Low Complexity Energy Detection for Spectrum Sensing with Random Arrivals of Primary Users

**ABSTRACT**

Due to random arrivals of primary user signals, the timing misalignment issue should be considered for spectrum sensing in cognitive radio (CR) systems, such as CR based femtocell networks. To deal with this issue, two approaches were recommended in the literature, including Bayesian and generalized likelihood ratio test (GLRT) detectors. However, Bayesian test requires perfect knowledge of the distribution of unknown parameters. Therefore, it is impractical due to its implementation complexity. To design a low complexity energy detector (ED), this work proposes an ED scheme based on GLRT algorithm. As a result, maximum-likelihood (ML) estimation for the timing misalignment is devised, and the performance of the proposed scheme is analyzed. The results show that the proposed GLRT detector features a low complexity and satisfactory performance.

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **LIST OF TABLES** |  |
| **1.** | **INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM   4. DISADVANTAGE OF EXISTING SYSTEM   5. LITERATURE OF SURVEY   6. PROPOSED SYSTEM   7. ADVANTAGE OF PROPOSED SYSTEM |  |
| **2.** | **PROJECT DESCRIPTION**  **2.1** GENERAL  **2.2** TECHNIQUES  **2.3** MODULES  **2.4** MODULES EXPLANATION  **2.4.1** SYSTEM MODEL  **2.4.2** ENERGY DETECTOR STRUCTURE THROUGH THE APPROXIMATIONS IN DIFFERENT SNR REGIMES   * FALSE ALARM PROBABILITY * DETECTION PROBAILITY   **2.4.3** LOCALLY OPTIMUM DETECTOR STRUCTURE THROUGH THE APPROXIMATIONS IN DIFFERENT SNR REGIMES   * FALSE ALARM PROBABILITY * DETECTION PROBAILITY |  |
| **3.** | **SOFTWARE SPECIFICATION**  **3.1** GENERAL INTRODUCTION TO MATLAB  **3.2** FEATURES OF MATLAB  **3.3** THE MATLAB SYSTEM  **3.4** DESKTOP TOOLS  **3.5** DATA ANALYSIS  **3.2.1** VARIABLES  **3.2.2** NUMBERS  **3.2.3** OPERATORS  **3.2.4** FUNCTIONS  **3.6** VISUALIZING DATA  **3.7** HANDLING MATRICES  **3.7.1** ENTERING MATRICES AND ADDRESSING  THE ELEMENTS  **3.7.2** GENERATING MATRICES  **3.7.3** CONCATENATION  **3.7.4** DELETING ROWS AND COLUMNS  **3.7.5** ARRAY ORIENTATION  **3.7.6** SCALAR-ARRAY MATHEMATICS  **3.3.7** ARRAY-ARRAY MATHEMATICS  **3.8** CURVE FITTING  **3.8.1** CREATING A PLOT  **3.8.2** CONTROLLING AXES  **3.9** FLOW CONTROL  **3.10** VECTOR MANIPULATIONS  **3.10.1** BASIC OPERATIONS  **3.10.2** MATHEMATICAL OPERATIONS ON  VECTORS  **3.10.3** NATURAL FUNCTIONS OF VECTORS  **3.11** BASIC PLOTTING  **3.11.1** PLOTS WITH BASIC FUNCTIONALITY  **3.11.2** PLOTS USING ‘SUBPLOT’  **3.11.3** PROBLEMS  **3.11.3.1** SAMPLING OF ’CONTINUOUS’ WAVEFORM  **3.11.3.2** SIMPLE 3D-PLOTS  **3.12** RANDOM FUNCTIONS  **3.12.1** CONTINUOUS UNIFORM DISTRIBUTION  **3.12.2** DISCRETE UNIFORM DISTRIBUTION  **3.12.3** GAUSSIAN DISTRIBUTION |  |
| **4.** | **IMPLEMENTATION**  **4.1.** CODING IMPLEMENTATION  **4.2**. SCREEN SHOTS |  |
| **5.** | **CONCLUSION** |  |
|  | **REFERENCES** |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG NO** | **FIGURE NAME** | **PAGE NO** |
| **1** | BEAM FORMING |  |
| **2** | BLOCK DIAGRAM |  |
| **3** | COMMAND WINDOW |  |
| **4** | SINE PLOT |  |
| **5** | MULTIPLE GRAPHS WITH A SINGLE CALL TO PLOT |  |
| **6** | EXAMPLE FOR CONTROLLING A AXIS |  |
| **7** | AVERAGE FALSE ALARM PROBABILITIES USING ANALYTICAL RESULTS AS WELL  AS SIMULATION RESULTS AT DIFFERENT SNRS FOR DETECTION PROBABILITY OF 0.95 AND *Ρ* = 0*.*5. |  |
| **8** | AVERAGE DETECTION PROBABILITIES USING ANALYTICAL RESULTS AS WELL AS  SIMULATION RESULTS AT DIFFERENT SNRS FOR FALSE ALARM PROBABILITY OF 0.05 AND *Ρ* = 0*.*5. |  |
| **9** | AVERAGE FALSE ALARM PROBABILITIES VERSUS THE NUMBER OF SAMPLES FOR  DETECTION PROBABILITY OF 0.95 |  |
| **10** | AVERAGE DETECTION PROBABILITIES VERSUS THE NUMBER OF SAMPLES FOR  FALSE ALARM PROBABILITY OF 0.05 |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TABLE NAME** | **PAGE NO** |
| **1** | BASIC PLOTTING FUNCTION |  |
| **2** | SUBPLOT |  |
| **3** | PROPERTIES |  |
| **4** | UNIFORM DISTRIBUTION |  |
| **5** | GAUSSIAN DISTRIBUTION |  |

**LIST OF ABBRIVATIONS**

|  |  |  |
| --- | --- | --- |
| **SNO** | **ABBREVIATIONS** | **EXPANSION** |
| **1** | ML | maximum-likelihood |
| **2** | GLRT | generalized likelihood ratio test |
| **3** | ED | energy detector |
| **4** | SNR | Signal-to-Noise Ratio |

**CHAPTER 1**

**INTRODUCTION**

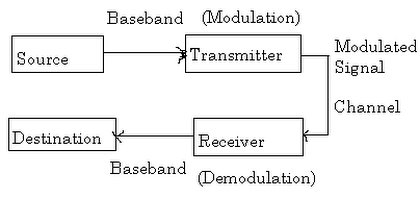
* 1. **GENERAL**

Communication is the activity of conveying meaningful information. Communication requires a sender, a message, and an intended recipient, although the receiver need not be present or aware of the sender's intent to communicate at the time of communication; thus communication can occur across vast distances in time and space. The communication process is complete once the receiver has understood the message of the sender.

**TWO TYPES OF COMMUNICATIONS:**

Wired communication: Wired technology has been around for ages. It first became popular in the early 1900's with the introduction of the telephone network. The use of wired connections spawned the creation of other technologies like multiplexing and SONET. Using physical wires means that electronic signals are being transmitted over a metal conductor. Currently, this is the most reliable way to transmitting/receiving data or voice on the planet. The Internet itself transmits a large amount of data through fiber optic cabling but also employs a large amount of T1/T3 lines that run over standard copper wiring.

Wireless communication: Wireless communication is the transfer of information over a distance without the use of electrical conductors or "wires". The distances involved may be short (a few meters as in television remote control) or long (thousands or millions of kilometers for radio communications). Wireless communication is generally considered to be a branch of telecommunications.



**Source:** A source is from which the information is taken.

**Destination:** A destination is at which the information is received.

**Transmitter:** Transmitter sends the information that is received from the source into the channel by modulating it.

**Channel:** A channel is the medium in which the modulated signal is sent from the transmitter is propagated to the receiver.

**Receiver:** Receiver gets the modulated signal then it demodulates it then sends the information to the destination.

Thus in wireless communication the information is conveyed.

Wireless communication is yet divided into:

**RF communication:** Short for radio frequency, any frequency within the electromagnetic spectrum associated with radio wave propagation. When an RF current is supplied to an antenna, an electromagnetic field is created that then is able to propagate through space. Many wireless technologies are based on RF field propagation.

**FSO communication:** Free-space optics (FSO), also called free-space photonics (FSP), refers to the transmission of modulated visible or infrared (IR) beams through the atmosphere to obtain broadband communications. Most frequently, laser beams are used, although non-lasing sources such as light-emitting diodes (LED s) or IR-emitting diodes (IREDs) will serve the purpose.

The theory of FSO is essentially the same as that for fiber optic transmission. The difference is that the energy beam is collimated and sent through clear air or space from the source to the destination, rather than guided through an optical fiber. FSO systems can function over distances of several kilometers. As long as there is a clear line of sight between the source and the destination, communication is theoretically possible. Even if there is no direct line of sight, strategically positioned mirrors can be used to reflect the energy.

Applications: We use wireless communication in variety of fields like Bluetooth, GSM, Zigbee, Wi-Max, Tetra etc.

**COGNITIVE NETWORK**

In recent years, the words cognitive and smart have become buzzwords that are applied to many different networking and communications systems. At a minimum, in the current literature we find mention of cognitive networks. The opportunistic use of the wireless spectrum has been a hot research topic in the wireless communications community in recent years due to the intense competition for the use of spectrum at frequencies below 3 GHz. Cognitive network has a cognitive process that can perceive current network conditions, and then plan, decide and act on those conditions. The network can learn from these adaptations and use them to make future decisions; all while taking into account end to end goals.

A cognitive network consists of a number of traditional wireless service subscribers and they are called as cognitive users. The traditional wireless service subscribers have the legacy priority access to the spectrum and are usually called primary users in this network. Cognitive users presented in this system are also known as the secondary users, are allowed to access the spectrum only if communication does not create significant interference to the licensed primary users.

A cognitive radio is a kind of two-way radio that automatically changes its transmission or reception parameters, in such a way that the entire wireless communication network -- of which it is a node -- communicates efficiently, while avoiding interference with licensed or unlicensed users. This alteration of parameters is based on the active monitoring of several factors in the external and internal radio environment, such as radio frequency spectrum, user behavior and network state.

The Cognitive Radio (CR) concept is a new wireless communication paradigm that improves the spectrum usage efficiency by exploiting the existence of spectrum holes. CRNs are networks that have cognitive and reconfigurable properties and the capability to detect unoccupied spectrum holes and change frequency for end-to-end communication. In most of the existing proposals, CRNs employ three steps of basic functionality. Observing and sensing is the first step of the cognitive process. The next step is to identify and analyze the spectrum. The last step is sharing the spectrum information and executing spectrum assignment.

Soft decision based CS is designed analytically and to analyze the detector in several situations, i.e., signal model where single-carrier case and multi-carrier case are assumed and two scenarios; in the first scenario, SNR values of secondary users are totally equal and in the second scenario, a certain SNR difference between secondary users is assumed. We present numerical results as follows. The first scenario shows that there is little difference between the signal models in terms of detection performance. The second scenario shows that CS is superior to non-cooperative sensing. In addition, we presents that detection performance of soft decision based CS outperform detection performance of hard decision based CS.

With the growing number of wireless devices and increased spectrum occupancy, the unlicensed spectrum is getting scarce. Additionally, large portions of the licensed spectrum, even in urban areas, are underutilized. To address the potential spectrum exhaustion problem, new wireless communication paradigms have been proposed for future wireless communication devices. The Cognitive Radio (CR) concept is a new wireless communication paradigm that improves the spectrum usage efficiency by exploiting the existence of spectrum holes. Devices using CRs referred to as Secondary Users (SUs), are aware of their spectrum environments and change their transmission and reception parameters to avoid interference with licensed spectrum users referred to as Primary Users (PUs). Networks consisting of nodes equipped with CRs are referred to as Cognitive Radio Networks (CRNs).

CRNs are networks that have cognitive and reconfigurable properties and the capability to detect unoccupied spectrum holes and change frequency for end-to-end communication. In most of the existing proposals, CRNs employ three steps of basic functionality. Observing and sensing is the first step of the cognitive process. The next step is to identify and analyze the spectrum. The last step is sharing the spectrum information and executing spectrum assignment. In addition to these awareness functionalities, to maintain seamless communication, several proposals envision spectrum mobility which is caused by three reasons such as PU detection, channel degradation, and SU mobility.

**SPECTRUM HANDOFF IN CRNS:**

In CRNs, spectrum mobility causes a new type of handoff referred to as spectrum handoff, which is different from traditional cellular handoff and mainly caused by the presence of PUs. In cellular networks, mobile devices transfer an ongoing connection from one channel to another channel between base stations due to user mobility or channel degradation. However, the concept of user movement has also new meanings in CRNs because the number and characteristic of available spectrum at a new location may vary with PU spectrum usage. Moreover, the spectrum handoffs in CRNs are likely to incur longer delays or temporary communication disruptions because SUs must search for spectrum holes and choose a proper channel at every spectrum handoff. We are developing a new type of spectrum handoff to reduce temporary communication disruption time which is caused by spectrum handoffs.

* 1. **OBJECTIVE**

The optimal estimation of the arrival time for a CR energy detection is devised. The estimation performance of the arrival time is analyzed. The GLRT detector is formulated and its performance is analyzed. The performance of the proposed GLRT detector offers a performance close to the benchmark, and it features a low computational complexity.

* 1. **EXISTING SYSTEM**

Using cognitive radios (CRs), the secondary users (SUs) are allowed to use the spectrum originally allocated to primary users (PUs) as long as the primary users are not using it temporarily. To avoid interference to the primary users, the SUs have to perform spectrum sensing before their attempts to transmit over the spectrum. Upon detecting that the PU is idle, the SUs can make use of the spectrum for transmission, and the overall utilization efficiency of the spectrum is enhanced

* 1. **EXISTING SYSTEM DISADVANTAGES**
* Spectrum utilization is not efficient.
* Noise is high.
  1. **LITERATURE OF SURVEY**

**TITLE**  : Enabling Mobile Traffic Offloading via Energy Spectrum Trading

**AUTHOR** : Tao Han, *Student Member, IEEE*, and Nirwan Ansari, *Fellow, IEEE*

**YEAR**  : JUNE 2014

Green communications has received much attention recently. For mobile networks, the base stations (BSs) account for more than 50% of the energy consumption of the networks. Therefore, reducing the power consumption of BSs is crucial to greening mobile networks. In this paper, we propose a novel energy spectrum trading (EST) scheme which enables the macro BSs to offload their mobile traffic to Internet service providers’ (ISPs’) wireless access points by leveraging cognitive radio techniques. Since the ISP’s wireless access points are usually closer to the mobile users, the energy and spectral efficiency of mobile networks are enhanced. However, in the EST scheme, achieving optimal mobile traffic offloading in terms of minimizing the energy consumption of the macro BSs is NP-hard. We thus propose a heuristic algorithm to approximate the optimal solution with low computation complexity. We have proved that the energy savings achieved by the proposed heuristic algorithm is at least 50% of that achieved by the brute-force search. Simulation results demonstrate the performance and viability of the proposed EST scheme and the heuristic algorithm.

**TITLE**  : Improved energy detection spectrum sensing for cognitive radio

**AUTHOR**  : M. Lo´ pez-Benı´tez F. Casadevall Department of Signal Theory and Communications, Universitat Polite` cnica de Catalunya, Barcelona, Spain

**YEAR** : July 2010

Energy detection constitutes a preferred approach for spectrum sensing in cognitive radio owing to its simplicity and applicability (it works irrespective of the signal format to be detected) as well as its low computational and implementation costs. The main drawback, however, is its well-known detection performance limitations. Various alternative detection methods have been shown to outperform energy detection, but at the expense of increased complexity and confined field of applicability. In this context, this work proposes and evaluates an improved version of the energy detection algorithm that is able to outperform the classical energy detection scheme while preserving a similar level of algorithm complexity as well as its general applicability regardless of the particular signal format or structure to be detected. The performance improvement is evaluated analytically and corroborated with the experimental results.

**TITLE** : Iterative Synchronization-Assisted Detection of OFDM Signals in Cognitive Radio Systems

**AUTHOR** : Wen-Long Chin, Chun-Wei Kao, Hsiao-Hwa Chen, *Fellow, IEEE*, and Teh-Lu Liao, *Member, IEEE*

**YEAR**  : May 2014

Despite many attractive features of an orthogonal frequency-division multiplexing (OFDM) system, the signal detection in an OFDM system over multipath fading channels remains a challenging issue, particularly in a relatively low signal to- noise ratio (SNR) scenario. This paper presents an iterative synchronization- assisted OFDM signal detection scheme for cognitive radio (CR) applications over multipath channels in low-SNR regions. To detect an OFDM signal, a log-likelihood ratio (LLR) test is employed without additional pilot symbols using a cyclic prefix (CP). Analytical results indicate that the LLR of received samples at a low SNR can be approximated by their log-likelihood (LL) functions, thus allowing us to estimate synchronization parameters for signal detection. The LL function is complex and depends on various parameters, including correlation coefficient, carrier frequency offset (CFO), symbol timing offset, and channel length. Decomposing a synchronization problem into several relatively simple parameter estimation subproblems eliminates a multidimensional grid search. An iterative scheme is also devised to implement a synchronization process. Simulation results confirm the effectiveness of the proposed detector.

**TITLE :** On Green‐Energy‐Powered Cognitive Radio Networks

**AUTHOR :** Xueqing Huang, Tao Han, and Nirwan Ansari,” IEEE Communications Surveys and Tutorials, vol. 17, no. 2, pp. 827‐842, 2nd Quarter, 2015

**YEAR**  April 2015

Green energy powered cognitive radio (CR) network is capable of liberating the wireless access networks from spectral and energy constraints. The limitation of the spectrum is alleviated by exploiting cognitive networking in which wireless nodes sense and utilize the spare spectrum for data communications, while dependence on the traditional unsustainable energy is assuaged by adopting energy harvesting (EH) through which green energy can be harnessed to power wireless networks. Green energy powered CR increases the network availability and thus extends emerging network applications. Designing green CR networks is challenging. It requires not only the optimization of dynamic spectrum access but also the optimal utilization of green energy. This paper surveys the energy efficient cognitive radio techniques and the optimization of green energy powered wireless networks. Existing works on energy aware spectrum sensing, management, and sharing are investigated in detail. The state of the art of the energy efficient CR based wireless access network is discussed in various aspects such as relay and cooperative radio and small cells. Envisioning green energy as an important energy resource in the future, network performance highly depends on the dynamics of the available spectrum and green energy. As compared with the traditional energy source, the arrival rate of green energy, which highly depends on the environment of the energy harvesters, is rather random and intermittent. To optimize and adapt the usage of green energy according to the opportunistic spectrum availability, we discuss research challenges in designing cognitive radio networks which are powered by energy harvesters..

* 1. **PROPOSED SYSTEM**

Generalized likelihood ratio test (GLRT) detector to tackle the issue on random arrivals of primary users that follow a Poisson process, while that in proposed a Bayesian detector for uniform arrival times. However, the distribution of timing misalignments is essentially unknown in a real system. The detection of idle spectra is typically considered as a binary hypothesis test in a low signal to noise ratio (SNR) region.

* 1. **ADVANTAGES OF PROPOSED SYSTEM**
* Spectrum utilization is high.
* Noise is low compare to the existing system.

**CHAPTER 2**

**PROJECT DESCRIPTION**

* 1. **GENERAL**

RADIO Frequency (RF) spectrum is an expensive and limited resource for wireless communications. The increasing demands for additional bandwidth have led to studies that indicate the spectrum assigned to primary license holders is under-utilized. Cognitive radio technology helps to use the RF spectrum more efficiently, by introducing secondary usage of the spectrum licensed to primary users (PU) but with a lower priority. A cognitive radio is able to change its transmitter parameters based on interaction with the environment. Secondary users (SU’s) equipped with cognitive radios can sense the spectrum and dynamically use spectrum holes in PU frequency bands for data transmission. Secondary users are not allowed to introduce any interference to the primary license holders. Therefore, before starting their transmission, they need to be aware of the presence of the PUs. Spectrum sensing is one method for detecting the presence or absence of a primary license holder. This is a challenging task because the PU signal is usually very weak due to fading, shadowing, etc. There are a few main categories of spectrum sensing including matched filtering, energy detection cyclostationarity-based detection and eigenvalue-based detection. Energy detection is the simplest method but it is optimized for impairment with additive white Gaussian noise (AWGN). It is often assumed that the additive noise samples are statistically independent. Although the AWGN assumption is valid in several situation, the work in this paper is mainly motivated by situations that we have encountered where the noise exhibits significant correlation. As a particular example, the authors have recently explored some of the characteristics of noise within power substation for smart grid wireless monitoring applications .In such applications, the noise as experimentally measured presents several characteristics, one of them being correlation in the time domain. The noise models needed in these cases quickly become complicated and involve most often Markov transition models. Any realistic cognitive radio environment would require to at least take into account at some level the noise correlation. This fact shows the importance of investigations on signal detection in the presence of correlated noise. General optimum detection in the presence of correlated noise models have been considered in for the −mixing noise model, for m−independent noise model and for moving average non-Gaussian noise models. In case the dependence is weak, the first order moving average (MA) of an i.i.d. random process is considered as a weakly correlated noise model In all these studies, detection schemes are designed for known signals.

In this paper, we investigate a Locally Optimum (LO) detection of random signals under a weakly correlated noise model over fading channels. In practice, simple energy detection [6] is often preferred over more complex detection techniques. Therefore, we try to come up with an LO detection technique with comparable complexity to energy detection for correlated noise environments. The performance of LO detection is measured using false alarm and detection probabilities. In fading channel, these two probabilities depend on the channel gain h and we need to perform averaging over h in order to find final average false alarm and detection probabilities. We have derived theoretical averages for these probabilities under fading condition. In order to validate these theoretical results, we perform simulations over a large number of channel gains and obtain averages. We will show that the simulation results are in good match with theoretical results. The performance of the proposed locally optimum detection is shown to be better compared to the simple energy detection. In case the estimated correlation between noise samples is different from the real correlation, we have also derived the detection and false alarm probabilities and their theoretical averages in terms of both estimated and actual correlations and investigated the effect of correlation mismatch on the performance of the proposed detection method.

**PROPOSED SYSTEM TECHNIQUE**

**2.2.1 BEAM FORMING**

Beamforming can be used for radio or sound waves. It has found numerous applications in radar, sonar, seismology, wireless communications, radio astronomy, acoustics, and biomedicine. Adaptive beamforming is used to detect and estimate the signal-of-interest at the output of a sensor array by means of optimal spatial filtering and interference rejection. Beam forming is a signal processing technique used in sensor arrays for directional signal transmission or reception. This is achieved by combining elements in the array in such a way that signals at particular angles experience constructive interference while others experience destructive interference. Beam forming can be used at both the transmitting and receiving ends in order to achieve spatial selectivity.

Beam forming techniques are mainly used to change the directionality of the array. When transmitting, a beamformer controls the phase and relative amplitude of the signal at each transmitter, in order to create a pattern of constructive and destructive interference in the wave front.

Beamforming techniques can be broadly divided into two categories

* conventional (fixed or switched beam) beamformers
* Adaptive beamformers or phased array

Conventional beamformers use a fixed set of weightings and time-delays (or phasing’s) to combine the signals from the sensors in the array, primarily using only information about the location of the sensors in space and the wave directions of interest. In contrast, adaptive beamforming techniques generally combine this information with properties of the signals actually received by the array, typically to improve rejection of unwanted signals from other directions. This process may be carried out in either the time or the frequency domain.

All the weights of the antenna elements can have equal magnitudes. The beamformer is steered to a specified direction only by selecting appropriate phases for each antenna. If the noise is uncorrelated and there are no directional interferences, the signal-to-noise ratio of a beamformer is given by

 (1)

Where P = Transmitting power, = Noise Power

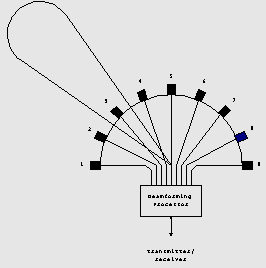


Fig.1 BEAM FORMING

**2.2.2. PRECODING**

Precoding is a generalization of beamforming to support multi-layer transmission in multi-antenna wireless communications. In conventional single-layer beamforming, the same signal is emitted from each of the transmit antennas with appropriate weighting such that the signal power is maximized at the receiver output. When the receiver has multiple antennas, single-layer beamforming cannot simultaneously maximize the signal level at all of the receive antennas. Thus, in order to maximize the throughput in multiple receive antenna systems, multi-layer beamforming is required. In point-to-point systems, precoding means that multiple data streams are emitted from the transmit antennas with independent and appropriate weightings such that the link throughput is maximized at the receiver output. In multi-user MIMO, the data streams are intended for different users (known as SDMA) and some measure of the total throughput (e.g., the sum performance) is maximized. In point-to-point systems, some of the benefits of precoding can be realized without requiring channel state information at the transmitter, while such information is essential to handle the co-user interference in multi-user systems.

**Precoding for Point-to-Point MIMO Systems**

In point-to-point multiple-input multiple-output (MIMO) systems, a transmitter equipped with multiple antennas communicates with a receiver that has multiple antennas. Most classic precoding results assume narrowband, slowly fading channels, meaning that the channel for a certain period of time can be described by a single channel matrix which does not change faster. In practice, such channels can be achieved, for example, through OFDM. The precoding strategy that maximizes the throughput, called channel capacity, depends on the channel state information available in the system.

**Statistical channel state information**

If the receiver knows the channel matrix and the transmitter has statistical information, Eigen beamforming is known to achieve the MIMO channel capacity. In this approach, the transmitter emits multiple streams in Eigen directions of the channel statistics. As the actual channel realization is unknown at transmitter, interference will appear between the streams.

**Full channel state information**

If the channel matrix is completely known, singular value decomposition (SVD) precoding is known to achieve the MIMO channel capacity. In this approach, the channel matrix is diagonalzed by taking an SVD and removing the two unitary matrices through pre- and post-multiplication at the transmitter and receiver, respectively. Then, one data stream per singular value can be transmitted (with appropriate power loading) without creating any interference whatsoever.

**Precoding for Multi-user MIMO Systems**

In multi-user MIMO, a multi-antenna transmitter communicates simultaneously with multiple receivers (each having one or multiple antennas). This is known as space-division multiple access (SDMA). From an implementation perspective, precoding algorithms for SDMA systems can be sub-divided into linear and nonlinear precoding types. The capacity achieving algorithms are nonlinear, but linear precoding approaches usually achieve reasonable performance with much lower complexity. Linear precoding strategies include MMSE precoding and the simplified zero-forcing (ZF) precoding. There are also precoding strategies tailored for low-rate feedback of channel state information, for example random beamforming. Nonlinear precoding is designed based on the concept of dirty

paper coding (DPC), which shows that any known interference at the transmitter can be subtracted without the Precoding 2 penalty of radio resources if the optimal precoding scheme can be applied on the transmit signal. While performance maximization has a clear interpretation in point-to-point MIMO, a multi-user system cannot simultaneously maximize the performance for all users. Thus, it is common to maximize the weighted sum capacity, where the weights correspond to user priorities. In addition, there might be more users than data streams, requiring a scheduling algorithm to decide which users to serve at a given time instant.

**Linear precoding with full channel state information**

This suboptimal approach cannot achieve the weighted sum capacity, but it can still maximize the weighted sum performance. Optimal linear precoding is known as MMSE precoding and is simple to characterize for single-antenna receivers; the precoding weights for a given user are selected to maximize a ratio between the signal gain at this user and the interference generated at other users (with some weights) plus noise. Thus, precoding means finding the optimal balance between achieving strong signal gain and limiting co-user interference. Finding the optimal MMSE precoding is often difficult, leading to approximate approaches that concentrate on either the numerator or denominator of the mentioned ratio; that is, maximum ratio transmission (MRT) and zero-forcing (ZF) precoding. MRT only maximizes the signal gain at the intended user. MRT is close-to-optimal in noise-limited systems, where the co-user interference is negligible compared to the noise. ZF precoding aims at nulling the co-user interference, at the expense of losing some signal gain. ZF precoding can achieve close to the system capacity when the number of users is large or the system is interference-limited (i.e., the noise is weak compared to the interference). If receivers have multiple antennas, then regularized zero-forcing precoding has the corresponding properties.

**Linear precoding with limited channel state information**

In practice, the channel state information is limited at the transmitter due to estimation errors and quantization. Inaccurate channel knowledge may result in significant loss of system throughput, as the interference between the multiplexed streams cannot be completely controlled. In closed-loop systems, the feedback capabilities decide which precoding strategies that is feasible. Each receiver can either feedback a quantized version of its complete channel knowledge or focus on certain critical performance indicators (e.g., the channel gain). If the complete channel knowledge is fed back with good accuracy, then one can use strategies designed for having full channel knowledge with minor performance degradation. Zero-forcing precoding may even achieve the full multiplexing gain, but only provided that the accuracy of the channel feedback increases linearly with signal-to-noise ratio (in dB). Quantization and feedback of channel state information is based on vector quantization, and codebooks based on Grassmannian line packing have shown good performance. Other precoding strategies have been developed for the case with very low channel feedback rates. Random beamforming (or opportunistic beamforming) was proposed as a simple way of achieving good performance that scales like the sum capacity when the number of receivers is large. In this suboptimal strategy, a set of beamforming weights are selected randomly and users feed back a few bits to tell the transmitter which beam that gives the best performance and what rate they can support using it. When the number of users is large, it is likely that each random beamforming weight will provide good performance for some user. In spatially correlated environments, the long-term channel statistics can be combined with low-rate feedback to perform SDMA precoding. As spatially correlated statistics contain much directional information, it is only necessary for users to feed back their current channel gain to achieve reasonable channel knowledge. As the beamforming weights are selected from the statistics, and not randomly, this approach outperforms random beamforming under strong spatial correlation.

**2.2.3 SPECTRUM SHARING**

According to conventional wisdom, we currently suffer from a shortage of spectrum. This supposedly limits our ability to introduce new wireless products and services such as ubiquitous broadband Internet access, limits our ability to make current systems like cellular telephony more common and less expensive, limits our ability to increase the data rates and ranges of existing products like wifi, and even limits our ability to provide firefighters, police, and paramedics with the communications systems they need to do their jobs.

In actuality, if one measures spectrum utilization (as CMU students have), it is clear that much of the spectrum sits idle at any given time. One reason is that we often prevent interference between systems by giving each system exlusive access to a block of spectrum. Thus, whenever such a system is not transmitting, spectrum sits idle. In this project, we seek new methods that allow disparate wireless systems to share spectrum without causing excessive harmful interference to their neighbors. Our goal is to increase the amount of communications that can take place in a given amount of spectrum by orders of magnitude, which would lead to a revolution in wireless products and services.

**2.2.4 RAYLEIGH AND RICIAN FADING CHANNELS**

Rayleigh fadingis a statistical model for the effect of a propagation environment on a radio signal, such as that used by wireless devices. Rayleigh fading models assume that the magnitude of a signal that has passed through such a transmission medium (also called a communications channel) will vary randomly, or fade, according to a Rayleigh distribution — the radial component of the sum of two uncorrelated Gaussian random variables. Rayleigh fading is viewed as a reasonable model for tropospheric and ionospheric signal propagation as well as the effect of heavily built-up urban environments on radio signals. Rayleigh fading is most applicable when there is no dominant propagation along a line of sight between the transmitter and receiver. If there is a dominant line of sight, Rician fading may be more applicable.

The requirement that there be many scatterers present means that Rayleigh fading can be a useful model in heavily built-up city centers where there is no line of sight between the transmitter and receiver and many buildings and other objects attenuate, reflect, refract, and diffract the signal. Experimental work in Manhattan has found near-Rayleigh fading there. In tropospheric and ionospheric signal propagation the many particles in the atmospheric layers act as scatterers and this kind of environment may also approximate Rayleigh fading. If the environment is such that, in addition to the scattering, there is a strongly dominant signal seen at the receiver, usually caused by a line of sight, then the mean of the random process will no longer be zero, varying instead around the power-level of the dominant path. Such a situation may be better modelled as Rician fading.

Rician fadingis a stochastic model for radio propagation anomaly caused by partial cancellation of a radio signal by itself — the signal arrives at the receiver by several different paths (hence exhibiting multipath interference), and at least one of the paths is changing (lengthening or shortening). Rician fading occurs when one of the paths, typically a line of sight signal, is much stronger than the others. In Rician fading, the amplitude gain is characterized by a Rician distribution. Rayleigh fading is the specialized model for stochastic fading when there is no line of sight signal, and is sometimes considered as a special case of the more generalized concept of Rician fading. In Rayleigh fading, the amplitude gain is characterized by a Rayleigh distribution.

**2.3 MODULES NAME**

* System model
* Ml estimation of arrival time and its performance
* Proposed glrt detector
* Performance evaluation and discussions
* Detection performance
* Analysis of computational complexity

**2.4 MODULES EXPLANATION**

**2.4.1 SYSTEM MODEL**

Subtracting the common parts from the left-hand and right hand sides of the inequalities of the aforementioned events gives



E1 and E2 are independent because they concern different sets of random variables.



The joint distribution of Z = (Z1, . . . ,Zn0−j)T can be shown to obey a density function as



and g0,0 = 1. Notably, q·,· and g·,· can be iteratively derived. Likewise, the probability for ˆn0 to be located on the righthand side of n0 with an offset j, j > 0, is written as

**Proposed GLRT detector**

Based on ML estimation, when ˆn0 = n0, all useful signals of length N −n0 can be used for detection, and the proposed detector offers the same performance as the benchmark. When ˆn0 6= n0, there are two distinct cases defined as follows. 1) ˆn0 = n0 − j, where j > 0. In addition to N − n0 useful samples, j unwanted noise samples will also be utilized for detection. 2) ˆn0 = n0+j, where j > 0. The number of useful samples reduces to N −n0 −j. The pdf under H1 hypothesis can be expressed as

The proposed energy detection algorithm can be expressed as



where N−ˆn0 denotes the decision threshold and is a function of the number of available samples, N − ˆn0, used for detection. Thus, the GLRT detector under H0 hypothesis obeys a distribution defined by

**Performance evaluation and discussions**

Monte Carlo simulations were conducted to assess the performance of the proposed detector. The total number of observations is N = 160



**Estimation Performance**

The performance P(ˆn0 | n0) of the proposed ML estimation with arrival time n0 = N/2 = 80, plotted as a function of sampling point ˆn0 for ⇠ = 0 dB and −5 dB. P(ˆn0|n0) has the maximum value at n0. In addition, analytical results match to the simulation results very well. The estimation performance is critical for the proposed detector, because only when n0 = ˆn0, all useful samples can be employed for detection to achieve the optimal detection performance.

**Detection Performance**

Probability of detection Pd versus SNR for the proposed GLRT detector and conventional ED (under perfect synchronization) with Pfa = 0.1. The arrival time is assumed to be uniformly distributed over 0  n0  159. The conventional ED under perfect synchronization is used as a benchmark. As shown in the figure, the analytical results (Ana.) match to the simulation results (Sim.) very well. When SNR increases, Pd of the proposed detector improves consistently, and its performance can approach to that of the benchmark. The estimation is important for the proposed detector. when there exists a fixed synchronization error of 30 samples, the performance of the GLRT detector (Sync. Err.) degrades significantly. When N = 160 samples are used, the performance of the GLRT detector using the estimate (Est. Variance) degrades only slightly, the receiver operation characteristics (ROC), i.e., Pd, which was plotted as a function of Pfa of the proposed GLRT detector and conventional ED (under perfect synchronization). As displayed in the figure, the performance of the GLRT detector can still approach to that of the benchmark consistently for any Pfa. This plot further confirms the performance of the proposed detector under both H0 and H1.

**Analysis of Computational Complexity:**

Since the computational complexity of an addition, complex conjugation, or comparison operation is much lower than that of a multiplication operation, only the number of multiplications is calculated for simplicity. The proposed GLRT detector needs two steps, i.e., estimation and detection. The detection part is similar to a conventional ED. The magnitude squared in the decision metric (25) can reuse that obtained in the estimation (which will be evaluated later). Besides, the decision threshold can be implemented using a look-up table function by the argument (Pfa, ˆn0). Hence, the complexity for the detection can be omitted

the complexity of a complex multiplication needs three times more calculations than that of a real multiplication, the total complexity of the GLRT detector roughly requires 3N +2N +N = 6N real multiplications and three divisions; whereas the simplest ED (under perfect synchronization) requires 3N real multiplications (or N complex multiplications) for the magnitude squared of each sample. Therefore, the proposed technique is in particular suitable for real-time operation.

**CHAPTER 3**

**SOFTWARE SPECIFICATION**

**3.1 GENERAL**

**MATLAB 7.14**

Matlab (Matrix Laboratory) is a high-performance language for scientific and technological calculations. It integrates computation, visualization and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. It is a complete environment for high-level programming, as well as interactive data analysis. Some typical applications are

* system simulations,
* algorithm development,
* data acquisition, analysis, exploration, and visualization, as well as
* Modeling, simulation and prototyping.

Matlab was originally designed as a more convenient tool (than BASIC, FORTRAN or C/C++) for the manipulation of matrices. It was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. After- wards, it gradually became the language of general scientific calculations, visualization and program design. Today, Matlab engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computations. It received more functionalities and it still remains a high-quality tool for scientific computation. Matlab excels at numerical computations, especially when dealing with vectors or matrices of data. It is a procedural language, combining an efficient programming structure with a bunch of predefined mathematical commands. While simple problems can be solved interactively with Matlab, its real power is its ability to create large program structures which can describe complex technical as well as non-technical systems. Mat- lab has evolved over a period of years with input from many users. In university environments, it is the standard computational tool for introductory and advanced courses in mathematics, engineering and science. In industry, Matlab is the tool of choice for highly-productive research, development and analysis.

This tutorial script summarizes the tasks and experiments done during the seminar Matlab for Communications offered by the Department of Communication Systems of the university Duisburg-Essen. This seminar gives the students the opportunity to get first in touch with Matlab and further to have background knowledge about the simulation of communication systems. After a detailed introduction describing the main usage as well as the different definitions in Matlab, some relevant selected topics, like amplitude modulation, fast Fourier transformation or convolution, are treated.

**3.2 FEATURES OF MATLAB**

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

* Math and computation
* Algorithm development
* Modeling, simulation, and prototyping
* Data analysis, exploration, and visualization
* Scientific and engineering graphics
* Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

**3.3 The MATLAB System**

The MATLAB system consists of five main parts:

* **Development Environment**.

This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

* **The MATLAB Mathematical Function Library**.

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

* **The MATLAB Language**.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

* **Handle Graphics**.

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

* **The MATLAB Application Program Interface (API).**

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

**3.4 DESKTOP TOOLS**

This section provides an introduction to MATLAB's desktop tools. You can also use MATLAB functions to perform most of the features found in the desktop tools. The tools are:

* Current Directory Browser
* Workspace Browser
* Editor/Debugger
* Command Window
* Command History
* Help Browser
* **Command History**

Lines you enter in the Command Window are logged in the Command History window. In the Command History, you can view previously used functions, and copy and execute selected lines. To save the input and output from a MATLAB session to a file, use the diary function.

* **Command Window**

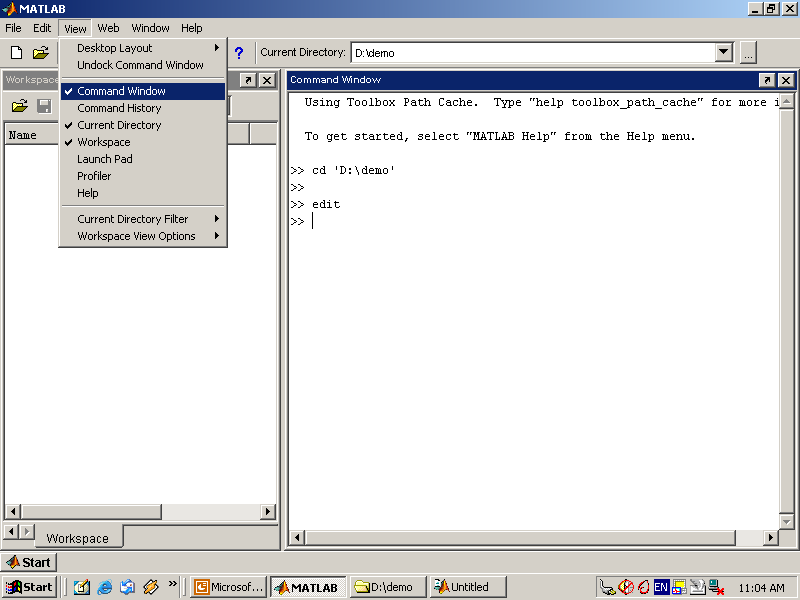


Fig.2 COMMAND WINDOW

* **Running External Programs**

You can run external programs from the MATLAB Command Window. The exclamation point character! is a shell escape and indicates that the rest of the input line is a command to the operating system. This is useful for invoking utilities or running other programs without quitting MATLAB. On Linux, for example,!emacs magik.m invokes an editor called emacs for a file named magik.m. When you quit the external program, the operating system returns control to MATLAB.

* **Launch Pad**

MATLAB's Launch Pad provides easy access to tools, demos, and documentation.

* **Help Browser**

Use the Help browser to search and view documentation for all your Math Works products. The Help browser is a Web browser integrated into the MATLAB desktop that displays HTML documents.

To open the Help browser, click the help button in the toolbar, or type help browser in the Command Window. The Help browser consists of two panes, the Help Navigator, which you use to find information, and the display pane, where you view the information.

* **Help Navigator**

Use to Help Navigator to find information. It includes:

* **Product filter**

Set the filter to show documentation only for the products you specify.

* **Contents tab**

View the titles and tables of contents of documentation for your products.

* **Index tab**

Find specific index entries (selected keywords) in the Math Works documentation for your products.

* **Search tab**

Look for a specific phrase in the documentation. To get help for a specific function, set the Search type to Function Name.

* **Favorites tab**

View a list of documents you previously designated as favorites.

* **Print pages**

Click the print button in the toolbar.

* **Display Pane**

After finding documentation using the Help Navigator, view it in the display pane. While viewing the documentation, you can:

* **Browse to other pages**

Use the arrows at the tops and bottoms of the pages, or use the back and forward buttons in the toolbar.

* **Bookmark pages**

Click the Add to Favorites button in the toolbar.

* **Find a term in the page**

Type a term in the Find in page field in the toolbar and click Go.

* **Current Directory Browser**

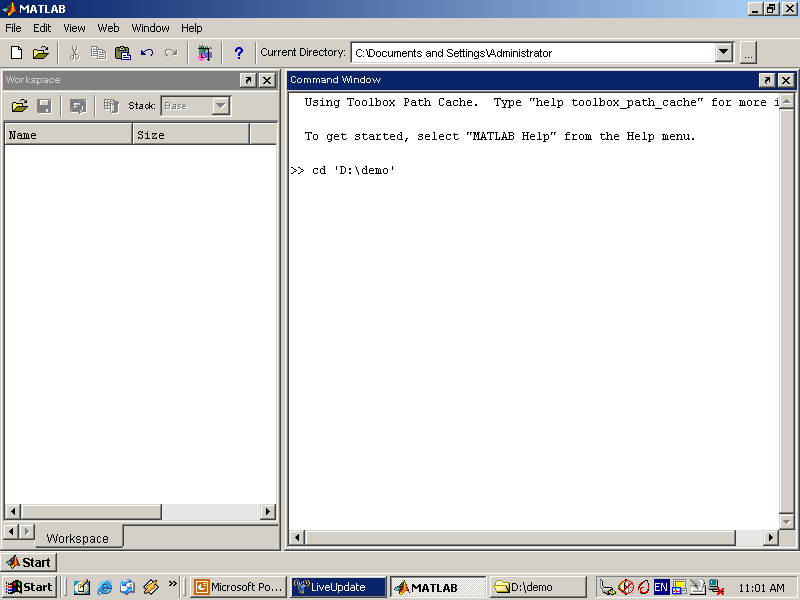
****

Fig.3 CURRENT DIRECTORY BROWSER

MATLAB file operations use the current directory and the search path as reference points. Any file you want to run must either be in the current directory or on the search path.

* **Search Path**

To determine how to execute functions you call, MATLAB uses a search path to find M-files and other MATLAB-related files, which are organized in directories on your file system. Any file you want to run in MATLAB must reside in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and Math Works toolboxes are included in the search path.

* **Workspace Browser**

The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. You add variables to the workspace by using functions, running M-files, and loading saved workspaces.

To delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the clear function.

The workspace is not maintained after you end the MATLAB session. To save the workspace to a file that can be read during a later MATLAB session, select Save Workspace As from the File menu, or use the save function. This saves the workspace to a binary file called a MAT-file, which has a .mat extension. There are options for saving to different formats. To read in a MAT-file, select Import Data from the File menu, or use the load function.

* **Array Editor**

Double-click on a variable in the Workspace browser to see it in the Array Editor. Use the Array Editor to view and edit a visual representation of one- or two-dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace. If you just need to view the contents of an M-file, you can display it in the Command Window by using the type function.

* **Editor/Debugger**

Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for basic textediting, as well as for M-file debugging.

You can use any text editor to create M-files, such as Emacs, and can use preferences (accessible from the desktop File menu) to specify that editor as the default. If you use another editor, you can still use the MATLAB Editor/Debugger for debugging, or you can use debugging functions, such as dbstop, which sets a breakpoint.

If you just need to view the contents of an M-file, you can display it in the Command Window by using the type function.

MATLAB supports the entire data analysis process, from acquiring data from external devices and databases, through preprocessing, visualization, and numerical analysis, to producing presentation-quality output.

**3.5 DATA ANALYSIS**

MATLAB is an efficient platform for accessing data from files, other applications, databases, and external devices. You can read data from popular file formats, such as Microsoft Excel; ASCII text or binary files; image, sound, and video files; and scientific files, such as HDF and HDF5. Low-level binary file I/O functions let you work with data files in any format. Additional functions let you read data from Web pages and XML.

MATLAB provides interactive tools and command-line functions for data analysis operations, including:

* Expressions
* Visualizing Data
* Handling matrices
* Curve fitting
* Matrix analysis
* Analyzing and accessing data

Like most other programming languages, Matlab provides mathematical expressions, but unlike most programming languages, these expressions involve entire matrices. The building blocks of expressions are

* Variables
* Numbers
* Operators
* Functions

**3.5.1 Variables**

Matlab does not require any type declarations or dimension statements. When a new variable name is introduced, it automatically creates the variable and allocates the appropriate amount of memory. If the variable already exists, Matlab changes its contents and, if necessary, allocates new storage.

For example

>> books = 10

Creates a 1-by-1 matrix named books and stores the value 10 in its single element. In the expression above, >> constitutes the Matlab prompt, where the commands can be entered.

Variable names consist of a string, which start with a letter, followed by any number of letters, digits, or underscores. Matlab is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are not the same variable. To view the matrix assigned to any variable, simply enter the variable name.

**3.5.2 Numbers**

Matlab uses the conventional decimal notation. A decimal point and a leading plus or minus sign is optional. Scientific notation uses the letter e to specify a power-of-ten scale factor. Imaginary numbers use either i or j as a suffix. Some examples of legal numbers are:

7 -55 0.0041 9.657838 6.10220e-10 7.03352e21 2i -2.71828j 2e3i 2.5+1.7j.

**3.5.3 Operators**

Expressions use familiar arithmetic operators and precedence rules. Some examples are:

+ Addition

- Subtraction

\* Multiplication

/ Division

’ Complex conjugate transpose

**3.5.4 Functions**

Matlab provides a large number of standard elementary mathematical functions, including sin, sqrt, exp, and abs. Taking the square root or logarithm of a negative number does not lead to an error; the appropriate complex result is produced automatically. Matlab also provides a lot of advanced mathematical functions, including Bessel and Gamma functions. Most of these functions accept complex arguments. For a list of the elementary mathematical functions, type

>> help elfun

Some of the functions, like sqrt and sin are built-in. They are a fixed part of the Matlab core so they are very efficient. The drawback is that the computational details are not readily accessible. Other functions, like gamma and sinh, are implemented in so called M-files. You can see the code and even modify it if you want.

**3.6 VISUALIZING DATA**

All the graphics features that are required to visualize engineering and scientific data are available in MATLAB. These include 2-D and 3-D plotting functions, 3-D volume visualization functions, tools for interactively creating plots, and the ability to export results to all popular graphics formats. You can customize plots by adding multiple axes; changing line colors and markers; adding annotation, Latex equations, and legends; and drawing shapes.

**3.6.1 2-D Plotting**

Visualizing vectors of data with 2-D plotting functions that create:

* Line, area, bar, and pie charts.
* Direction and velocity plots.
* Histograms.
* Polygons and surfaces.
* Scatter/bubble plots.
* Animations.

**3.6.2 3-D Plotting and Volume Visualization**

MATLAB provides functions for visualizing 2-D matrices, 3-D scalar, and 3-D vector data. You can use these functions to visualize and understand large, often complex, multidimensional data. Specifying plot characteristics, such as camera viewing angle, perspective, lighting effect, light source locations, and transparency.

3-D plotting functions include:

* Surface, contour, and mesh.
* Image plots.
* Cone, slice, stream, and iso-surface.

**3.7 HANDLING MATRICES**

Matlab was mainly designed to deal with matrices. In Matlab, a matrix is a rectangular array of numbers. So scalars can be interpreted to be 1-by-1 matrices and vectors are matrices with only one row or column. Matlab has other ways to store both numeric and nonnumeric data, but in the beginning of learning Matlab, it is usually best to think of everything as a matrix. The operations in Matlab are designed to be as natural as possible. Where other programming languages work only with single numbers, Matlab allows working with entire matrices quickly and easily.

**3.7.1 Entering Matrices and Addressing the Elements**

The elements of a matrix must be entered one-by-one in a list where the elements of a row must be separated with commas or blank spaces and the rows are divided by semicolons.

The whole list must be surrounded with square brackets, e.g.

>> A = [1 2 3; 8 6 4; 3 6 9]

After pressing “Enter” Matlab displays the numbers entered in the command line

A = 1 2 3

8 6 4

3 6 9

Addressing an element of a matrix is also very easy. The n-th element of the m-th column in matrix A from above is A(n,m). So typing

>> A(1,3) + A(2,1) + A(3,2)

will compute the answer ans = 17

The k-th to l-th elements of the m-th to n-th columns can be addressed by A(k:l,m:n),

Further examples:

>> A(2:3,1:2)

ans = 8 6

3 6

>> A(1,1:2)

addresses the first two elements of the first row.

ans = 1 2

>> A(:,2)

addresses all elements of the second column.

ans = 8

6

4

**3.7.2 Generating Matrices**

There are different ways to generate matrices. Assigning elements explicitly was presented in the paragraph above. To create a row vector with 101 equidistant values starting at 0 , this method would be very tedious. So two other possibilities are shown below:

>> x = linspace(0,pi,101) or x = (0:0.01:1)\*pi

In the first case, the Matlab function linspace is used to create x. The function’s arguments are described by:

linspace(first value, last value, number of values)

With the default number of values = 100.

In the second case, the colon notation (0:0.01:1) creates an array that starts at 0, increments by 0.01 and ends at 1. Afterwards each element in this array is multiplied by π to create the desired values in x.

Both of these array creation forms are common in Matlab. While the colon notation form allows to specify the increment between data elements directly, but not the number of data elements, the Matlab function linspace allows to specify the number of data elements directly, but not the increment value between these data elements.

Example

>> v = (10:-2:0)

v = 10 8 6 4 2 0

If the increment is 1, then its usage is optional:

>> w = (5:10)

w = 5 6 7 8 9 10

Matlab also provides four functions that generate basic matrices: zeros, ones, rand and randn.

Some more examples:

>> B = zeros(3,4)

B = 0 0 0 0

0 0 0 0

0 0 0 0

>> C = ones(2,5)\*6

C = 6 6 6 6 6

6 6 6 6 6

>> D = rand(1,5) generates uniformly distributed random elements

D = 0.5028 0.7095 0.4289 0.3046 0.1897

>> E = randn(3,3) generates normally -also called Gaussian- distributed random elements

E = -0.4326 0.2877 1.1892

-1.6656 -1.1465 -0.0376

0.1253 1.1909 0.3273

**3.7.3 Concatenation**

Concatenation is the process of joining small matrices to make bigger ones. In fact, the first matrix A was created by concatenating its individual elements. The pair of square brackets, [ ], is the concatenation operator. For an example, start with the 3-by-3 matrix A, and form

>> F = [A A+10; A\*2 A\*4].

The result is a 6-by-6 matrix, obtained by joining the four sub matrices.

F = 1 2 3 11 12 13

8 6 4 18 16 14

3 6 9 13 16 19

2 4 6 4 8 12

16 12 8 32 24 16

6 12 19 12 24 38

**3.7.4 Deleting rows and columns**

To delete rows or columns of a matrix, just use a pair of square brackets, e.g.

>> A(2,:) = [ ]

Deletes the second row of A.

A = 1 2 3

3 6 9

It is not possible to delete a single element of a matrix, because afterwards it would not still be a matrix. (Exception: vectors, since here deleting an element are the same as deleting a row/column.)

**3.7.5 Array Orientation**

The orientation of an array can be changed with the Matlab transpose operator ’:

>> a = 0:3

a = 0 1 2 3

>> b = a’

b = 0

1

2

3

**3.7.6 Scalar-Array Mathematics**

Addition, subtraction, multiplication and division by a scalar apply the operation to all elements of the array:

>> c = [1 2 3 4;5 6 7 8;9 10 11 12]

c = 1 2 3 4

5 6 7 8

9 10 11 12

>> 2\*c-1 multiplies each element in c by two and subtracts one from each element of the result.

ans = 1 3 5 7

9 11 13 15

17 19 21 23

**3.7.7 Array-Array Mathematics**

When two arrays have the same dimensions, which means that they have the same number of rows and columns, addition, subtraction, multiplication and division apply on an element-by-element basis in Matlab.

Examples

>> d = [1 2 3; 4 5 6]

d = 1 2 3

4 5 6

>> e = [2 2 2; 3 3 3]

e = 2 2 2

3 3 3

>> f = d+e adds d to e on an element-by-element basis

f = 3 4 5

7 8 9

>> g = 2\*d-e multiplies d by two and subtracts e from the result

g = 0 2 4

5 7 9

Element-by-element multiplication and division work similarly, but the notation is slightly different:

>> h = d.\*e

h = 2 4 6

12 15 18

The element-by-element multiplication uses the dot multiplication symbol.\*, the element-by-element array division uses either. / or.\

>> d./e

ans = 0.500 1.000 1.500

1.333 1.666 2.000

>> e.\d

ans = 0.500 1.000 1.500

1.333 1.666 2.000

In both cases, the elements of the array in front of the slash are divided by the elements of the array behind the slash. To compute a matrix multiplication only the asterisk \* must be used, e.g.

>> C = A \* B

Therefore the number of columns of A must equal the number of rows of B.

>> A = [1 2 3; 4 5 6]

A = 1 2 3

4 5 6

>> B = [1 2; 3 4; 5 6]

1 2

B = 3 4

5 6

>> C = A \* B

C = 22 28

49 64

**3.8 CURVE FITTING**

Matlab offers extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. This section describes some of the most important graphics functions and gives some examples of some typical applications.

**3.8.1 Creating a Plot**

The plot function has different forms, depending on the input arguments. If y is a vector, plot(y) produces a piecewise linear graph of the elements of y versus the index of the elements of y. If two vectors are specified as arguments, plot(x,y) produces a graph of y versus x. For example to plot the value of the sine function from zero to 2π, use

>> x = 0:pi/100:2\*pi;

>> y = sin(x);

>> plot(x,y)



Fig.4 Sine Plot

The xlabel, ylabel and zlabel functions are useful to add x-, y- and z-axis labels. The zlabel function is only necessary for three-dimensional plots. The title function adds a title to a graph at the top of the figure and the text function inserts a text in a figure.

The following commands create the final appearance of figure 1.1.

>> xlabel(’x’);

>> ylabel(’y’);

>> title(’y = sin(x)’)

Multiple x-y pairs create multiple graphs with a single call to plot. Matlab automatically cycles through a predefined (but user settable) list of colors to distinguish between different graphs.

For example, these statements plot three related functions of x1, each curve in a separate distinguishing color:

>> x1 = 0:pi/100:2\*pi;

>> y1 = sin(x1);

>> y2 = sin(x1 - 0.25);

>> y3 = sin(x2 - 0.5);

>> plot(x1,y1,x1,y2,x1,y3)

The number of points of the individual graphs may be even different. It is possible to specify the color, the line style and the markers, such as plus signs or circles, with: plot(x,y,’color style marker’)



Fig.5 Multiple graphs with a single call to plot

A color style marker is a 1-, 2-, or 3-character string. It may consist of a color type, a line style type, and a marker type:

* Color strings are ’c’, ’m’, ’y’, ’r’, ’g’, ’b’, ’w’ and ’k’. These correspond to cyan, magenta, yellow, red, green, blue, white, and black.
* Line style strings are ’-’ for solid, ’--’ for dashed, ’:’ for dotted, ’-.’ for dash-dotted and ’none’ for no line.
* The most common marker types include ’+’, ’o’, ’\*’ and ’x’.

For example, the statement plot(x1,y1,’b:\*’) plots a blue dotted line and places asterisk sign markers at each data point. If only a marker type is specified but not a line style, Matlab draws only the marker.

The plot function automatically opens a figure window to plot the graphic. If there is already an existing figure window, these windows will be used for the new plot. The command figure can be used to keep an existing figure window and open a new one, which will be used for the next plot. To make an existing window the current window, type *figure(n)* where n is the number in the title bar of the window to be selected. The next graphic will be plotted in this selected window.

To add further plots to an existing graph, the hold command is useful. The hold on command keeps the content of the figure and plots can be added. So the above example could be done with three single plot commands and the hold on command. *hold off* ends the hold on status of a figure window. hold can be used to toggle between on and off.

**3.8.2 Controlling Axes**

Usually, Matlab finds the maxima and minima of the data to be plotted by it and uses them to create an appropriate plot box and axes labeling. The axis function overwrites this default by setting custom axis limits,

>> axis([xmin xmax ymin ymax]).

The following example illustrates the use of the functions presented above.

>> t = -pi:pi/100:pi;

>> s = cos(t);

>> plot(t,s)

>> axis([-pi pi -1 1])

>> xlabel(’-\pi \leq t \leq \pi’)

>> ylabel(’cos(t)’)

>> title(’The cosine function’)

>> text(-2, -0.5,’This is a note at position (-2, -0.5)’)

****

Fig.6 Example for controlling the axes

To take a closer look at an interesting part of a plot, the zoom command can be used. Afterwards it is possible to zoom by marking this part with the mouse. The grid command is used to turn a grid on and off.

**3.9 FLOW CONTROL**

Computer programming languages offer possibilities to allow the programmer to control the flow of command execution. This flow control is based on decision making structures. Some of the most important structures are the for-loop, the while-loop and the if-else-end-structure. Since the constructions often affect several Matlab commands, they are mostly used in M-files (see also 1.6). The for-loop repeats a group of statements a fixed, predetermined number of times.

>> for x = array

commands...

end

The commands... between for and the end statements are executed one time for every column in array. For example

>> for k = 1:10

z(k) = 2 \* i;

end

>> z

z = [ 2 4 6 8 10 12 14 16 18 20 ]

It is a good idea to indent the loops for readability, especially when they are nested.

>> for l = 1:5

for m = 1:8

H(l,m) = 1/(l+m);

end

end

While the for-loop evaluates a group of commands a fixed number of times, a while-loop evaluates a group of statements and indefinite number of times. The general form of a while-loop is

>> while expression

commands...

end

The commands... between the while and the end statements are executed as all elements in expression are true (nonzero). For example

>> a = 1; b = 10;

>> x = 1:10;

>> while (a =< b)

z(a) = x(b-a);

a = a+1;

end

>> z

z = [ 10 9 8 7 6 5 4 3 2 1 ]

Many times, sequences of commands must be conditionally evaluated. In Matlab this is provided by the if-else-end construction.

>> if expression

commands1...

else

commands2...

end

The commands1... between the if and the else statements are evaluated if all elements in expression are true (nonzero). Alternatively the commands2... between the else and the end statements are executed. For example

>> a = 5;

>> if a > 0

c = 2\*a;

else

c = -2\*a;

end

>> c

c = 10

**3.10 VECTOR MANIPULATIONS**

In Matlab applications it is often necessary to create vectors, to delete elements of given vectors, to substitute elements in given vectors, to append vectors or to insert a given vector into another vector. All these methods are discussed in this short tutorial. The corresponding m-files are available.

**3.10.1 Basic Operations**

With the aim to visualize generated vectors, the first thing to do is to define figures and perhaps subfigures. This is done as follows:

%This m-file demonstrates basic MATLAB line vector manipulations

clear all %=Clear all variables

close all %=close all windows

figure(1);

subplot(211); %upper subfigure

grid on;

box on

hold on;

subplot(212); %lower subfigure

grid on;

box on

hold on;

In the first two lines all variables are cleared and all figures are closed.

Then figure 1 is defined and displayed with an upper and a lower sub-screen, a bounding box and a grid. With “hold on” all plots are kept on the screen.

The command subplot(m,n,p), or subplot(mnp), breaks the Figure window into an m-by- n matrix of small axes and selects the p-th axes for the current plot. Thus, subplot(211) creates two rows and one column of subplots in the current figure and selects the upper subfigure for plotting. Correspondingly, a command subplot(224) for example would create 2 rows and 2 columns of subplots in the current figure and selects the lower right

subfigure for plotting.

Notice, that all text strings preceded by a % character or all lines starting with a % character are comment lines. Thus, the following strings are not interpreted as Matlab commands

In the next lines some line vectors are defined and plotted:

%After we have defined some general screen settings we start here

%with the demonstration program.

%Methods to define a line vector

%define a line vector with 10 zero elements

a0=zeros(1,10)

%define a line vector with 10 one elements

a1=ones(1,10)

%Defintion of a line vector element by element

a=[0 1 2 3 4 5 6 7 8 9 10];

%same vector using the linspace definition

b=linspace(0,10,11)

%same vector using another definition

b=[0:1:10] or b=[0:10]

In the lines above, several methods to define a line vector are given, i.e. defining a vector with all zero elements or a vector with all 1 elements or the definition of a vector element by element. The ”linespace(x, y, z)” method defines linearly spaced vector elements with x the first element, with y the last element and with z the number of elements. Whereas the [first:incr:last] method defines a row vector with first element the increment value and the last vector element. If no increment value is given, 1 is assumed as increment.

%visualization of the vector a

subplot(211);

plot(a);

title(’Plot of vector a’);

with the lines above, the vector a is plotted to the upper sub-window. Notice, that the horizontal axes of the plot(a) command is automatically scaled by the index of the vector a, which starts always with 1 for the first element and ends with the N-th element, i.e. from 1 to N=11 in this case.

In the following lines the command “fliplr” is used, which arranges all vector elements in reverse order and the corresponding vector is plotted to the lower sub-window.

%arrange the vector elements in reverse order

c=fliplr(a);

%visualization of the vector c

subplot(212);

plot(c);

title(’Plot of vector c’);

pause

How to join two line vectors into a new one (appending vectors) is shown in the following few lines. Moreover, a title line is added to the sketch in the lower sub-window. Hold off is used to overwrite the previous plot in the lower sub-window.

hold off;

%appending vectors

d=[a c];

plot(d);

title(’Plot of vector d’);

grid on;

%which is a trapezoid function

pause;

How to delete single or multiple subsequent elements of a line vector is shown in the following lines. Before and after carrying out this operation give the following command to the command window: “length(c)”, which confirms, that the length of the vector is reduced by one.

%Delete the first element of vector c

c(1)=[];

%accordingly we can delete elements 3..6 of vector c using

%c(3:6)=[];

%if we now redefine vector d we get a triangle

d=[a c];

plot(d);

title(’Plot of vector d’);

grid on;

pause;

delete 2 elements starting at the center of vector d

d(11:12)=[];

plot(d);

title(’Plot of vector d’);

grid on;

pause;

%Now we delete more elements from pos 7 to pos 14 and again get a triangle

d(7:14)=[];

subplot(211);

hold off;

plot(d);

title(’Plot of vector d’);

grid on;

The following lines show how to substitute vector elements by other vector elements. To avoid calculations we can use either the “length()” of the vector to be substituted or the “size()” definition. Notice, that the size() operation refers always to the matrix dimensions of variables, because all variables are in principle considered to be matrices. Hence, a line vector is a 1 by N matrix. Thus, “size(d,1)” is the number of rows = 1 here, “size(d,2)” is the number of columns (elements of the vector).

%inserting vectors into other vectors

%first we generate a zero line vector with 30 elements

e=zeros(1,30);

%and insert the previously defined vector d at position 10

%all of the following 3 lines carry out the same operation

e(1,10:1:20)=d

e(1,10:1:10+length(d)-1)=d

e(1,10:1:10+size(d,2)-1)=d

plot(e);

title(’Plot of vector e’);

grid on;

pause;

**3.10.2 Mathematical Operations on Vectors**

The operations discussed in the following are usually carried out on all elements of the vector(s).

%Now calculate the sum of all elements of the vector e

fprintf(’Sum of e is: %3.2f\n’,sum(e));

%Now determine the norm of the vector e

fprintf(’norm of e is sum(e.\*e): %3.2f or %3.2f\n’,sum(e.\*e),e\*e’);

Notice, that sum(e.\*e) is the same operation as e\*e’. Whereas the point multiplication

e.\*e carries out the multiplication element by element of the vector e, the e\*e’ operation is in principle a matrix multiplication. If e is a line vector, then e’ is the transposed e-vector, i.e. a column vector and e\*e’ is the scalar quantity sum(e.\*e) =, with ei the vector elements of vector e.

The fprintf() command used above creates a formatted string. The principle construction rule of this command is well known from standard programming languages like C, FORTRAN etc.

t=fprintf(’text %format string(s)\n’, variable(s));

where t is a string variable (which may be omitted), the text string for output and the format string(s) and perhaps a line break must be enclosed in apostrophes, the format string(s) always start with a % character followed by the total number of characters to be printed, the . separator which separates the integer values from the fractions and the number of characters to be printed after the . separator. The character f, which follows in the format string defines the floating point format. \n carries out one line break. Separated by a comma, the variable(s) follow(s) in the same order and with as many variables (or expressions) as format strings are defined.

Greek characters within the text string of fprintf() are generated by the LATEX-notation starting with a \followed by the name of the Greek character; like for example \Omega, \omega or \Gamma, \gamma for the corresponding upper- and lower case characters.

%square the elements of vector e or build any power of the

%vector elements of e.

f=e.\*e

f=e.^2

subplot(212);

plot(f);

title(’Plot of vector f’);

grid on;

e.\*e and e.ˆ 2 show the same results but e.ˆ x is more universal because x may be any power of the elements of vector e and x can be any real number.

**3.10.3 Natural Functions of Vectors**

%Natural functions of vectors a plot(sin(a));

%The plot shows a polygon, which does not resemble

%a sine function very much, because the horizontal

%spacing is not fine enough.

%The simple plot(sin(a)) command always connects the precise values

%of the vector components of sin(a) with straight lines. Thus, the plot()

%command always creates a polygon. However, if we create a vector with much

%narrower spacing the corresponding plot result resembles a sine function.

clear all;

a=linspace(0,2\*pi,101);

plot(sin(a)); %This is still plotted against the vector index

grid;

pause;

plot(a,sin(a)); %Now sin(a) is plotted versus vector a

pause;

set(gca,’Xlim’,[0,2\*pi]);

grid;

The plot of natural functions requires usually a much more narrow spacing of the horizontal axis. If we use for example our original vector a with eleven elements and plot sin(a), then we get a polygon which does not very much resemble the sine function.

If we clear all vectors, redefine vector a with 100 intervals between 0 and 2\_ and plot it we get a much better resolution.

The notation plot(a,sin(a)) plots sin(a) versus vector a, which is now a properly scaled function.

“set(gca,’Xlim’,[0,2\*pi]);” sets the entire horizontal range of the plot.

“set(gca,’Ylim’,[-2,2]);” would set the vertical range.

Alternatively, we could use the axis() command to fix the vertical and horizontal scale of the plot, like for example:

axis([0,2\*pi,-2,2]) i.e. axis([xmin,xmax,ymin,ymax])

Matlab provides all basic mathematical functions like all trigonometric and exponential functions and even more specific functions like Bessel, Gamma, Beta functions etc. Type “help elfun” or “help specfun” to the command window to find the notation for all the build in functions.

**3.11 BASIC PLOTTING**

**3.11.1 Plots with basic functionality**

Plot two sine functions with frequency f1 = 200 Hz and f2 = 50 Hz and amplitudes A1 = 1 V and A2 = 1.5 V within the time interval t=[0,50] ms in the same figure. Use the sampling period T = 10μs.

|  |  |
| --- | --- |
| f1=200; f2=50; T=1e-5; | Parameter settings... |
| Tst=0; Te=5e-2; A1=1; A2=1.5 |  |
| t=[Tst:T:Te]; | A vector is defined containing needed sampling time instants. |
| y1=A1\*sin(2\*pi\*f1\*t); | The first output vector is calculated. |
| y2=A2\*sin(2\*pi\*f2\*t); | The second output vector is calculated. |
| fig1=figure; | A figure is opened and its handle is named ’fig1’. |
| plot(y1); | Plots the first output vector versus its indices and connects subsequent points with lines. |
| plot1=plot(t,y1); | Plots the first output vector versus input vector. Since we want to change the appearance of this plot later, we need to define the handle ’plot1’. |
| xlab=xlabel(’t (in s)’); | Displays label on x-axis. |
| ylab=ylabel(’y (in V)’); | Displays label on y-axis. |
| ti=title(’Sine Functions’); | Displays title above the figure. |
| hold on; | The active figure will not be overwritten by the next plot command.  Note: ’hold off’ will deactivate the effect; ’hold’ toggles between ’hold on’ and ’hold off’. |
| plot2=plot(t,y2,’r--’); | Plots the second output vector versus input vector.  Note: ’plot(t,y2,’g–’)’ defines color (here: green) and line  style (here: dash dot) manually; for further information type ’help plot’. |
| grid on; | Activates grid of both x- and y-axis (syntax like ’hold’). |
| axis([Tst,Te,-1.7,1.7]); | axis([xmin xmax ymin ymax]) controls axis scaling. |
| set(plot1,’linewidth’,’2’); | Changes linewidth of first curve.  Note: The same syntax can be used to change other Para-  meters like color, font type and font size of labels etc.; a list  of parameters can found in Matlab help. |
| set(plot2,’linewidth’,’2’); | Changes linewidth of second curve. |
| leg=legend([plot1,plot2],’A1=1 V, f1=200 Hz’,’A2=1.5 V, f2=50 Hz’); | Displays a legend for the curves with handles ’plot1’ and  ’plot2’. |
| set(ti,’Fontsize’,13); | Changes the font size of the title. |

Table 1: Basic Plotting Function

**3.11.2 Plots using ‘subplot’**

Plot the same sine functions into two subfigures by using the command ‘subplot’.

|  |  |
| --- | --- |
| f1=200; ... fig1=figure; | same as in 1.1 |
| sub1=subplot(211); | ‘subplot(nr,nc,counter)’ plots a subfigure within the active figure. The figure is divided into ’nr’ rows and ’nc’ columns and the last parameters determines the position of the subfigure (the numbering is row-wise and 1≤counter≤Nr\*Nc). |
| plot1=plot(t,y1); | same as in 1.1 ... |
| xlab1=xlabel(’t (in s)’); |  |
| ylab1=ylabel(’y (in V)’); |  |
| set(plot1,’linewidth’,2); |  |
| grid on; |  |
| axis([Tst,Te,-1.7,1.7]); |  |
| sub2=subplot(212); | Activates the second subfigure (below the first one). |
| leg1 = legend([plot1],’A1=1 V, f1=200 Hz’);  leg2 = legend([plot2],’A2=2 V, f2=50 Hz’); | Adds legend for the first and second subplot. |

Table 2: Subplot

**3.11.3 PROBLEMS**

**3.11.3.1 Sampling of ’continuous’ waveform**

Given is the Matlab routine ‘[s]=waveform(t)’ which generates the periodical signal s(t) with period T = 20 ms. It is sampled in the time interval t=[0,65]ms with four different sampling frequencies *f1* = 100Hz, *f2* = 250Hz, *f3* = 500Hz and *f4* = 1000Hz. Display in four subfigures the ’continuous’ waveform and the corresponding sampled waveform, respectively. Use for the sampled waveform the command ’stem’. Moreover, consider following properties of the figures:

|  |  |
| --- | --- |
| General Font Name of subplots: | ‘Times’ |
| General Font Size of subplots: | 8 |
| x/y-Label: | Time (in s) / Amplitude (in V) |
| Font Name of labels: | ‘Times’ |
| Font Size of labels: | 11 |
| Linestyle/-color/-width of ’continuous’ waveform: | dotted / RGB=[0.4,0.9,0.3] / 2 |
| Linestyle/-color/-width of sampled waveforms: | solid,’.’ / blue / 1 |
| Axis scaling: | x=[0,65]\*1e-3, y=[-1.1,1.1] |
| Legend: | ‘continuous’, ‘sampled’ |
| Grid: | on |

Table 3: Properties

**3.11.3.2 Simple 3D-Plots**

Plot a two-dimensional Gaussian probability density function *f*x,y(x, y) with zero mean and variance 1 within the range x  [−5, 5], y  [−5, 5]:



(4)

**3.12. RANDOM FUNCTIONS**

**3.12.1. Continuous Uniform Distribution**

Uniformly distributed, continuous random numbers are to be generated and their histograms are to be discussed.

|  |  |
| --- | --- |
| a=rand(1,100) | rand(x,y) generates a matrix with x rows and y columns, whose elements are uniformly distributed within the range [0,1]. |
| hist(a); | hist(a) plots the histogram. The value range of a is divided into 10 intervals. The number of elements in these intervals is plotted. |
| a=rand(1,1e5); |  |
| hist(a,100); | hist(a,N) plots the histogram. The value range of a is divided into  N intervals. The number of elements in these intervals is plotted. |

Table 4: Uniform Distribution

Mostly, random numbers not uniformly distributed within [0,1] but within an arbitrary range [a,b] are needed:

|  |  |
| --- | --- |
| a=rand(1,1e5); |  |
| hist(a,100); | a −→ a is RV  [0, 1] |
| a=rand(1,1e5)+0.5; |  |
| hist(a,100); | a −→ a is RV  [0.5, 1.5] |
| a=rand(1,1e5)+n; | a −→ a is RV [n, n+1] |
| a=2\*rand(1,1e5); |  |
| hist(a,100); | a −→ a is RV  [0, 2] |
| a=A\*rand(1,1e5); | a −→ a is RV  [A·0, A·1] |
| a=A\*rand(1,1e5)+N; | a −→ a is RV  [N, A+N] |

**3.12.2. Discrete Uniform Distribution**

Now, discrete uniformly distributed random numbers shall be generated.

|  |  |
| --- | --- |
| a=floor(N\*rand(x,y)+1); | This command generates a matrix with x rows and y columns, whose elements are uniformly distributed integer numbers within the range [1...N]. |
| a=floor(20\*rand(1,1e5)+1); |  |
| hist(a) | In order to see the functionality of this command, the standard resolution of hist is not sufficient. |

**3.12.3. Gaussian Distribution**

In the following, continuous, Gaussian distributed random numbers are generated and their histograms plotted.

|  |  |
| --- | --- |
| b=randn(1,1e5); | randn(x,y) generates a matrix with x rows and y columns, whose elements are normally distributed (Gaussian, zero mean, variance 1). |
| hist(b,100) | Histogram |
| x=sqrt(5)\*randn(1,1e5)+1; | Zero Mean and variance 1 is changed into mean 1 and var 5. Therefore, x is Gaussian distributed now. |
| hist(x,100) | Histogram |

Table 5: Gaussian Distribution

**CHAPTER 4**

**SIMULATION RESULTS**

**4.1 CODING DESIGN**

clc

clear all

close all

N = 160 ;%% The total number of observations

n0=N/2;%% ML estimation with arrival time

H0=1:n0-1; % hypotisis 0

H1=rand(1,1)+(1:n0-1);% hypotisis 1

sigma=var(H0);

n=ones(1,79);

sigmas=var(H1);

mu=mean(H0);

SNR=0;% db

no1=linspace(60,100,mu);

x(n)=H0;

pd=norm(0,N-1);

% gamma function

k=10;

g=gamma(N-n0+k\*pd);

% threshold

pfa=0.1;

sig=distributed(qfunc(ps\*N-n0)\*pfa)+1;

%probability detection

beta=2^-1;

cw=sigma\*(gamma(N-n0+(1\*N-n0./beta')))./gamma(N-n0+k);

k=10;b=6;a=12.7135;

e=linspace(0.100,0.5 ,26);

m=(gamma(a).\*gamma(b))./(gamma(a+b)\*sig);

snr=-20:1:5;

H0=pi\*x.^2\*(N-n0); % GLRD detector

for i=1:length(snr)

x(i)=k\*(1-e(i))./(k-1);

c=@(z)(1)\*(z.^(a-1)).\*((1-z).^(b-1));

d(i)=integral(c,1,x(i));

p=-(d./m)';

p=0.11+p;

end

hold on

plot(snr,(p),'k','LineWidth',1.5)

%% % GLRT DETECTOR vs variances

N = 160-1 ;%% The total number of observations

n0=N/2;%% ML estimation with arrival time

H0=1:n0-1; % hypotisis 0

H1=rand(1,1)+(1:n0-1);% hypotisis 1

sigma=var(H0);

n=ones(1,79);

sigmas=var(H1);

mu=mean(H0);

SNR=0 ;% db

no1=linspace(60,100,mu);

E=sigma/sigmas;

% pd

pd=norm(0,N-1);

% gamma function

k=10;

g=gamma(N-n0+k\*pd);

% threshold

pfa=0.1;

sig=distributed(qfunc(ps\*N-n0)\*pfa)+1;

%probability detection

beta=2^-1;

cw=sigma\*(gamma(N-n0+(1\*N-n0./beta')))./gamma(N-n0+k);

k=10;b=6;a=12.7135;

e=linspace(0.100,0.49 ,26);

m=(gamma(a).\*gamma(b))./(gamma(a+b)\*sig);

snr=-20:1:5;

H0=pi\*x.^2\*(N-n0); % GLRD detector

for i=1:length(snr)

x(i)=k\*(1-e(i))./(k-1);

c=@(z)(1)\*(z.^(a-1)).\*((1-z).^(b-1));

d(i)=integral(c,1,x(i));

p=-(d./m)';

p=0.11+p;

end

hold on

plot(snr,(p),'--k','LineWidth',1.5)

%% proposed GLRT

N = 160-1 ;%% The total number of observations

n0=N/2;%% ML estimation with arrival time

H0=1:n0-1; % hypotisis 0

H1=rand(1,1)+(1:n0-1);% hypotisis 1

sigma=var(H0);

n=ones(1,79);

sigmas=var(H1);

mu=mean(H0);

SNR=0 ;% db

no1=linspace(60,100,mu);

E=sigma/sigmas;

% pd

pd=norm(0,N-1);

% gamma function

k=10;

g=gamma(N-n0+k\*pd);

% threshold

pfa=0.1;

sig=distributed(qfunc(ps\*N-n0)\*pfa)+1;

%probability detection

beta=2^-1;

cw=sigma\*(gamma(N-n0+(1\*N-n0./beta')))./gamma(N-n0+k);

k=10;b=6;a=12.7135;

e=linspace(0.123,0.47 ,26);

m=(gamma(a).\*gamma(b))./(gamma(a+b)\*sig);

snr=-20:1:5;

H0=pi\*x.^2\*(N-n0); % GLRD detector

for i=1:length(snr)

x(i)=k\*(1-e(i))./(k-1);

c=@(z)(1)\*(z.^(a-1)).\*((1-z).^(b-1));

d(i)=integral(c,1,x(i));

p=-(d./m)';

p=0.11+p;

end

hold on

plot(snr,(p),'-dk','LineWidth',1.5)

grid on

legend('ED with perfect sync ','proposed GLRT an','proposed GLRT simu','proposed GLRT Est var','proposed GLRT sync','Location','SouthEast')

xlabel('{\xi} (dB)')

ylabel('Pd')

%% figure 3

figure

%% ROC plotted as a function

Pf =linspace(0,1,12);

N=160;

n0=N/2;%% ML estimation with arrival time

H0=1:n0-1;

L = length(H0);

E = -8; % SNR in decibels

snr = 10.^(E./10); % Linear Value of SNR

%% Simulation of Energy Detection

for m = 1:length(Pf)

i = 0;

for kk=1:10000 % Number of Monte Carlo Simulations

n = randn(1,L); %AWGN noise with mean 0 and variance 1

s = sqrt(snr).\*randn(1,L); % Real valued Gaussina Primary User Signal

y = s + n; % Received signal at SU

energy = abs(y).^2; % Energy of received signal over N samples

energy\_fin =(1/L).\*sum(energy);

thresh(m) = (qfuncinv(Pf(m))./sqrt(L))+ 1;

if(energy\_fin >= thresh(m))

i = i+1;

end

end

Pd(m) = i/kk;

end

plot(Pf, Pd,'-o')

hold on

%% proposed GLRT Detection

thresh = (qfuncinv(Pf)./sqrt(L))+1.02;

Pd\_the = qfunc(((thresh - (snr + 1)).\*sqrt(L))./(sqrt(2).\*(snr + 1)));

plot(Pf, Pd\_the, '-\*r')

hold on,grid on

legend('ED with perfect sync','proposed GLRT Detection','Location','SouthEast')

xlabel('Pfa')

ylabel('Pd')

%% figure 4

figure

N = 160 ;%% The total number of observations

n0=N/2;%% ML estimation with arrival time

H0=1:n0-1; % hypotisis 0

H1=rand(1,1)+(1:n0-1);% hypotisis 1

sigma=var(H0);

n=ones(1,79);

sigmas=var(H1);lamda=30;

mu=mean(H0)./lamda;

% ML estimation

x(n)=H0;

ee=-20:5:5;

c=log(1./(1+ee));

ohm=sigma.\*((norm(x(n)).^2)./4-sigmas.\*(norm(x(n)).^2)./4-n0.\*c);

% chi-squre distribution

e0=0.1;

c1=-2.\*(1+(1./e0)).\*c;

p=1:N-1;

% proposed energy detection algorithm

for n=1:80-1

ps=norm((ohm))./sigma;

end

% pd

pd=norm(0,N-1);

% gamma function

k=10;

g=gamma(N-n0+k\*pd);

% threshold

pfa=0.1;

sig=distributed(qfunc(ps\*N-n0)\*pfa)+1;

%probability detection

beta=2^-1;

cw=sigma\*(gamma(N-n0+(1\*N-n0./beta')))./gamma(N-n0+k);

k=10;b=6;a=12.7135;

e=linspace(0.130,0.5 ,26);

m=(gamma(a).\*gamma(b))./(gamma(a+b)\*sig);

snr=-20:1:5;

for i=1:length(snr)

x(i)=k\*(1-e(i))./(k-1);

c=@(z)(1)\*(z.^(a-1)).\*((1-z).^(b-1));

d(i)=integral(c,1,x(i));

p=-(d./m)';

p=pfa+p;

end

plot(snr,(p),'-ok','LineWidth',1.5)

%% GLRT DETECTOR

N = 160-1 ;%% The total number of observations

n0=N/2;%% ML estimation with arrival time

H0=1:n0-1; % hypotisis 0

H1=rand(1,1)+(1:n0-1);% hypotisis 1

sigma=var(H0);

n=ones(1,79);

sigmas=var(H1);

mu=mean(H0);

no1=linspace(60,100,mu);

E=sigma/sigmas;

% pd

pd=norm(0,N-1);

% gamma function

k=10;

g=gamma(N-n0+k\*pd);

% threshold

pfa=0.11;

sig=distributed(qfunc(ps\*N-n0)\*pfa)+1;

%probability detection

beta=2^-1;

cw=sigma\*(gamma(N-n0+(1\*N-n0./beta')))./gamma(N-n0+k);

k=10;b=6;a=12.7135;

e=linspace(0.100,0.5 ,26);

m=(gamma(a).\*gamma(b))./(gamma(a+b)\*sig);

snr=-20:1:5;

H0=pi\*x.^2\*(N-n0); % GLRD detector

for i=1:length(snr)

x(i)=k\*(1-e(i))./(k-1);

c=@(z)(1)\*(z.^(a-1)).\*((1-z).^(b-1));

d(i)=integral(c,1,x(i));

p=-(d./m)';

p=pfa+p;

end

hold on

plot(snr,(p),'+k','LineWidth',1.5)

ylim([0 1])

%% GLRT proposed simulation

%% GLRT DETECTOR

N = 160-1 ;%% The total number of observations

n0=N/2;%% ML estimation with arrival time

H0=1:n0-1; % hypotisis 0

H1=rand(1,1)+(1:n0-1);% hypotisis 1

sigma=var(H0);

n=ones(1,79);

sigmas=var(H1);

mu=mean(H0);

SNR=0 ;% db

no1=linspace(60,100,mu);

E=sigma/sigmas;

% pd

pd=norm(0,N-1);

% gamma function

k=10;

g=gamma(N-n0+k\*pd);

% threshold

pfa=0.11;

sig=distributed(qfunc(ps\*N-n0)\*pfa)+1;

%probability detection

beta=2^-1;

cw=sigma\*(gamma(N-n0+(1\*N-n0./beta')))./gamma(N-n0+k);

k=10;b=6;a=12.7135;

e=linspace(0.100,0.5 ,26);

m=(gamma(a).\*gamma(b))./(gamma(a+b)\*sig);

snr=-20:1:5;

H0=pi\*x.^2\*(N-n0); % GLRD detector

for i=1:length(snr)

x(i)=k\*(1-e(i))./(k-1);

c=@(z)(1)\*(z.^(a-1)).\*((1-z).^(b-1));

d(i)=integral(c,1,x(i));

p=-(d./m)';

p=pfa+p;

end

**4.2 SCREEN SHOTS**









**CHAPTER 5**

**CONCLUSION**

Spectrum sensing in asynchronous transmissions is important in many wireless communication systems, in particular for the studies of CR femtocell networks. Theoretical analysis on the proposed estimator and detector was given in this paper. The performance of the proposed GLRT detector was shown to approach to optimal performance, regardless the distribution of timing misalignment. In this sense, the proposed approach is practical and promising for real-time CR applications in the presence of unknown arrival times of the primary signals. Our future research effort will be dedicated to extending the current study to a more complex primary signal traffic model with random arrival and departure times.

**REFERENCES**

[1] D. Qu, Z. Wang, and J. Tao, “Extended active interference cancellation for sidelobe suppression in cognitive radio OFDM systems with cyclic prefix,” IEEE Trans. Vehicular Tech., vol. 59, no. 4, pp. 1689-1695, May 2010.

[2] X. Wang, “Joint sensing-channel selection and power control for cognitive radios,” IEEE Trans. Wireless Commun., vol. 10, no. 3, pp. 958-967, Mar. 2011.

[3] X. Huang, T. Han, and N. Ansari, ?On Green Energy Powered Cognitive Radio Networks,? IEEE Communications Surveys and Tutorials, DOI: 10.1109/COMST.2014.2387697, 2015.

[4] T. Han and N. Ansari, ?Enabling Mobile Traffic Offloading via Energy Spectrum Trading,? IEEE Transactions on Wireless Communications, vol. 13, no.6, pp. 3317-3328, June 2014.

[5] H. Sun, D. I. Laurenson, and C. X. Wang, “Computationally tractable model of energy detection performance over slow fading channels,” IEEE Commun. Lett., vol. 14, no. 10, pp. 924-926, Oct. 2010.

[6] M. Lopez-Benitz and F. Casadevall, “Improved energy detection spectrum sensing for cognitive radio,” IET Commun., vol. 6, no. 8, pp. 785-796, Jul. 2012.

[7] A. Bagwari and G. S. Tomar, “Adaptive double-threshold based energy detector for spectrum sensing in cognitive radio networks,” International J. Electron. Lett., vol. 1, no. 1, pp. 24-32, Apr. 2013

[8] T. Do and B. L. Mark, “Improving Spectrum Sensing Performance by Exploiting Multiuser Diversity,” Foundation of Cognitive Radio Systems, Prof. Samuel Cheng (Ed.), ISBN: 978-953-51-0268-7, Mar. 2012.

[9] W. L. Chin, C. W. Kao, H. H. Chen, and T. L. Liao, “Iterative Synchronization Assisted Detection of OFDM Signals in Cognitive Radio Systems,” IEEE Trans. Vehicular Tech., vol. 63, no. 4, pp. 1633-1644, May 2014.

[10] H. Claussen, “Performance of macro- and co-channel femtocells in a hierarchical cell structure,” Proc. IEEE PIMRC’07, pp. 1-5, Jul. 2007.

[11] D. Lopez, A. Valcarce, G. de la Roche, and J. Zhang, “OFDMA femtocells: a roadmap on interference avoidance,” IEEE Commun. Mag., vol. 47, no. 9, pp. 41-48, Sep. 2009.

[12] N. C. Beaulieu and Y. Chen, “Improved Energy Detectors for Cognitive Radios With Randomly Arriving or Departing Primary Users,” IEEE Signal Process. Lett., vol. 17, no. 10, pp. 867-870, Oct. 2010.

[13] J. Y. Wu, C. H. Wang, and T. Y. Wang, “Performance analysis of energy detection based spectrum sensing with unknown primary signal arrival time,” IEEE Trans. Commun., vol. 59, no. 7, pp. 1779-1784, Jul. 2011.

[14] S. M. Kay, Fundamental of Statistical Signal Processing vol. 2-detection theory, Prentice-Hall PTR, 1998.

[15] H. T. Hsieh and W. R. Wu, “Maximum likelihood timing and carrier frequency offset estimation for OFDM systems with periodic preambles,” IEEE Trans. Vehicular Tech., vol. 58, no. 8, pp. 4224-4237, Oct. 2009.

[16] X. Cai, Y. Wu, H. Lin, and K. Yamashita, “Estimation and compensation of CFO and I/Q imbalance in OFDM systems under timing ambiguity,” IEEE Trans. Vehicular Tech., vol. 60, no. 3, pp. 1200-1205, Mar. 2011.

[17] K. Bury, Statistical Distributions in Engineering, Cambridge University Press, 1999.

[18] P. G. Moschopoulos, “The distribution of the sum of independent Gamma random variables,” Springer Ann. Inst. Statist. Math., 37 (1985), Part A, pp. 541-544, 1985.