capture program is ran for 10 seconds, it will capture 330 spectra and MATLAB will analyze all these spectrums one by one. The results are shown in a form of movie where a viewer can see spectrums from 1 to 330 along with the marked RFI. The difference between the displays of two consecutive spectrums is 100ms.

Following are some recommendations for future work;

- Improving the GUI so that it can include the time asked from the user to capture iBOB data.
- Improving the C code so that it can fully execute on a Windows based operating system.
- Improving the Linux Shell script so that it can be used to do monitoring round the clock.

# Bibliography

*Spectrum Analyzer Fundamentals.* n.d. http://www.spectrum-analyzer.info/Spectrum_Spectrum_Analyzer_Fundamentals.aspx (accessed August 20, 2010).

*About LOFAR.* n.d. http://www.lofar.org/about-lofar/general-information/introduction (accessed August 15, 2010).

*ADC.* n.d. http://casper.berkeley.edu/wiki/Adc (accessed September 19, 2010).

*Block Documentation.* n.d. http://casper.berkeley.edu/wiki/Category:Block_Documentation (accessed September 19, 2010).

Camps, Jose Miguel Tarongi & Adriano. "Normality Analysis for RFI Detection in Microwave Radiometry." *Remote Sensing*, December 2009: 191-210.

CASPER. "IBOB." *CASPER.* n.d. http://casper.berkeley.edu/memos/ibob_prom_proc.pdf (accessed August 2010).

—. *MSSGE Toolflow.* n.d. http://casper.berkeley.edu/wiki/MSSGE_Toolflow (accessed May 2010).

—. *MSSGE Toolflow Setup.* n.d. http://casper.berkeley.edu/wiki/MSSGE_Toolflow_Setup (accessed May 2010).

Dou, Yujiang, et al. "The Korean Solar Radio Burst Locator (KSRBL)." (The Astronomical Society of the Pacific) 121 (May 2009): 512-526.

Dwyer, R. F. "A technique for improving detection and estimation of signals contaminated by under ice noise." *Journal of the Acoustical Society of America* 74, no. 1 (1983): 124-130.

Dwyer, R. F. "Use of the kurtosis statistic in the frequency domain as an aid in detecting random signals." *IEEE Journal of Oceanic Engineering* OE-9, no. 2 (1984): 85-92.

Dwyer, R.F. "Detection of non-Gaussian signals by frequency domain kurtosis estimation." *International Conference on Acoustic, Speech, and Signal Processing.* Boston, 1983. 607-610.

Ellingson, S.W., G.A. Hampson, and J.T. Johnson. "Design of an L-band microwave radiometer with active mitigation of interference." (Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International ) July 2003: 1751-1753.

Feng Liu, S. N. Khanna, and P. Jena. "Magnetism and local order. II. Self-consistent cluster calculations." (M. R. Press) 40, no. 1 (July 1989).

*Fft wideband real.* n.d. http://casper.berkeley.edu/wiki/Fft_wideband_real (accessed September 17, 2010).

Gary, G. M. Nita and D. E. "The Generalized Spectral Kurtosis Estimator." May 2010.

Gelu M. Nita, Dale E. Gary, and Zhiwei Liu. "Radio Frequency Interference Excision Using Spectral-Domain Statistics." 119 (July 2007): 805–827.

Guner, B., J.T. Johnson, and N. Niamsuwan. "Time and Frequency Blanking for Radio-Frequency Interference Mitigation in Microwave Radiometry." (IEEE Geoscience and Remote Sensing Society ) 45, no. 11 (November 2007): 3672 - 3679 .

*Heterodyne.* n.d. http://en.wikipedia.org/wiki/Heterodyne (accessed September 09, 2010).

*IBOB.* n.d. http://casper.berkeley.edu/wiki/IBOB (accessed August 16, 2010).

Jim Cohen, Titus Spoelstra, Roberto Ambrosini and Wim van Driel, ed. *CRAF Handbook for Radio Astronomy.* 5th. European Science Foundation, 2005.

Johnson, J.T. Gasiewski, A.J. Guner, B. Hampson, G.A. Ellingson, S.W. Krishnamachari, R. Niamsuwan, N. McIntyre, E. Klein, M. Leuski, V.Y. "Airborne radio-frequency interference studies at C-band using a digital receiver." (IEEE Transactions on Geoscience and Remote Sensing) 44, no. 7 (July 2006): 1974 - 1985.

*Karl Guthe Jansky.* n.d. http://en.wikipedia.org/wiki/Karl_Guthe_Jansky (accessed August 29, 2010).

Kendall, M. G.& Stuart, A. "The Advanced Theory of Statistics." 1 (1958): 280.

*Kurtosis Spectrometer.* n.d. http://casper.berkeley.edu/wiki/Kurtosis_Spectrometer (accessed September 14, 2010).

*Main Page.* n.d. http://casper.berkeley.edu/wiki/Main_Page (accessed August 15, 2010).

P.O Amblard, M Gaeta, and J-L. Lacoume. "Statistics for complex variables and signals - part i: Variables." *Signal Processing* 53 (1996): 1-13.

P.O Amblard, M Gaeta, and J-L. Lacoume. "Statistics for complex variables and signals - part ii: Signals." *Signal Processing* 53 (1996): 15-25.

*Pfb fir real.* n.d. http://casper.berkeley.edu/wiki/Pfb_fir_real (accessed September 15, 2010).

*Radio Astronomy.* n.d. http://en.wikipedia.org/wiki/Radio_astronomy (accessed August 26, 2010).

*Sensor Fields.* n.d. http://www.lofar.org/about-lofar/system/sensor-fields/sensor-fields (accessed September 15, 2010).

*Significant Radio Astronomy Frequencies.* n.d. http://www.setileague.org/articles/protectd.htm (accessed August 2010).

*SKA Home Page.* n.d. http://www.skatelescope.org/pages/page_genpub.htm (accessed August 15, 2010).

Xilinx. "Xilinx Support." *Xilinx.* n.d.
http://www.xilinx.com/support/documentation/user_guides/ug012.pdf (accessed September 30, 2010).

*Z-DOK Overview.* n.d.
http://www.tycoelectronics.com/catalog/bin/TE.Menu?M=MINF&MHID=173&MRID=22488&LG=1&I=13/ (accessed September 15, 2010).

# Appendix A: Code

```
% ================================================
% Code for testing SK with Analog Spectrometers
% ================================================

%% Initializing
clc, clear all, close all;
global m1 m2 m3 m4 p
M = 18;                         % No. of samples
s1 = 0;
s2 = 0;
k = 1002;                       % Channels or No of scans or Data Points
d = 0.5;                        % Duty Cycle
N = 0;                          % To be calculated later in the code
PFA = 0.0013499;                % Probability of false alarm
p = PFA;
d = 1;                          % Duty cycle
x(1) = 1;

%% Calculating Frequency Points
ini_f = 1e+9;
fin_f = 2e+9;
f = ini_f:(fin_f-ini_f)/(k-1):fin_f;
f = f.';

%% Calculating N
N = calc_N(M);                  % Calculating N from a function calc_N

%% Loading Power Values from file

% file_v = load('pow_mw.mat');
% p = file_v(2:M+1,2);            % P = power levels in dBm
% f = file_v(2:M+1,1);            % f = frequencies

load('pow_dBm.mat');
P_dBm = b1;
P_mW = (db2pow(P_dBm-30))*10^-3;    % Power in milli Watt

S_1 = sum(P_mW.');           % Calculating Sum of powers
S_2 = sum((P_mW.').^2);      % Calculating Sum of Square of powers

%% Calculating Moments for Threshold Calculation
m1 = 1*d;
m2=(2*( M^2)* (N*d) *(1 + N*d))/((-1 + M) *(6 + 5* M* (N*d) + (M^2)*( (N*d)^2)));
m3=(8*( M^3)* (N*d)* (1 + N*d)* (-2 + (N*d)* (-5 + M *(4 + N*d))))/(((-1 + M)^2)* (2 + M* N*d)
*(3 +M* N*d)* (4 + M* N*d)* (5 + M* N*d));
m4=(12*( M^4)* (N*d)* (1 + N*d)* (24 + (N*d) *(48 + 84* N*d + M *(-32 + (N*d) *(-245 - 93
*(N*d) + M* (125 + (N*d)* (68 + M + (3 + M)* (N*d)))))))/(((-1 + M)^3)* (2 + M* N*d)* (3 + M
*N*d)* (4 + M* N*d) *(5 + M *N*d)* (6 + M* N*d)* (7 + M *N*d));

%% Calculating Values of alpha, delta and duty cycle
delta = m1 - 2 * (m2^2) / m3;
beta = 4 * (m2^3) / (m3^2);
alpha = m3 / (2*m2);
disp(['alpha = ',num2str(alpha),': delta = ',num2str(delta),': d = ',num2str(d)]);

err4 = abs((3* beta *(2 + beta)* (alpha^4)/m4-1)*100);     % Compute fourth moment error

%% Calculating Thresholds
up_thres  = fsolve(@upper_root_x,x);      % Upper Threshold
%up_thres  = fzero(@upper_root_x,x);       % Upper Threshold
low_thres = fsolve(@lower_root_x,x);      % Lower Threshold
%low_thres = fzero(@lower_root_x,x);       % Lower Threshold
disp([low_thres,up_thres]);               % Displaying the Values

%% Power Spectrum Plot
figure
pcolor(b1)
shading interp
colorbar
title 'Spectrum'
axis tight
```

```
%% Original Signal Plot
figure
plot(f,b1)
xlabel 'Frequency'
ylabel 'Signal (dBm)'
title ('Accumulated Spectrum')

figure
plot(f,P_mW)
xlabel 'Frequency'
ylabel 'Signal (mW)'
title ('Single Spectrum')

%% Waterfall Plots
figure
waterfall(f,1:18,P_mW.')
xlabel 'Frequency'
set(get(gca,'xlabel'),'rotation',15)
ylabel 'M = Number of Scans'
set(get(gca,'ylabel'),'rotation',332)
zlabel 'Signal (mW)'
axis tight

%%  Calculating SK Estimator Variance
Vk = (((M*N*d)+1)/(M-1))*((M*(S_2./S_1.^2))-1);
%Vk = ((M/(M-1))*(M*(S_2./S_1.^2))-1);
Mean_SK = mean(Vk);

%% SK Plots along with Interference Detection
figure;
semilogy(f,Vk);
axis tight
grid
xlabel 'Frequency'
ylabel 'Spectral Kurtosis Estimator'
title ('SK Plot (Logarithmic Plot)')

figure;
plot(f,Vk,'r');
hold on
plot([ini_f,fin_f],low_thres*[1,1],'b')
plot([ini_f,fin_f],up_thres*[1,1],'b')
set(gca,'TickDir','out');
grid minor
xlabel 'Frequency'
ylabel 'Spectral Kurtosis Estimator'


% ============================================================
% ========= Code for Graphical User Interface (GUI) =========
% ============================================================
function varargout = Main(varargin)
% MAIN M-file for Main.fig
%      MAIN, by itself, creates a new MAIN or raises the existing
%      singleton*.
%
%      H = MAIN returns the handle to a new MAIN or the handle to
%      the existing singleton*.
%
%      MAIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAIN.M with the given input arguments.
%
%      MAIN('Property','Value',...) creates a new MAIN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Main_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Main_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Main

% Last Modified by GUIDE v2.5 05-Oct-2010 06:08:32
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Main_OpeningFcn, ...
                   'gui_OutputFcn',  @Main_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Main is made visible.
function Main_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Main (see VARARGIN)

% Choose default command line output for Main
handles.output = hObject;
clc;
global do_all
do_all = 2;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Main wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Main_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in capture_data.
function capture_data_Callback(hObject, eventdata, handles)
% hObject    handle to capture_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global do_all;
global a;
capture_time = 2;           % Setting capture time
td = datestr(now, 'yy-mmm-dd_HH:MM:SS');         % Getting time and date
if do_all==1
    %case 1          % Checkbox is checked
        sec = 0.1;
        disp('Recording. Please wait......!')
        h = waitbar(0,'Capturing and saving data, please wait...');
        steps = 1000*sec;
        for step = 1:steps
            waitbar(step / steps)
        end
        close(h)

        [status, result] = system(['./script.sh ' num2str(capture_time) ' ' td])
        a = [td '.out'];
        h = waitbar(0,'Analyzing, please be patient...');
        steps = 100;
        for step = 1:steps
            waitbar(step / steps)
```

```
        end
        close(h)
        GSK(a);
else
        %case 2          % Checkbox is not checked
        sec = 0.1;
        disp('Recording. Please wait......!')
        h = waitbar(0,'Capturing and saving data, please wait...');
        steps = 1000*sec;
        for step = 1:steps
            waitbar(step / steps)
        end
        close(h)
        [status, result] = system(['./script.sh ' num2str(capture_time) ' ' td])
end

% --- Executes on button press in results.
function results_Callback(hObject, eventdata, handles)
% hObject    handle to results (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global a
a = get(handles.fp,'String');
if isempty(a)
    errordlg('Please Enter the Correct File Name')
else
    h = waitbar(0,'Analyzing, please be patient...');
    steps = 100;
    for step = 1:steps
        waitbar(step / steps)
    end
    close(h)
    GSK(a);
end

% --- Executes on button press in sel_file.
function sel_file_Callback(hObject, eventdata, handles)
% hObject    handle to sel_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[FileName,PathName] = uigetfile('*.out');
if isequal(FileName,0)
    %errordlg('You Cancelled the File! Try Again', 'Error Dialog');
    warning off all;
else
    set(handles.fp, 'String',[PathName FileName]);
end


function fp_Callback(hObject, eventdata, handles)
% hObject    handle to fp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fp as text
%        str2double(get(hObject,'String')) returns contents of fp as a double


% --- Executes during object creation, after setting all properties.
function fp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global do_all
```

```
if (get(hObject,'Value') == get(hObject,'Max'))
 % Checkbox is checked-take approriate action
 do_all = 1;
else
 % Checkbox is not checked-take approriate action
 do_all = 2;
end
% Hint: get(hObject,'Value') returns toggle state of checkbox1


% ========================================================
% ===== Main Function for Detecting and Flagging RFI =====
% ========================================================
function G = GSK(a)

clc, close all;
%% Initializing
global m1 m2 m3 m4 p
M = 6250;                       % No. of accumulations
N = 1;                          % Number of averages before squaring power
d = 1;                          % Factor for exponential distribution
PFA = 0.0013499;                % Probability of false alarm
p = PFA;
x(1) = 1;

%% Loading Power Values from file
raw_data = dlmread(a);

nspec = uint32(size(raw_data,1));
npts = uint32(size(raw_data,2)/4);   % Number of 4-byte values
nf = npts/2;

%% Calculating Frequency Points
ini_f = 0;                          % 0.5 corresponds to 0.5MHz
fin_f = 0.512;                          % 1 corresponds to 1GHz
finc = (fin_f - ini_f)/double(nf);
f = double(1:nf)*finc / 1.1;
f = f.';
%f = f';
%% Converting to Unsigned Integer
interleaved_data = uint32(zeros(nspec,npts));
for i = 1:nspec
    interleaved_data(i,:) = byte2long(uint8(raw_data(i,:)));
end

%% Putting Data in their respective arrays
pow_data = uint32(zeros(nspec,nf));
pow_sq_data = uint32(zeros(nspec,nf));
for i = 0:7
   pow_data(:,256*i+1:256*(i+1)) = interleaved_data(:,256*i*2+1:256*(i*2+1));
   pow_sq_data(:,256*i+1:256*(i+1)) = interleaved_data(:,256*(i*2+1)+1:256*(i+1)*2);
end

%% Plot power and power squared as dynamic spectra
fullscreen = get(0,'ScreenSize');
figure('Position',[0 -50 fullscreen(3) fullscreen(4)])
subplot(2,1,1)
imagesc(f,1:nspec,log(double(pow_data)))
title('Complete Spectra')
xlabel('Frequency (GHz)')
ylabel('No of Scans')
% subplot(3,1,2)
% imagesc(f,1:nspec,log(double(pow_sq_data)))
% title('Power Squared')
subplot(2,1,2)
mesh(log(double(pow_sq_data)))
title('3D Spectrum Plot')
xlabel 'Frequency/No of Channels'
set(get(gca,'xlabel'),'rotation',5)
ylabel 'Number of Scans'
set(get(gca,'ylabel'),'rotation',352)
zlabel 'Power (arb. Units)'
pause;

%% Scaling for register settings
p_bitsel = 0;
```

```
p2_bitsel = 0;
p2_bfacc_bitsel = 0;

l = 1 - 8*(4-p_bitsel);
m = 1 - 8*(4-p2_bitsel);
n = 1 - 8*(4-p2_bfacc_bitsel);

S_1 = double(pow_data).*2^l;           % Scaling power
S_2 = double(pow_sq_data).*2^(m+n);    % Scaling square of powers



% %% Display Spectra as dynamic spectrum image
% fullscreen = get(0,'ScreenSize');
% figure('Position',[0 -50 fullscreen(3) fullscreen(4)])
% imagesc(f,1:nspec,log(S_1));
% title('Scaled power S_1')
% pause;

%% Calculating Moments for Threshold Calculation
m1 = 1*d;
m2=(2*( M^2)* (N*d) *(1 + N*d))/((-1 + M) *(6 + 5* M* (N*d) + (M^2)*( (N*d)^2)));
m3=(8*( M^3)* (N*d)* (1 + N*d)* (-2 + (N*d)* (-5 + M *(4 + N*d))))/(((-1 + M)^2)* (2 + M* N*d)
*(3 +M* N*d)* (4 + M* N*d)* (5 + M* N*d));
m4=(12*( M^4)* (N*d)* (1 + N*d)* (24 + (N*d) *(48 + 84* N*d + M *(-32 + (N*d) *(-245 - 93
*(N*d) + M* (125 + (N*d)* (68 + M + (3 + M)* (N*d)))))))/(((-1 + M)^3)* (2 + M* N*d)* (3 + M
*N*d)* (4 + M* N*d) *(5 + M *N*d)* (6 + M* N*d)* (7 + M *N*d));

%% Calculating Thresholds
up_thres  = fsolve(@upper_root_x,x) ;      % Upper Threshold
low_thres = fsolve(@lower_root_x,x) ;      % Lower Threshold
% disp([low_thres,up_thres]);               % Displaying the Values

%% Calculate SK
sk = (((M*N*d)+1)/(M-1))*((M*(S_2./S_1.^2))-1);

%% Mitigating RFI from Original Signal
rfi = pow_data;
for i = 1:nspec
    for j=1:nf-1
        if ((sk(i,j) > up_thres+0.03) | (sk(i,j) < low_thres-0.03))
            rfi(i,j) = 1;
        else
            rfi(i,j) = 0;
        end
    end
end

%% Getting Original Singal After Removing RFI
rfi_free = S_1;
for i = 1:nspec
    for j=1:nf-1
        if ((sk(i,j) > up_thres) | (sk(i,j) < low_thres))
            rfi_free(i,j) = 0;
        else
            continue;
        end
    end
end

%% Plotting the results
fullscreen = get(0,'ScreenSize');
figure('Position',[0 -50 fullscreen(3) fullscreen(4)])
whitebg('black')
for i = 1:nspec-1
    subplot(3,2,1:2)

    %semilogy(interleaved_data(i,:));
    semilogy(f,S_1(i,:)*1000.);
    AA = axis;
    axis([AA(1) AA(2) 0 2])
    title(['Original Spectrum ', num2str(i)]);
    xlabel('Frequency (GHz)');
    ylabel('Power (arb. Units)');
    pause(0.02);  % Plot spectra of Original data

    subplot(3,2,3)
```

```
    plot([ini_f,fin_f],low_thres*[1,1],'r');
    hold on
    plot([ini_f,fin_f],up_thres*[1,1],'r');
    plot(f,sk(i,:));
    A=axis;
    axis([A(1) A(2) -1 2]);
    title (['SK Value ', num2str(i)]);
    xlabel('Frequency (GHz)');
    ylabel('SK Variance');
    axis tight
    pause(0.02); % Plot the SK values
    hold  off

    subplot(3,2,4)
    plot(f,rfi(i,:));
    A=axis;
    axis([A(1) A(2) -1 2]);
    title(['Interference in Spectrum ',num2str(i)]);
    xlabel('Frequency(GHz)');
    ylabel('Marked Interfence');
    pause(0.02);

    subplot(3,2,5:6)
    semilogy(f,rfi_free(i,:)*1000.);
    AA = axis;
    axis([AA(1) AA(2) 0 2])
    title(['Original Spectrum (RFI Removed) ', num2str(i)]);
    xlabel('Frequency (GHz)');
    ylabel('Power (arb. Units)');
    pause(0.02);  % Plot spectra of Original data

end

%% Power Level Comparision of iBOB and Spectrum Analyzer
Min = min(pow_data([1:(nspec-1)],867));
Max = max(pow_data([1:nspec-1],867));
Average = mean(pow_data([1:nspec-1],867));
disp([num2str(Average),'  ',num2str(Min),'  ',num2str(Max)]);

Min = min(S_1([1:(nspec-1)],867));
Max = max(S_1([1:nspec-1],867));
Average = mean(S_1([1:nspec-1],867));
disp([num2str(Average),'  ',num2str(Min),'  ',num2str(Max)]);

disp('Successful End of Analysis')


% ================================================
% ==== Function for Lower Root Calculation =====
% ================================================
function lower_x = lower_root_x(x)
   global m1 m2 m3 m4 p
   lower_x = gammainc((-(m3-2*m2^2)/m3+x(1))/(m3/2/m2),4*(m2^3)/(m3^2))-p;


% ================================================
% ==== Function for Upper Root Calculation ======
% ================================================
function upper_x = upper_root_x(x)
   global m1 m2 m3 m4 p
   upper_x = (1-gammainc((-(m3-2*m2^2)/m3+x(1))/(m3/2/m2),4*(m2^3)/(m3^2)))-p;


% ================================================
% ========= Function for Calculating N ===========
% ================================================
function N = calc_N(M)
raw_data = dlmread('ibob_test.out');
interleaved_data = byte2long(raw_data);

c = 1, e = 2;
for i = 1:1:16
   temp_var = interleaved_data(256*(i-1)+1:256*i);         % Put 256 values turn by turn
   rem = mod(i, 2);
   if (rem == 1)
       pow_data(256*(i-c)+1:(256*c)) = temp_var;
       c = c + 1;
```

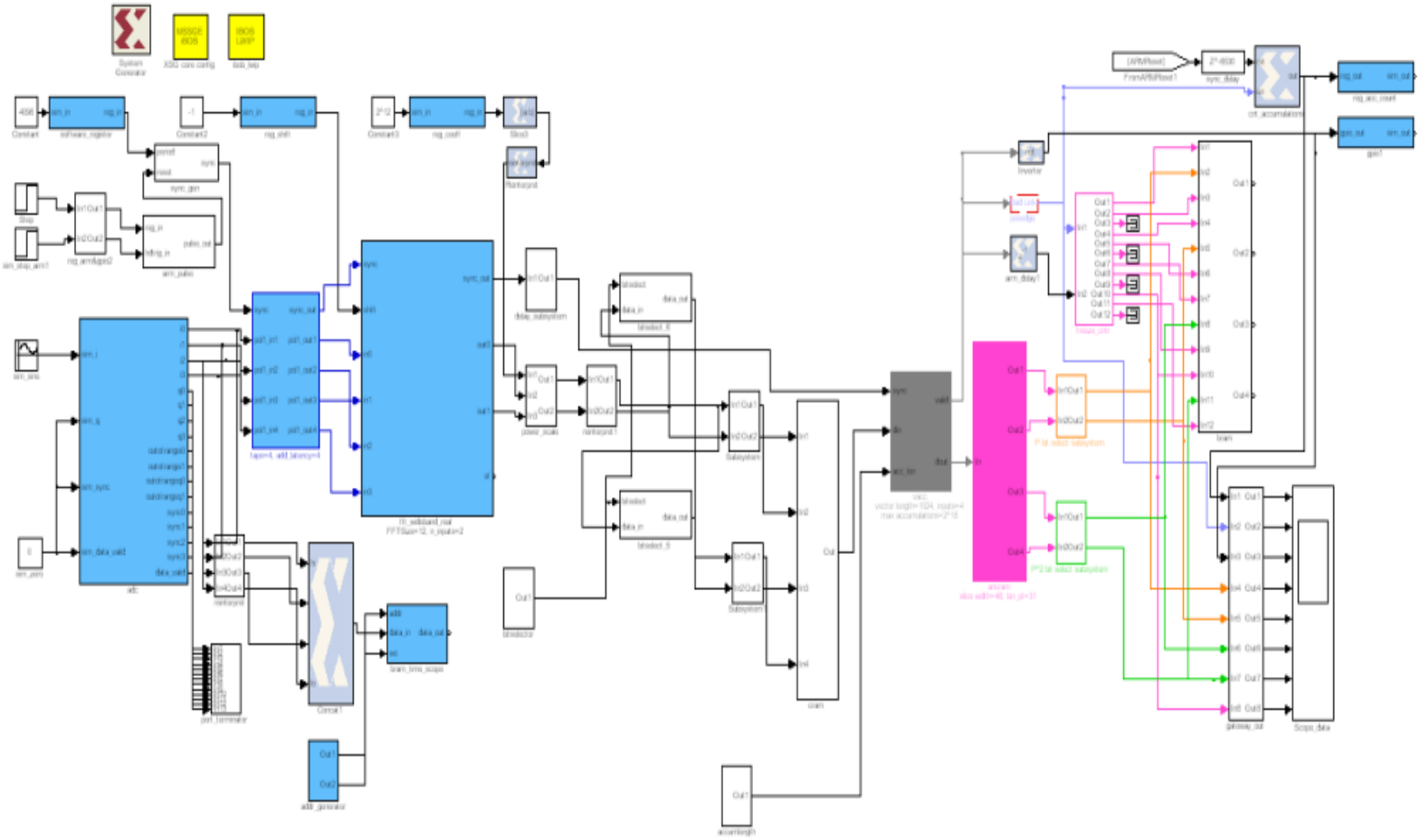
```
    else
        pow_sq_data(256*(i-e)+1:256*(e-1)) = temp_var;
        e = e + 1;
    end
end
S_1 = sum(pow_data.');           % Calculating Sum of powers
S_2 = sum(pow_sq_data.');        % Calculating Sum of Square of powers

Vk = ((M / (M-1)) * ((M*(S_2./S_1.^2)) - 1));
Vk_mean = mean(Vk);

N = 1 / Vk_mean;


% ============================================================
% ========= Function for converting bytes to long ============
% ============================================================
function out = byte2long(a)
    n = uint32(length(a))/4;
    out = uint32(zeros(1,n));
    t24 = 2^24;
    t16 = 2^16;
    t8 = 2^8;
    tmp = uint32(a);
    for i = 1:n
        out(i) = uint32(tmp(i*4-3)*t24 + tmp(i*4-2)*t16 + tmp(i*4-1)*t8 + tmp(i*4));
    end
```

# Appendix B: Simulink Diagram for iBOB



# Appendix C: Software Setup Guide

In order to run the RFI monitoring application, following software/components are needed;

- Linux OS
- MATLAB R2008 or above
- C++ Compiler

Following are the MATLAB/C++ files which should be in the same folder in order to run the RFI monitoring application. If not, MATLAB path should be manually set to all the places where these files are placed.

| | |
|---|---|
| Main.m | *Main file for running the GUI* |
| GSK.m | *Function file for calculating the SK and showing results.* |
| calc_N.m | *Function file for calculating N within the GSK.m* |
| Byte2long.m | *Function file for converting the data from iBOB* |
| upper_root_x.m | *Function file for the calculation of upper root* |
| lower_root_x.m | *Function file for the calculation of lower root* |
| ibobKurtSpect.exe | *Executable file for capturing data from iBOB.* |
| script.sh | *File that contains the Linux script for saving the iBOB capture file* |

Following are the important considerations before running the RFI monitoring application;

- The ipv4 address of the Linux system should be known as iBOB will be forwarding packets to this address.
- There should be adequate disk space in the system before capturing iBOB data files as the file save rate is around 2.1 MB/s
- The default data capture time of iBOB is 2 seconds. This can be changed by editing the file 'Main.m' at line 85 and changing '2' to any desired number.
- More captured data file size from iBOB implies more processing and consequently more RAM. MATLAB might crash if RAM is less than 2 GB and file sizes generated with more than 30 seconds capture time are used.
- The path to save data can be specified in the MATLAB code, if required, otherwise it will save the data in working directory.
- The procedure for uploading the software to iBOB can be found at (CASPER, IBOB u.d.).
- The MSSGE Toolflow information can be found at (CASPER, MSSGE Toolflow u.d.) and (CASPER, MSSGE Toolflow Setup u.d.).

# Index

## S



## T



## V



## X