NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic Engineering

USER INTERFACE FOR AUDIO SIGNAL PROCESSING

EE 400

Graduation Project

Student: Ahmed Osman Nabil(20020501)

Supervisor: Dr. Ali Serener

Nicosia - 2008

ACKNOWLEDGMENTS

This work would not have been possible without the generous help of God and then the following people as well as their significant contribution to my work.

Dr. Ali SERENER: I would like to sincerely thank him for his invaluable supervision, support and encouragement through this work and for introducing me to the world of communications. Also, for his suggestions and his instructions through the under graduate years and for always being kind and helpful to me all these years.

I would like especially to express my sincere thanks and dedication to my parents and family and gift them this work for their always constant love and support, spiritual and financial, in my decisions through the years.

I wish to thank the administration of Near East University for making all this work possible.

Finally, I would like to thank my friends specially Tarig Fared and Murat Khader for their help in Matlab programming and their helpful ideas.

Ahmed Osman

TABLE OF CONTENTS

ABSTRACT

Ever since the beginning of communication the need for enhancement was obvious. During the beginning years of telecommunications quality of sound travelling through their various channels diminished so much by the time they reached their destination that many times either the information was not understood at all or a person's voice changed to a level where it was not recognizable no more. This happens due to a phenomenon known as noise. In the field of communications, noise is our worst enemy. This is why we try to eliminate it to a maximum degree. Sadly, it cannot be eliminated 100%. But we are getting close.

Within this project we will see what noise is and what it does to a simple .wav file, simulating that which might occur in real world cases where sound might travel over a channel that is quite noisy. This will be done with the GUI we are going to build using Matlab. We will study, and observe the effects of noise on a signal and see what happens during filtering.

INTRODUCTION

Signal processing provides the basic analysis, modelling and synthesis tools for a diverse area of technological fields, including telecommunication, artificial intelligence, biological computation and system identification. Signal processing is concerned with the modelling, detection, identification and utilisation of patterns and structures in a signal process. Applications of signal processing methods include audio hi-fi, digital TV and radio, cellular mobile phones, voice recognition, vision, radar, sonar, geophysical exploration, medical electronics, bio-signal processing and in general any system that is concerned with the communication or processing and retrieval of information. Signal processing theory plays a central role in the development of digital telecommunication and automation systems, and in the efficient transmission, reception and decoding of information.

Noise can be defined as an unwanted signal that interferes with the communication or measurement of another signal. Noise and distortion are the main factors limiting the capacity of data transmission in telecommunications and accuracy in signal measurement systems. Therefore the modeling and removal of the effects of noise and distortions have been at the core of the theory and practice of communications and signal processing. Noise reduction and distortion removal are important problems in applications such as cellular mobile communications, speech recognition, image processing, medical signal processing, radar and sonar, and in any application where the signals cannot be isolated from noise and distortion.

Speech enhancement in noisy environments, such as in cars, trains, streets and at noisy public venues, improves the quality and intelligibility of speech. Noise reduction benefits a wide range of applications, such as mobile phones, hands-free phones, teleconferencing, in-car cabin communication systems and automated speech recognition services. This chapter provides an overview of the main methods for single-input speech enhancement in noise. De-noising speech improves the quality and the intelligibility of voice communication in noisy environments and reduces communication fatigue. Noise reduction benefits the users of hands-free phones, mobile phones and voice-controlled automated services used in noisy moving environments such as cars, trains, streets, conference halls and other public venues. We present a brief overview of the speech enhancement problem for wide-band noise sources that are not

correlated with the speech signal. Our main focus is on the spectral subtraction approach and some of its derivatives in the forms of linear and non-linear minimum mean square error estimators. For the linear case, we review the signal subspace approach, and for the non-linear case, we review spectral magnitude and phase estimators. On line estimation of the second order statistics of speech signals using parametric and non-parametric models is also addressed.

Here in this project we will also look into something known as a graphical user interface, or GUI as we will call from here on. A user interface helps with making a program simpler for the end user. It removes the need for knowledge on language processing to perform required processing.

Chapter 1 begins with a definition of signals, and a brief introduction to various signal processing methodologies. We will also talk about several key applications of digital signal processing in adaptive noise reduction, channel equalisation, audio signal coding, signal detection, and Dolby noise reduction.

In chapter 2, we study the characteristics and modeling of several different forms of noise. These are done to get a better understanding of how we can later remove these noises if we know how they were formed.

In chapter 3 we will talk on speech enhancement in more detail. We will see some methods of removal of noise from audio signals.

Chapter 4 shows how the GUI was built o Matlab with easy to follow instructions and pictorial representation.

CHAPTER ONE

SIGNAL PROCESSING

1.1 Preview

Signal processing provides the basic analysis, modelling and synthesis tools for a diverse area of technological fields, including telecommunication, artificial intelligence, biological computation and system identification. Signal processing is concerned with the modelling, detection, identification and utilisation of patterns and structures in a signal process. Applications of signal processing methods include audio hi-fi, digital TV and radio, cellular mobile phones, voice recognition, vision, radar, sonar, geophysical exploration, medical electronics, bio-signal processing and in general any system that is concerned with the communication or processing and retrieval of information. Signal processing theory plays a central role in the development of digital telecommunication and automation systems, and in the efficient transmission, reception and decoding of information.

This chapter begins with a definition of signals, and a brief introduction to various signal processing methodologies. We consider several key applications of digital signal processing in adaptive noise reduction, channel equalisation, pattern classification/recognition, audio signal coding, signal detection, spatial processing for directional reception of signals and Dolby noise reduction.

1.2 Signals and Information

A signal is the variation of a quantity by which information is conveyed regarding the state, the characteristics, the composition, the trajectory, the evolution, the course of action or the intention of the information source. A signal is a means of conveying information regarding the state(s) of a variable.

The information conveyed in a signal may be used by humans or machines for communication, forecasting, decision-making, control, geophysical exploration, medical diagnosis, forensics, etc. The types of signals that signal processing deals with include textual data, audio, ultrasonic, subsonic, image, electromagnetic, medical, biological, financial and seismic signals.

Figure 1.1 illustrates a communication system composed of an information source, I(t), followed by a system, T[.] for transformation of the information into variation of a signal, x(t), a communication channel, h[.], for propagation of the signal from the transmitter to the receiver, additive channel noise, n(t), and a signal processing unit at the receiver for extraction of the information from the received signal.

In general, there is a mapping operation that maps the output, I(t), of an information source to the signal, x(t), that carries the information; this mapping operator may be denoted as T[.] and expressed as

$$x(t) = T[I(t)] \tag{1.1}$$

The information source I(t) is normally discrete-valued, whereas the signal x(t) that carries the information to a receiver may be continuous or discrete. For example, in multimedia communication the information from a computer, or any other digital communication device, is in the form of a sequence of binary numbers (ones and zeros), which would need to be transformed into voltage or current variations and modulated to the appropriate form for transmission in a communication channel over a physical link.

As a further example, in human speech communication the voice-generating mechanism provides a means for the speaker to map each discrete word into a distinct pattern of modulation of the acoustic vibrations of air that can propagate to the listener. To communicate a word, w, the speaker generates an acoustic signal realisation of the word, x(t); this acoustic signal may be contaminated by ambient noise and/or distorted by a communication channel, or impaired by the speaking abnormalities of the talker, and received as the noisy, distorted and/or incomplete signal y(t), modelled as

$$y(t) = h[x(t)] + n(t) \tag{1.2}$$

In addition to conveying the spoken word, the acoustic speech signal has the capacity to convey information on the prosody (i.e. pitch, intonation and stress patterns in pronunciation) of speech and the speaking characteristics, accent and emotional state of the talker. The listener extracts this information by processing the signal y(t).

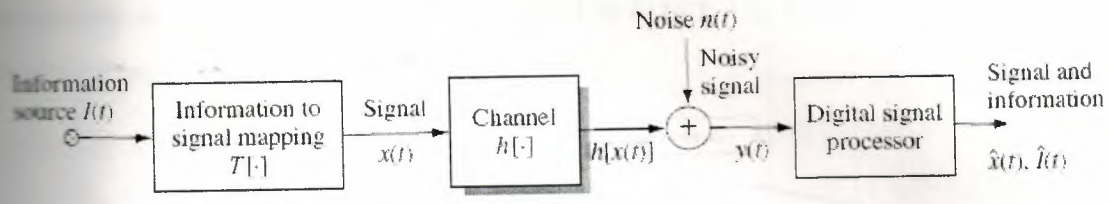


Figure 1.1 Illustration of a communication and signal processing system.

In the past few decades, the theory and applications of digital signal processing have evolved to play a central role in the development of modern telecommunication and information technology systems.

Signal processing methods are central to efficient communication, and to the development of intelligent man–machine interfaces in areas such as speech and visual pattern recognition for multimedia systems. In general, digital signal processing is concerned with two broad areas of information theory:

(1) efficient and reliable coding, transmission, reception, storage and representation of signals in communication systems; and

(2) extraction of information from noisy signals for pattern recognition, detection, forecasting, decision-making, signal enhancement, control, automation, etc.

In the next section we consider four broad approaches to signal processing.

1.3 Signal Processing Methods

Signal processing methods have evolved in algorithmic complexity, aiming for optimal utilisation of the information in order to achieve the best performance. In general, the computational requirement of signal processing methods increases, often exponentially, with the algorithmic complexity. However, the implementation cost of advanced signal processing methods has been offset and made affordable by the consistent trend in recent years of a continuing increase in the performance, coupled with a simultaneous decrease in the cost, of signal processing hardware.

Depending on the method used, digital signal processing algorithms can be categorised into one or a combination of four broad categories. These are transform-based signal processing, model-based signal processing, Bayesian statistical signal

processing and neural networks, as illustrated in Figure 1.2. These methods are briefly described below.

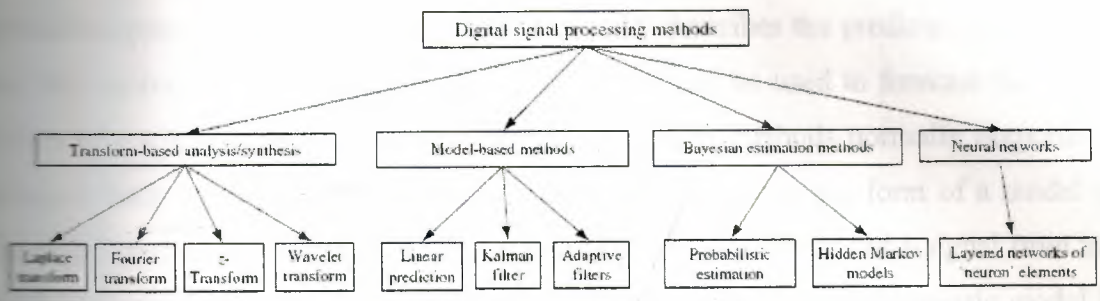


Figure 1.2 A broad categorisation of some of the most commonly used signal processing methods.

1.3.1 Transform-Based Signal Processing

The purpose of a transform is to describe a signal or a system in terms of a combination of a set of elementary simple signals (such as sinusoidal signals) that lend themselves to relatively easy analysis, interpretation and manipulation. Transform-based signal processing methods include Fourier transform, Laplace transform, z-transform and wavelet transforms. The most widely applied signal transform is the Fourier transform, which is effectively a form of vibration analysis, in that a signal is expressed in terms of a combination of the sinusoidal vibrations that make up the signal. Fourier transform is employed in a wide range of applications, including popular music coders, noise reduction and feature extraction for pattern recognition. The Laplace transform, and its discrete-time version the z-transform, are generalisations of the Fourier transform and describe a signal or a system in terms of a set of sinusoids with exponential amplitude envelopes. In Fourier, Laplace and z-transform, the different sinusoidal basis functions of the transforms all have the same duration and differ in terms of their frequency of vibrations and amplitude envelopes. In contrast, the wavelets are multi-resolution transforms in which a signal is described in terms of a combination of elementary waves of different durations. The set of basis functions in a wavelet is composed of contractions and dilations of a single elementary wave. This allows non-stationary events of various durations in a signal to be identified and analysed.

1.3.2 Model-Based Signal Processing

Model-based signal processing methods utilise a parametric model of the signal generation process. The parametric model normally describes the predictable structures and the expected patterns in the signal process, and can be used to forecast the future values of a signal from its past trajectory. Model-based methods normally outperform nonparametric methods, since they utilise more information in the form of a model of the signal process. However, they can be sensitive to the deviations of a signal from the class of signals characterised by the model. The most widely used parametric model is the linear prediction model. Linear prediction models have facilitated the development of advanced signal processing methods for a wide range of applications such as low-bit-rate speech coding in cellular mobile telephony, digital video coding, high-resolution spectral analysis, radar signal processing and speech recognition.

1.3.3 Bayesian Signal Processing

The fluctuations of a purely random signal, or the distribution of a class of random signals in the signal space, cannot be modelled by a predictive equation, but can be described in terms of the statistical average values, and modelled by a probability distribution function in a multidimensional signal space. For example, a linear prediction model driven by a random signal can provide a source-filter model of the acoustic realization of a spoken word. However, the random input signal of the linear prediction model, or the variations in the characteristics of different acoustic realisations of the same word across the speaking population, can only be described in statistical terms and in terms of probability functions.

The Bayesian inference theory provides a generalised framework for statistical processing of random signals, and for formulating and solving estimation and decision-making problems.

1.3.4 Neural Networks

Neural networks are combinations of relatively simple nonlinear adaptive processing units, arranged to have a structural resemblance to the transmission and processing of signals in biological neurons. In a neural network several layers of parallel

processing elements are interconnected by a hierarchically structured connection network. The connection weights are trained to perform a signal processing function such as prediction or classification.

Neural networks are particularly useful in nonlinear partitioning of a signal space, in feature extraction and pattern recognition and in decision-making systems. In some hybrid pattern recognition systems neural networks are used to complement Bayesian inference methods. Since the main objective of this project is to provide a coherent presentation of the theory and applications of statistical signal processing and how noise is formed and removed, neural networks are not discussed in this project.

1.4 Applications of Digital Signal Processing

In recent years, the development and commercial availability of increasingly powerful and affordable digital computers has been accompanied by the development of advanced digital signal processing algorithms for a wide variety of applications such as noise reduction, telecommunications, radar, sonar, video and audio signal processing, pattern recognition, geophysics explorations, data forecasting, and the processing of large databases for the identification, extraction and organisation of unknown underlying structures and patterns. Figure 1.3 shows a broad categorisation of some digital signal processing (DSP) applications.

1.4.1 Adaptive Noise Cancellation

In speech communication from a noisy acoustic environment such as a moving car or train, or over a noisy telephone channel, the speech signal is observed in an additive random noise.

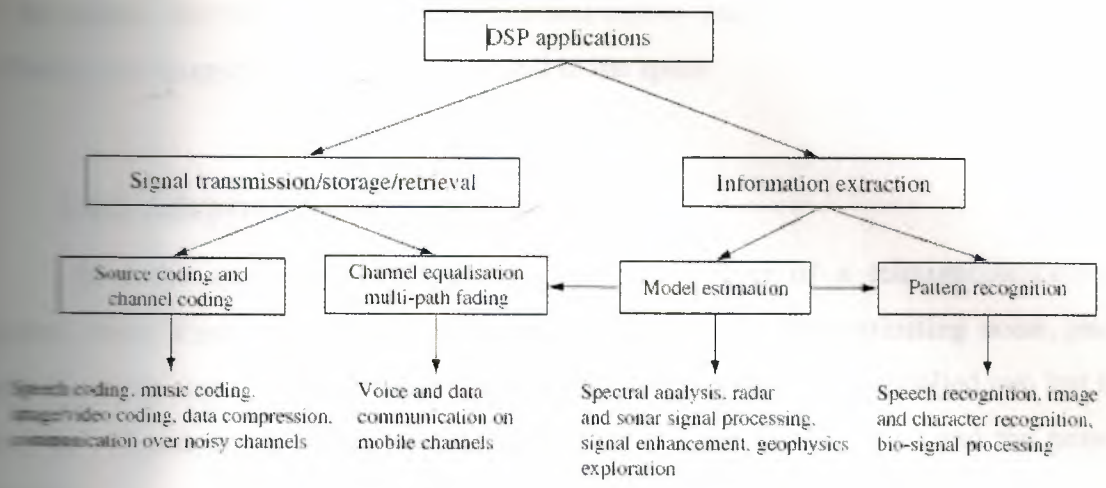


Figure 1.3 A classification of the applications of digital signal processing.

In signal measurement systems the information-bearing signal is often contaminated by noise from its surrounding environment. The noisy observation, y(m), can be modelled as

$$y(m) = x(m) + n(m) \tag{1.3}$$

where x(m) and n(m) are the signal and the noise, and m is the discrete-time index. In some situations, for example when using a mobile telephone in a moving car, or when using a radio communication device in an aircraft cockpit, it may be possible to measure and estimate the instantaneous amplitude of the ambient noise using a directional microphone. The signal, x(m), may then be recovered by subtraction of an estimate of the noise from the noisy signal.

Figure 1.4 shows a two-input adaptive noise cancellation system for enhancement of noisy speech. In this system a directional microphone takes as input the noisy signal x(m)+n(m), and a second directional microphone, positioned some distance away, measures the noise $\alpha n(m+T)$. The attenuation factor, α, and the time delay, T, provide a rather over-simplified model of the effects of propagation of the noise to different positions in the space where the microphones are placed. The noise from the second microphone is processed by an adaptive digital filter to make it equal to the noise contaminating the speech signal, and then subtracted from the noisy signal to cancel out the noise. The adaptive noise canceller is more effective in cancelling out the low-frequency part of the noise, but generally suffers from the nonstationary character

of the signals, and from the over-simplified assumption that a linear filter can model the diffusion and propagation of the noise sound in the space.

1.4.2 Adaptive Noise Reduction

In many applications, for example at the receiver of a telecommunication system, there is no access to the instantaneous value of the contaminating noise, and only the noisy signal is available. In such cases the noise cannot be cancelled out, but it may be reduced, in an average sense, using the statistics of the signal and the noise process.

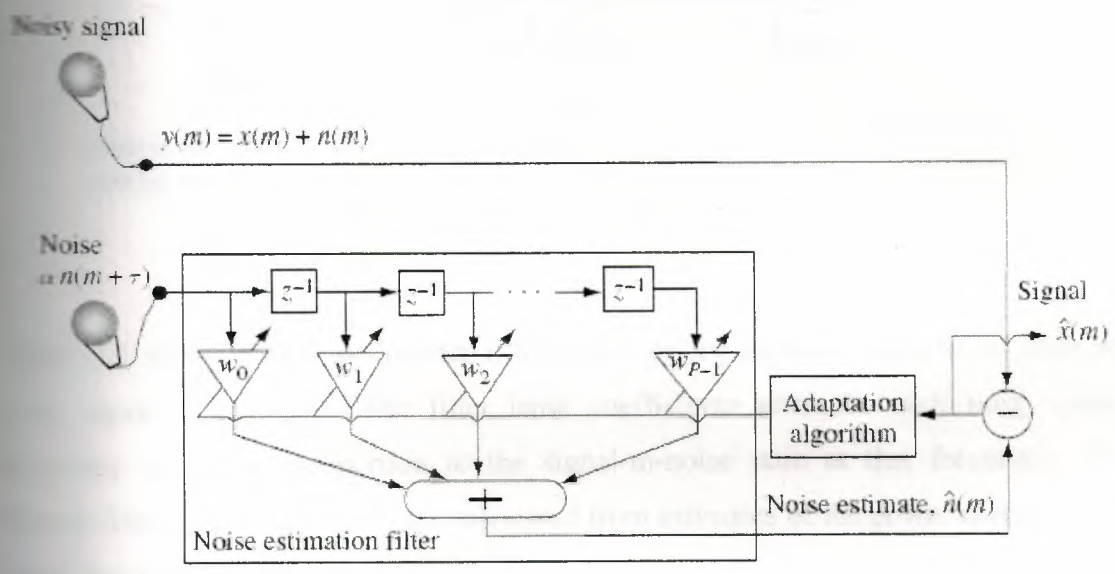


Figure 1.4 Configuration of a two-microphone adaptive noise canceller.

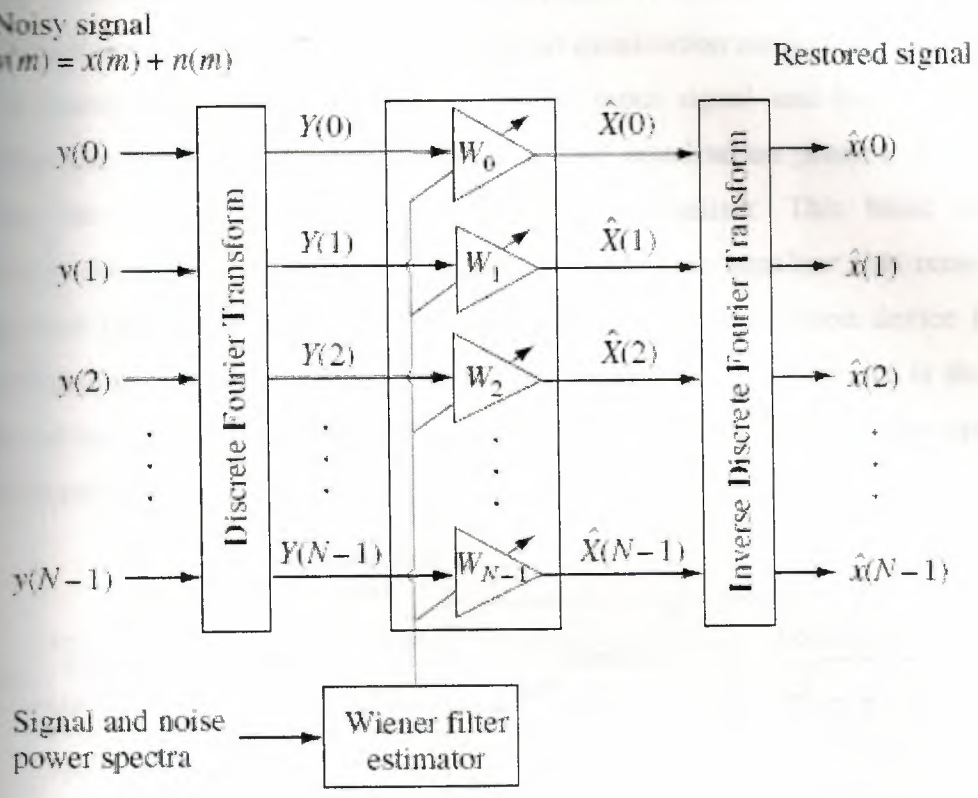


Figure 1.5 A frequency-domain Wiener filter for reducing additive noise.

Figure 1.5 shows a bank of Wiener filters for reducing additive noise when only the noisy signal is available. The filter bank coefficients attenuate each noisy signal frequency in inverse proportion to the signal-to-noise ratio at that frequency. The Wiener filter bank coefficients are calculated from estimates of the power spectra of the signal and the noise processes.

1.4.3 Blind Channel Equalisation

Channel equalisation is the recovery of a signal distorted in transmission through a communication channel with a nonflat magnitude or a nonlinear phase response. When the channel response is unknown, the process of signal recovery is called 'blind equalisation'. Blind equalisation has a wide range of applications, for example in digital telecommunications for removal of inter-symbol interference due to nonideal channel and multipath propagation, in speech recognition for removal of the effects of the microphones and communication channels, in correction of distorted images, in analysis of seismic data and in de-reverberation of acoustic gramophone recordings.

In practice, blind equalisation is feasible only if some useful statistics of the channel input are available. The success of a blind equalisation method depends on how much is known about the characteristics of the input signal and how useful this knowledge can be in the channel identification and equalisation process. Figure 1.6 illustrates the configuration of a decision-directed equaliser. This blind channel equaliser is composed of two distinct sections: an adaptive equaliser that removes a large part of the channel distortion, followed by a nonlinear decision device for an improved estimate of the channel input. The output of the decision device is the final estimate of the channel input, and it is used as the desired signal *to direct* the equaliser adaptation process.

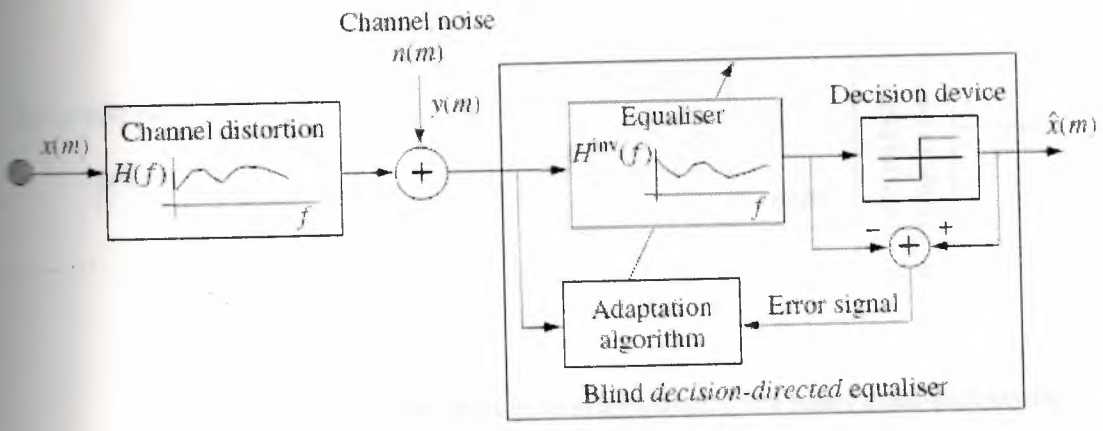


Figure 1.6 Configuration of a decision-directed blind channel equaliser.

1.4.4 Signal Classification and Pattern Recognition

Signal classification is used in detection, pattern recognition and decision-making systems. For example, a simple binary-state classifier can act as the detector of the presence, or the absence, of a known waveform in noise. In signal classification, the aim is to design a minimum-error system for *labelling* a signal with one of a number of likely classes of signal.

To design a classifier, a set of models is trained for the classes of signals that are of interest in the application. The simplest form that the models can assume is a bank, or code book, of waveforms, each representing the prototype for one class of signals. A

more complete model for each class of signals takes the form of a probability distribution function. In the classification phase, a signal is labelled with the nearest or the most likely class. For example, in communication of a binary bit stream over a band-pass channel, the binary phase-shift keying (BPSK) scheme signals the bit '1' using the waveform $A_c \sin \omega_c t$ and the bit '0' using $-A_c \sin \omega_c t$.

At the receiver, the decoder has the task of classifying and labelling the received noisy signal as a '1' or a '0'. Figure 1.7 illustrates a correlation receiver for a BPSK signalling scheme.

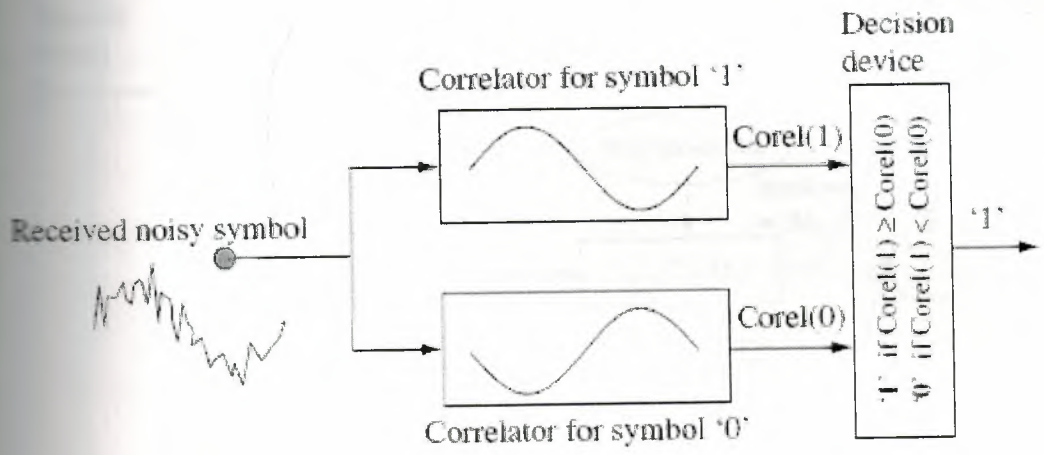


Figure 1.7 A block diagram illustration of the classifier in a binary phase-shift keying demodulation.

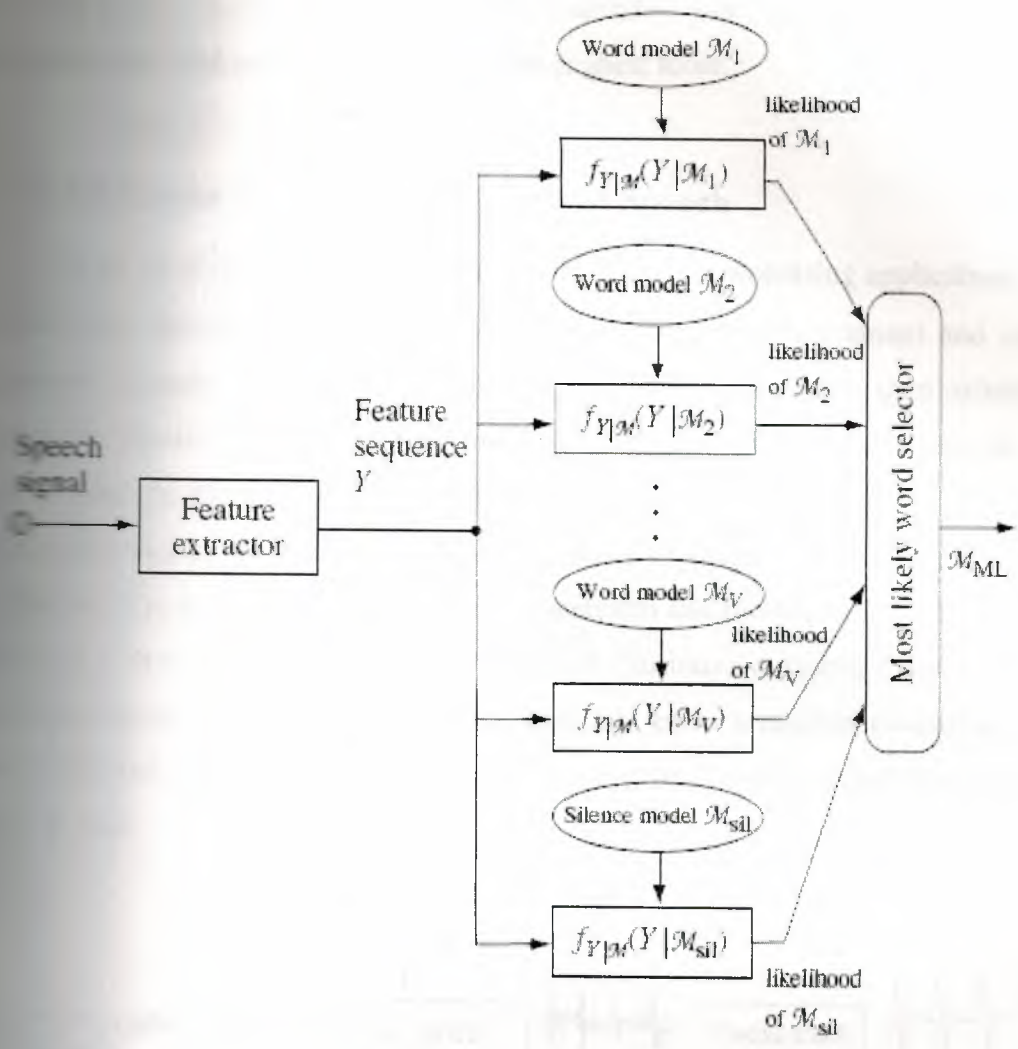


Figure 1.8 Configuration of a speech recognition system; f(Y | Mi) is the likelihood of the model Mi given an observation sequence *Y*.

The receiver has two correlators, each programmed with one of the two symbols representing the binary states for the bit '1' and the bit '0'. The decoder correlates the unlabelled input signal with each of the two candidate symbols and selects the candidate that has a higher correlation with the input.

Figure 1.8 illustrates the use of a classifier in a limited-vocabulary, isolated-word speech recognition system. Assume there are V words in the vocabulary. For each word a model is trained, on many different examples of the spoken word, to capture the average characteristics and the statistical variations of the word. The classifier has access to a bank of V+1 models, one for each word in the vocabulary and an additional model for the silence periods. In the speech-recognition phase, the task is to decode and label an acoustic speech feature sequence, representing an unlabelled spoken word, as

one of the V likely words or silence. For each candidate word the classifier calculates a probability score and selects the word with the highest score.

1.4.5 Linear Prediction Modelling of Speech

Linear predictive models are widely used in speech processing applications such as low-bit-rate speech coding in cellular telephony, speech enhancement and speech recognition. Speech is generated by inhaling air into the lungs, and then exhaling it through the vibrating glottis cords and the vocal tract. The random, noise-like, air flow from the lungs is spectrally shaped and amplified by the vibrations of the glottal cords and the resonance of the vocal tract. The effect of the vibrations of the glottal cords and the vocal tract is to introduce a measure of correlation and predictability to the random variations of the air from the lungs. Figure 1.9 illustrates a source-filter model for speech production. The source models the lung and emits a random excitation signal which is filtered, first by a pitch filter model of the glottal cords and then by a model of the vocal tract.

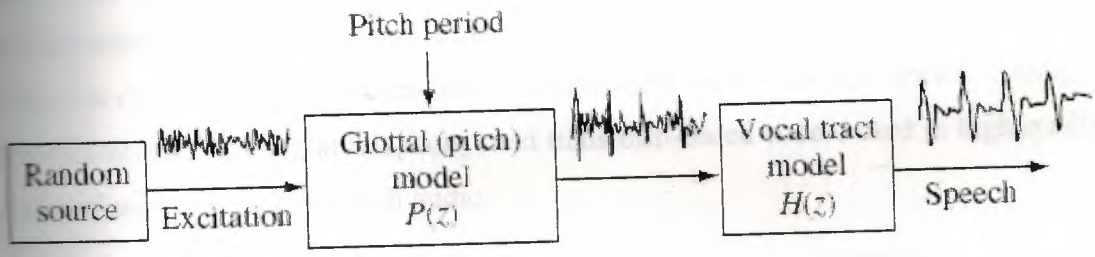


Figure 1.9 Linear predictive model of speech.

The main source of correlation in speech is the vocal tract modelled by a linear predictor. A linear predictor forecasts the amplitude of the signal at time m, x(m), using a linear combination of P previous samples [x(m−1),…,x(m−P)] as

$$\hat{x}(m) = \sum_{k=1}^{P} a_k x(m-k) \tag{1.4}$$

where x'(m) is the prediction of the signal x(m), and the vector $\boldsymbol{a}^T = [a_1,\ldots,a_p]$ is the coefficients vector of a predictor of order P. The prediction error e(m), i.e. the difference between the actual sample, x(m), and its predicted value, $\hat{x}(m)$, is defined as

$$e(m) = x(m) - \sum_{k=1}^{P} a_k x(m-k) \tag{1.5}$$

The prediction error e(m) may also be interpreted as the random excitation or the so-called innovation content of x(m). From Equation (1.5) a signal generated by a linear predictor can be synthesised as

$$x(m) = \sum_{k=1}^{P} a_k x(m-k) + e(m) \tag{1.6}$$

1.4.6 Digital Coding of Audio Signals

In digital audio, the memory required to record a signal, the bandwidth required for signal transmission and the signal-to-quantisation noise ratio are all directly proportional to the number of bits per sample. The objective in the design of a coder is to achieve high fidelity with as few bits per sample as possible, at an affordable implementation cost. Audio signal coding schemes utilise the statistical structures of the signal and a model of the signal generation, together with information on the psychoacoustics and the masking effects of hearing. In general, there are two main categories of audio coders: model-based coders, used for low-bit-rate speech coding in applications such as cellular telephony, and transform-based coders used in high-quality coding of speech and digital hi-fi audio.

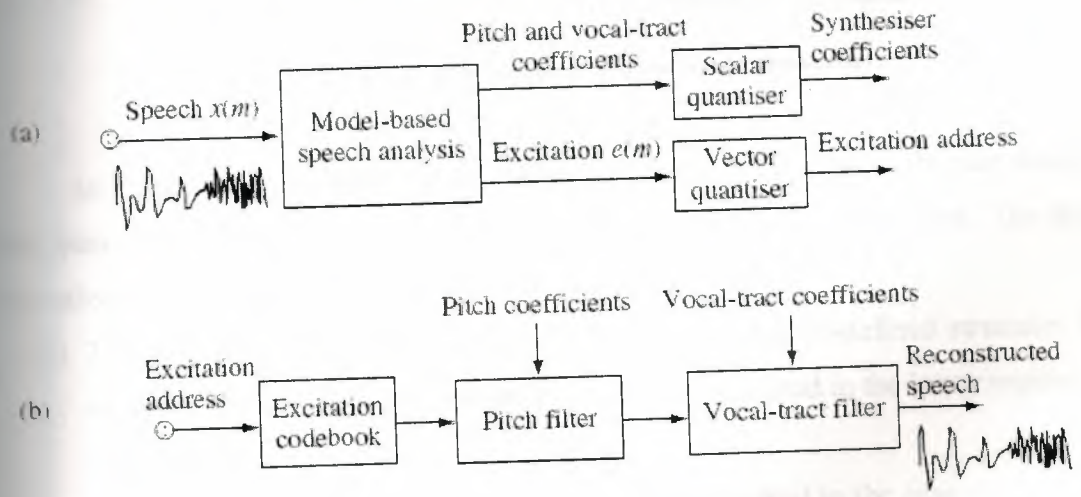


Figure 1.10 Block diagram configuration of a model-based speech (a) coder and (b) decoder.

Figure 1.10 shows a simplified block diagram configuration of a speech coder-decoder of the type used in digital cellular telephones. The speech signal is modelled as the output of a filter excited by a random signal. The random excitation models the air exhaled through the lung, and the filter models the vibrations of the glottal cords and the vocal tract. At the transmitter, speech is segmented into blocks about 30 ms long, during which speech parameters can be assumed to be stationary. Each block of speech samples is analysed to extract and transmit a set of excitation and filter parameters that can be used to synthesise the speech. At the receiver, the model parameters and the excitation are used to reconstruct the speech.

A transform-based coder is shown in Figure 1.11. The aim of transformation is to convert the signal into a form that lends itself to more convenient and useful interpretation and manipulation.

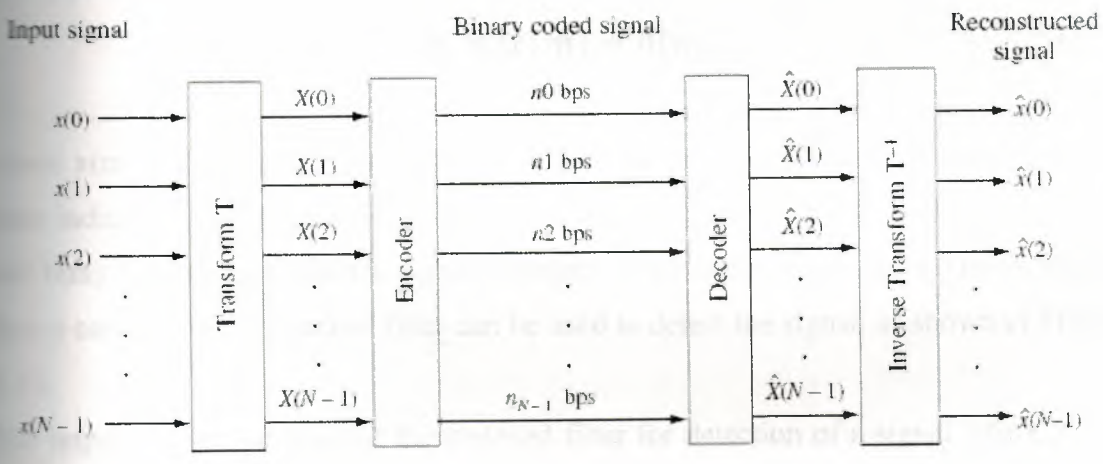


Figure 1.11 Illustration of a transform-based coder.

In Figure 1.11 the input signal is transformed to the frequency domain using a filter bank, or a discrete Fourier transform, or a discrete cosine transform. The three main advantages of coding a signal in the frequency domain are:

(1) The frequency spectrum of a signal has a relatively well-defined structure, for example most of the signal power is usually concentrated in the lower regions of the spectrum.

(2) A relatively low-amplitude frequency would be masked in the near vicinity of a large-amplitude frequency and can therefore be coarsely encoded without any audible degradation.

(3) The frequency samples are orthogonal and can be coded independently with different precisions.

The number of bits assigned to each frequency of a signal is a variable that reflects the contribution of that frequency to the reproduction of a perceptually high-quality signal. In an adaptive coder, the allocation of bits to different frequencies is made to vary with the time variations of the power spectrum of the signal.

1.4.7 Detection of Signals in Noise

In the detection of signals in noise, the aim is to determine if the observation consists of noise alone, or if it contains a signal. The noisy observation, y(m), can be modelled as

$$y(m) = b(m)x(m) + n(m) \tag{1.7}$$

where x(m) is the signal to be detected, n(m) is the noise and b(m) is a binary-valued state indicator sequence such that b(m) = 1 indicates the presence of the signal, x(m), and b(m) = 0 indicates that the signal is absent. If the signal, x(m), has a known shape, then a correlator or a matched filter can be used to detect the signal, as shown in Figure 1.12.

The impulse response h(m) of the matched filter for detection of a signal, x(m), is the time-reversed version of x(m) given by

$$h(m) = x(N - 1 - m) \tag{1.8}$$

where N is the length of x(m). The output of the matched filter is given by

$$z(m) = \sum_{m=0}^{N-1} h(m - k)y(m) \tag{1.9}$$

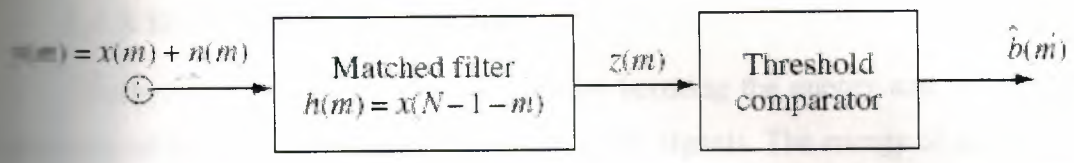


Figure 1.12 Configuration of a matched filter followed by a threshold comparator for detection of signals in noise.

$\hat{b}(m)$	$b(m)$	Detector decision
0	0	Signal absent *Correct*
0	1	Signal absent (*Missed*)
1	0	Signal present (*False alarm*)
1	1	Signal present *Correct*

Table 1.1 Four possible outcomes in a signal detection problem.

The matched filter output is compared with a threshold and a binary decision is made as

$$b(m) = \begin{cases} 1, & if\ z(m) \geq Threshold \\ 0, & otherwise \end{cases} \tag{1.10}$$

where $\hat{b}$(m) is an estimate of the binary state indicator sequence b(m), and may be erroneous, particularly if the signal-to-noise ratio is low. Table 1.1 lists four possible outcomes that, together, b(m) and its estimate, $\hat{b}$(m), can assume. The choice of the threshold level affects the sensitivity of the detector. The higher the threshold, the lower the likelihood that noise would be classified as signal is, so the false alarm rate falls, but the probability of misclassification of signal as noise increases. The risk in choosing a threshold value θ can be expressed as

$$\mathcal{R}(Threshold = \theta) = P_{False\ Alarm}(\theta) + P_{Miss}(\theta) \tag{1.11}$$

The choice of the threshold reflects a trade-off between the misclassification rate $P_{Miss}(\theta)$ and the false alarm rate $P_{False\ Alarm}(\theta)$.

1.4.8 Dolby Noise Reduction

Dolby noise-reduction systems work by boosting the energy and the signal-to-noise ratio of the high-frequency spectrum of audio signals. The energy of audio signals is mostly concentrated in the low-frequency part of the spectrum (below 2 kHz). The higher frequencies that convey quality and sensation have relatively low energy, and can be degraded by even a small amount of noise. For example, when a signal is recorded on a magnetic tape, the tape 'hiss' noise affects the quality of the recorded signal. On playback, the higher-frequency parts of an audio signal recorded on a tape have a smaller signal-to-noise ratio than the low frequency parts. Therefore noise at high frequencies is more audible and less masked by the signal energy. Dolby noise reduction systems broadly work on the principle of emphasizing and boosting the low energy of the high-frequency signal components prior to recording the signal. When a signal is recorded, it is processed and encoded using a combination of a pre-emphasis filter and dynamic range compression. At playback, the signal is recovered using a decoder based on a combination of a de-emphasis filter and a decompression circuit. The encoder and decoder must be well matched and cancel each other out in order to avoid processing distortion.

Dolby developed a number of noise-reduction systems designated Dolby A, Dolby B and Dolby C. These differ mainly in the number of bands and the pre-emphasis strategy that that they employ. Dolby A, developed for professional use, divides the signal spectrum into four frequency bands: band 1 is low-pass and covers 0 to 80 Hz; band 2 is band-pass and covers 80 Hz to 3 kHz; band 3 is high-pass and covers above 3 kHz; and band 4 is also high-pass and covers above 9 kHz. At the encoder the gain in each band is adaptively adjusted to boost low-energy signal components. Dolby A provides a maximum gain of 10–15 dB in each band if the signal level falls 45 dB below the maximum recording level. The Dolby B and Dolby C systems are designed for consumer audio systems, and use two bands instead of the four bands used in Dolby A. Dolby B provides a boost of up to 10 dB when the signal level is low (less than 45 dB below the maximum reference) and Dolby C provides a boost of up to 20 dB, as illustrated in Figure 1.15.

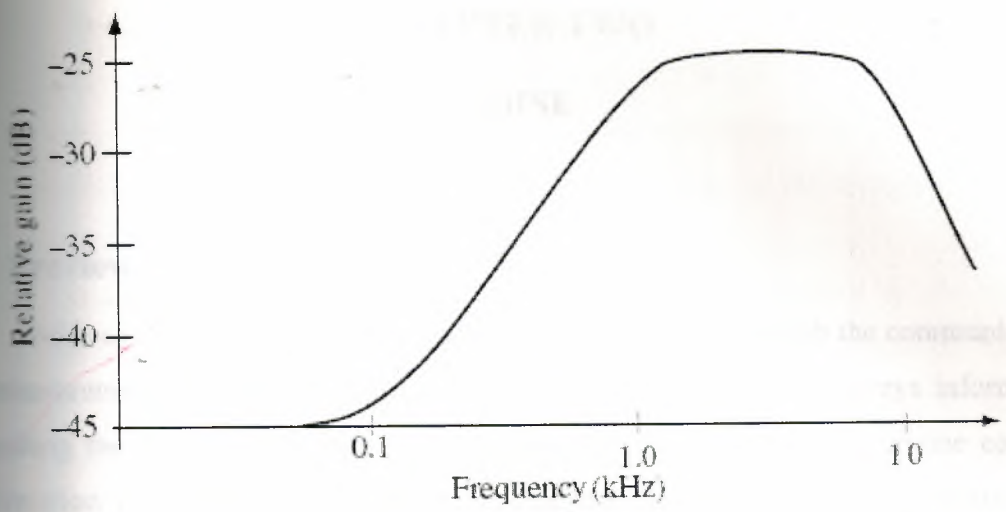


Figure 1.15 Illustration of the pre-emphasis response of Dolby C: up to 20 dB boost is provided when the signal falls 45 dB below maximum recording level.

CHAPTER TWO

NOISE

2.1 Preview

Noise can be defined as an unwanted signal that interferes with the communication or measurement of another signal. A noise itself is a signal that conveys information regarding the source of the noise. For example, the noise from a car engine conveys information regarding the state of the engine and how smoothly it is running. The sources of noise are many and varied and include thermal noise intrinsic to electric conductors, shot noise inherent in electric current flows, audio-frequency acoustic noise emanating from moving, vibrating or colliding sources such as revolving machines, moving vehicles, computer fans, keyboard clicks, wind, rain, etc. and radio-frequency electromagnetic noise that can interfere with the transmission and reception of voice, image and data over the radio-frequency spectrum. Signal distortion is the term often used to describe a systematic undesirable change in a signal and refers to changes in a signal due to the nonideal characteristics of the communication channel, reverberations, echo, multipath reflections and missing samples.

Noise is present in various degrees in almost all environments. For example, in a digital cellular mobile telephone system, there may be several varieties of noise that could degrade the quality of communication, such as acoustic background noise, thermal noise, shot noise, electromagnetic radio-frequency noise, co-channel radio interference, radio-channel distortion, acoustic and line echoes, multipath reflection, fading and signal processing noise. Noise can cause transmission errors and may even disrupt a communication process; hence noise processing is an important and integral part of modern telecommunications and signal processing systems. The success of a noise processing method depends on its ability to characterize and model the noise process, and to use the noise characteristics advantageously to differentiate the signal from the noise.

Depending on its source, a noise can be classified into a number of categories, indicating the broad physical nature of the noise, as follows:

(1) Acoustic noise – emanates from moving, vibrating or colliding sources and is the most familiar type of noise present to various degrees in everyday environments. Acoustic noise is generated by such sources as moving cars, air-conditioners, computer fans, traffic, people talking in the background, wind, rain, etc.

(2) Thermal noise and shot noise – thermal noise is generated by the random movements of thermally energized particles in an electric conductor. Thermal noise is intrinsic to all conductors and is present without any applied voltage. Shot noise consists of random fluctuations of the electric current in an electrical conductor and is intrinsic to current flow. Shot noise is caused by the fact that the current is carried by discrete charges (i.e. electrons) with random fluctuations and random arrival times.

(3) Electromagnetic noise – present at all frequencies and in particular at the radio frequency range (kHz to GHz range) where telecommunications take place. All electric devices, such as radio and television transmitters and receivers, generate electromagnetic noise.

(4) Electrostatic noise – generated by the presence of a voltage with or without current flow. Fluorescent lighting is one of the more common sources of electrostatic noise.

(5) Channel distortions, echo and fading – due to nonideal characteristics of communication channels. Radio channels, such as those at GHz frequencies used by cellular mobile phone operators, are particularly sensitive to the propagation characteristics of the channel environment and fading of signals.

(6) Processing noise – the noise that results from the digital-to-analogue processing of signals, e.g. quantization noise in digital coding of speech or image signals, or lost data packets in digital data communication systems.

Depending on its frequency spectrum or time characteristics, a noise process can be further classified into one of several categories as follows:

(1) White noise – purely random noise that has a flat power spectrum. White noise theoretically contains all frequencies in equal intensity.

(2) Band-limited white noise – a noise with a flat spectrum and a limited bandwidth that usually covers the limited spectrum of the device or the signal of interest.

(3) Narrowband noise – a noise process with a narrow bandwidth such as a 50–60 Hz 'hum' from the electricity supply.

(4) Colored noise – nonwhite noise or any wideband noise whose spectrum has a nonflat shape; examples are pink noise, brown noise and autoregressive noise.

(5) Impulsive noise – consists of short-duration pulses of random amplitude and random duration.

(6) Transient noise pulses – consists of relatively long duration noise pulses.

2.2 White Noise

White noise is defined as an uncorrelated random noise process with equal power at all frequencies (Figure 2.1). A random noise that has the same power at all frequencies in the range of $\pm\infty$ would necessarily need to have infinite power, and is therefore only a theoretical concept. However a band-limited noise process, with a flat spectrum covering the frequency range of a band-limited communication system, is to all intents and purposes from the point of view of the system a white noise process. For example, for an audio system with a bandwidth of 10 kHz, any flat-spectrum audio noise with a bandwidth of equal to or greater than 10 kHz looks like white noise. The autocorrelation function of a continuous-time zero-mean white noise process with a variance of σ^2 is a delta function [Figure 2.1(b)] given by

$$r_{NN}(\tau) = E[N(t)N(t+\tau)] = \sigma^2\delta(\tau) \tag{2.1}$$

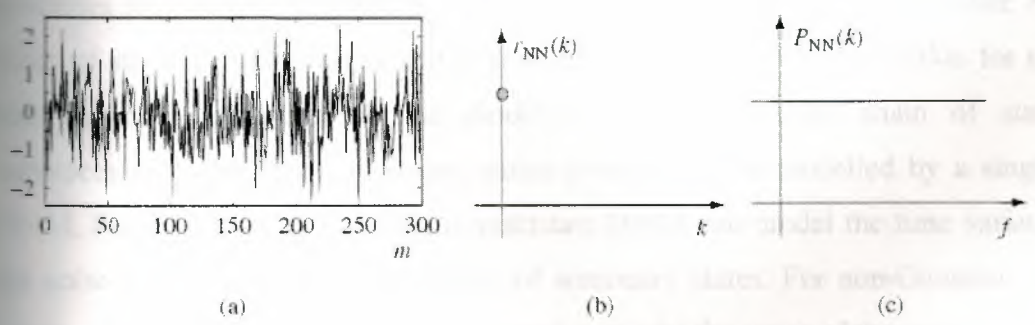


Figure 2.1 (a) Illustration of white noise. (b) Its autocorrelation function is a delta function. (c) Its power spectrum is constant.

The power spectrum of a white noise, obtained by taking the Fourier transform of Equation (2.1), is given by

$$P_{NN}(f) = \int_{-\infty}^{\infty} r_{NN}(t)e^{-j2\pi ft}dt = \sigma^2 \tag{2.2}$$

Equation (2.2) and Figure 2.1(c) show that a white noise has a constant power spectrum.

2.2.1 Additive White Gaussian Noise Model

In classical communication theory, it is often assumed that the noise is a stationary additive white Gaussian (AWGN) process. Although for some problems this is a valid assumption and leads to mathematically convenient and useful solutions, in practice the noise is often time-varying, correlated and non-Gaussian. This is particularly true for impulsive-type noise and for acoustic noise, which are nonstationary and non-Gaussian and hence cannot be modelled using the AWGN assumption. Nonstationary and non-Gaussian noise processes can be modelled by a Markovian chain of stationary subprocesses.

2.2.2 Hidden Markov Model (HMM) for Noise

Most noise processes are nonstationary; that is the statistical parameters of the noise, such as its mean, variance and power spectrum, vary with time. Nonstationary processes may be modelled using HMM. An HMM is essentially a finite-state Markov chain of stationary subprocesses. The implicit assumption in using HMMs for noise is that the noise statistics can be modelled by a Markovian chain of stationary subprocesses. Note that a stationary noise process can be modelled by a single-state HMM. For a nonstationary noise, a multistate HMM can model the time variations of the noise process with a finite number of stationary states. For non-Gaussian noise, a mixture Gaussian density model can be used to model the space of the noise within each state. In general, the number of states per model and number of mixtures per state required to accurately model a noise process depends on the nonstationary character of

noise. An example of a nonstationary noise is the impulsive noise of Figure 2.2(a). Figure 2.2(b) shows a two-state HMM of the impulsive noise sequence: the state S_0 models the 'impulseoff' periods between the impulses, and state S_1 models an impulse. In these cases where

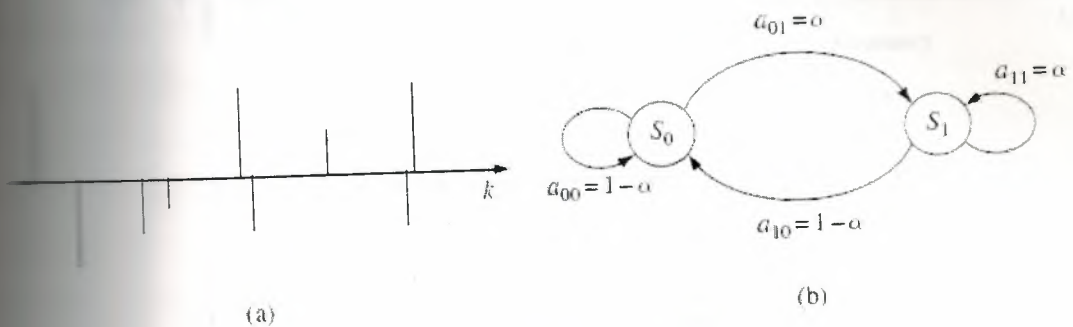


Figure 2.2 (a) An impulsive noise sequence. (b) A binary-state model of impulsive noise.

2.3 Coloured Noise

Although the concept of white noise provides a reasonably realistic and mathematically convenient and useful approximation to some predominant noise processes encountered in telecommunications systems, many other noise processes are nonwhite. The term 'colored noise' refers to any broadband noise with a nonwhite spectrum. For example most audio-frequency noise, such as the noise from moving cars, noise from computer fans, electric drill noise and people talking in the background, has a nonwhite predominantly low-frequency spectrum. Also, a white noise passing through a channel is 'colored' by the shape of the frequency response of the channel. Two classic varieties of colored noise are so-called 'pink noise' and 'brown noise', shown in Figures 2.3 and 2.4.

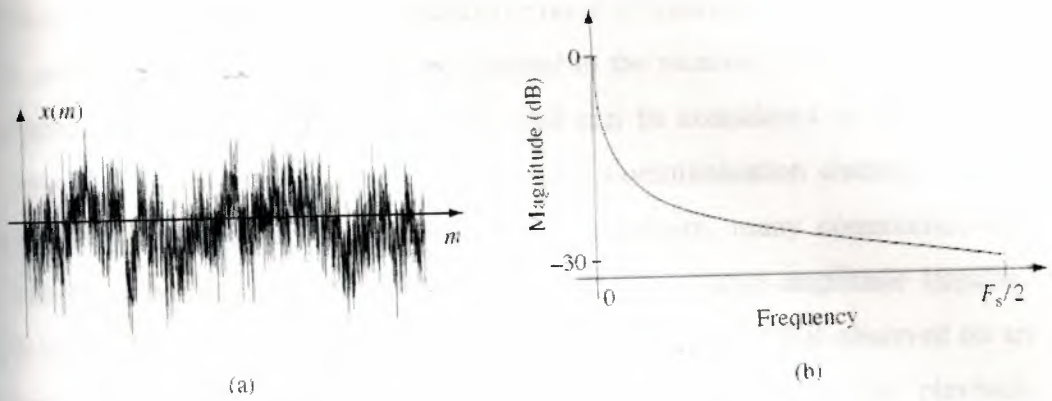


Figure 2.3 (a) A pink noise signal and (b) its magnitude spectrum.

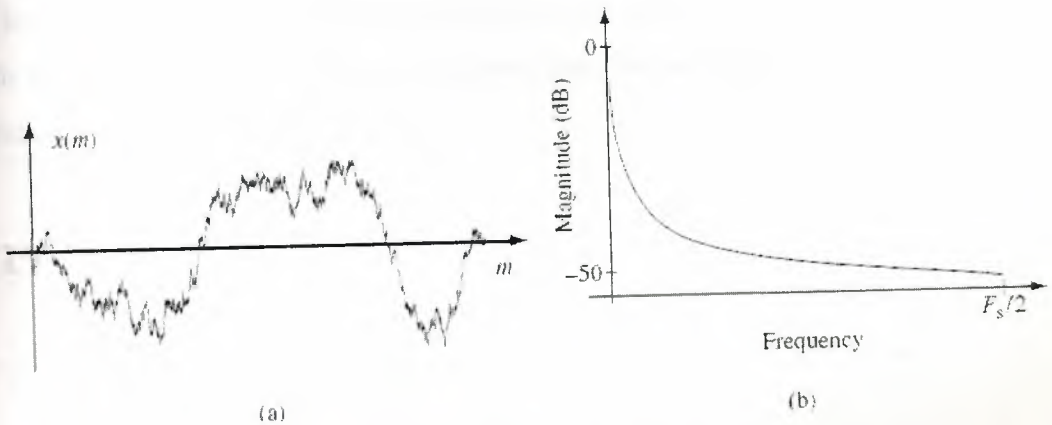


Figure 2.4 (a) A brown noise signal and (b) its magnitude spectrum.

2.4 Impulsive Noise

Impulsive noise consists of random short-duration 'on/off' noise pulses, caused by a variety of sources, such as switching noise, electromagnetic interference, adverse channel environment in a communication system, drop-outs or surface degradation of audio recordings, clicks from computer keyboards, etc.

Figure 2.5(a) shows an ideal impulse and its frequency spectrum. In communication systems, a real impulsive-type noise has a duration that is normally more than one sample long. For example, in the context of audio signals, short-duration, sharp pulses, of up to 3 ms (60 samples at a 20 kHz sampling rate) may be considered as impulsive noise. Figure 2.5(b) and (c) illustrates two examples of short-duration pulses and their respective spectra.

In a communications system, an impulsive noise originates at some point in time and space, and then propagates through the channel to the receiver. The received noise is time dispersed and shaped by the channel, and can be considered as the channel impulse response. In general, the characteristics of a communication channel may be linear or nonlinear, stationary or time-varying. Furthermore, many communications systems exhibit a nonlinear characteristic in response to a large-amplitude impulse. Figure 2.6 illustrates some examples of impulsive noise, typical of that observed on an old gramophone recording. In this case, the communication channel is the playback system, and may be assumed to be time-invariant. The figure also shows some variations of the channel characteristics with the amplitude of impulsive noise. For example, in Figure 2.6(c) a large impulse excitation has generated a decaying transient pulse with time-varying period. These variations may be attributed to the nonlinear characteristics of the playback mechanism.

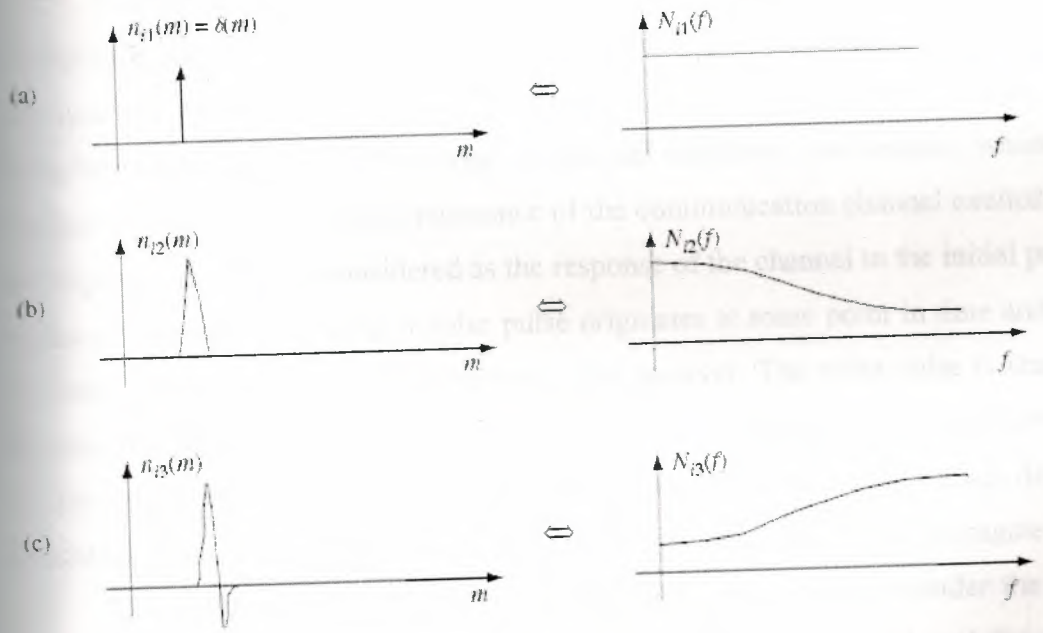


Figure 2.5 Time and frequency sketches of: (a) an ideal impulse; (b) and (c) short-duration pulses.

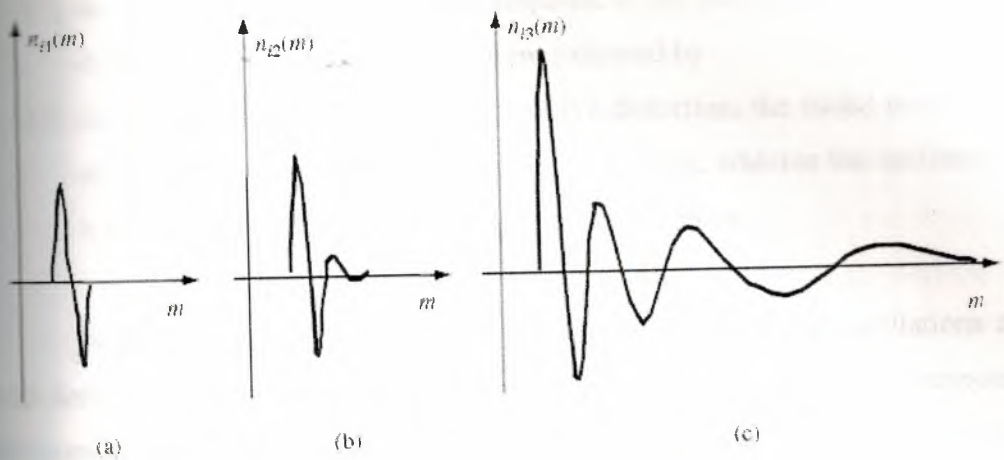


Figure 2.6 Illustration of variations of the impulse response of a nonlinear system with increasing amplitude of the impulse.

2.5 Transient Noise Pulses

Transient noise pulses, observed in most communications systems, are caused by interference. Transient noise pulses often consist of a relatively short, sharp initial pulse followed by decaying low-frequency oscillations, as shown in Figure 2.7. The initial pulse is usually due to some external or internal impulsive interference, whereas the oscillations are often due to the resonance of the communication channel excited by the initial pulse, and may be considered as the response of the channel to the initial pulse. In a telecommunications system, a noise pulse originates at some point in time and space, and then propagates through the channel to the receiver. The noise pulse is shaped by the channel characteristics, and may be considered as the channel pulse response. Thus, we should be able to characterize the transient noise pulses with a similar degree of consistency as in characterizing the channels through which the pulses propagate.

As an illustration of the shape of a transient noise pulse, consider the scratch pulses from a damaged gramophone record shown in Figure 2.7(a) and (b). Scratch noise pulses are acoustic manifestations of the response of the stylus and the associated electromechanical playback system to a sharp physical discontinuity on the recording medium. Since scratches are essentially the impulse response of the playback mechanism, it is expected that, for a given system, various scratch pulses exhibit similar characteristics. As shown in Figure 2.7(b), a typical scratch pulse waveform often exhibits two distinct regions:

(1) the initial high-amplitude pulse response of the playback system to the physical discontinuity on the record medium; followed by

(2) decaying oscillations that cause additive distortion; the initial pulse is relatively short and has a duration on the order of 1–5 ms, whereas the oscillatory tail has a longer duration and may last up to 50 ms or more.

Note in Figure 2.7(b) that the frequency of the decaying oscillations decreases with time. This behaviour may be attributed to the nonlinear modes of response of the electromechanical playback system excited by the physical scratch discontinuity. Observations of many scratch waveforms from damaged gramophone records reveals that they have a well-defined profile, and can be characterised by a relatively small number of typical templates.

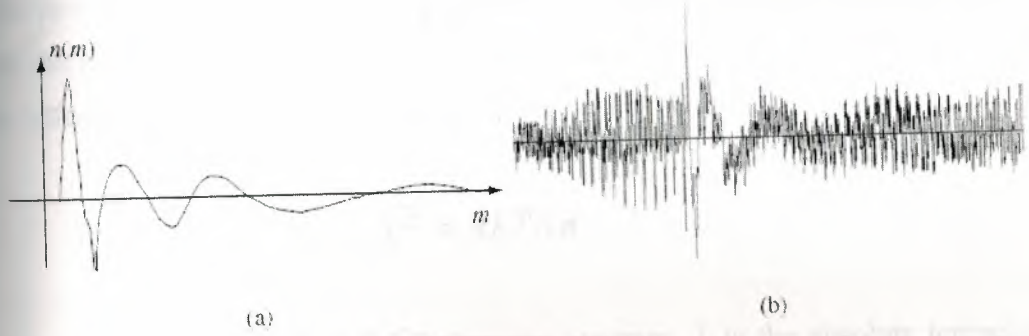


Figure 2.7 (a) A scratch pulse and music from a gramophone record. (b) The averaged profile of a gramophone record scratch pulse.

2.6 Thermal Noise

Thermal noise, also referred to as Johnson noise (after its discoverer, J.B. Johnson), is generated by the random movements of thermally energised (agitated) particles inside an electric conductor. Thermal noise is intrinsic to all resistors and is not a sign of poor design or manufacture, although some resistors may also have excess noise. Thermal noise cannot be circumvented by good shielding or grounding.

Note that thermal noise happens at equilibrium without the application of a voltage. The application of a voltage and the movement of current in a conductor cause an additional random fluctuation known as shot noise, as described in the next section.

The concept of thermal noise has its roots in thermodynamics and is associated with the temperature-dependent random movements of free particles such as gas molecules in a container or electrons in a conductor. Although these random particle movements average to zero, the fluctuations about the average constitute the thermal noise. For example, the random movements and collisions of gas molecules in a confined space produce random fluctuations about the average pressure. As the temperature increases, the kinetic energy of the molecules and the thermal noise increase.

Similarly, an electrical conductor contains a very large number of free electrons, together with ions that vibrate randomly about their equilibrium positions and resist the movement of the electrons. The free movement of electrons constitutes random spontaneous currents, or thermal noise, that average to zero since, in the absent of a voltage, electrons move in different directions. As the temperature of a conductor, from heat provided by its surroundings, increases, the electrons move to higher-energy states and the random current flow increases. For a metallic resistor, the mean square value of the instantaneous voltage due to the thermal noise is given by

$$\overline{v^2} = 4kTRB \tag{2.3}$$

where k = 138×10^{-23} J/k is the Boltzmann constant, T is the absolute temperature in degrees Kelvin, R is the resistance in ohms and B is the bandwidth. From Equation (2.3) and the preceding argument, a metallic resistor sitting on a table can be considered as a generator of thermal noise power, with a mean square voltage $\overline{v^2}$ and an internal resistance R. From circuit theory, the maximum available power delivered by a 'thermal noise generator', dissipated in a matched load of resistance R, is given by

$$P_N = \overline{i^2}R = \left(\frac{v_{rms}}{2R}\right)^2$$

$$R = \frac{\overline{v^2}}{4R} = kTB \text{ (W)} \tag{2.4}$$

where v_{rms} is the root mean square voltage. The spectral density of thermal noise is given by

$$P_N(f) = \frac{kT}{2} (\mathrm{W/Hz}) \tag{2.5}$$

From Equation (2.5), the thermal noise spectral density has a flat shape, i.e. thermal noise is a white noise. Equation (2.5) holds well up to very high radio-frequencies of 1013 Hz.

2.7 Electromagnetic Noise

Electromagnetic waves present in the environment constitute a level of background noise that can interfere with the operation of communication and signal processing systems. Electromagnetic waves may emanate from man-made devices or natural sources. The primary natural source of electromagnetic waves is the Sun. In the order of decreasing wavelength and increasing frequency, various types of electromagnetic radiation include: electric motors (kHz), radio waves (kHz to GHz), microwaves (10^{11} Hz), infrared radiation (10^{13} Hz), visible light (10^{14} Hz), ultraviolet radiation (10^{15} Hz), X-rays (10^{20} Hz) and γ-radiation (10^{23} Hz).

Virtually every electrical device that generates, consumes or transmits power is a source of pollution of radio spectrum and a potential source of electromagnetic noise interference for other systems. In general, the higher the voltage or the current level, and the closer the proximity of electrical circuits/devices, the greater will be the induced noise. The common sources of electromagnetic noise are transformers, radio and television transmitters, mobile phones, microwave transmitters, a.c. power lines, motors and motor starters, generators, relays, oscillators, fluorescent lamps and electrical storms.

Electrical noise from these sources can be categorized into two basic types: electrostatic and magnetic. These two types of noise are fundamentally different, and thus require different noise-shielding measures. Unfortunately, most of the common noise sources listed above produce combinations of the two noise types, which can complicate the noise reduction problem.

Electrostatic fields are generated by the presence of voltage, with or without current flow. Fluorescent lighting is one of the more common sources of electrostatic noise. Magnetic fields are created either by the flow of electric current or by the presence of permanent magnetism. Motors and transformers are examples of the former,

and the Earth's magnetic field is an instance of the latter. In order for noise voltage to be developed in a conductor, magnetic lines of flux must be cut by the conductor. Electric (and noise) generators function on this basic principle. In the presence of an alternating field, such as that surrounding a 50–60 Hz power line, voltage will be induced into any stationary conductor as the magnetic field expands and collapses. Similarly, a conductor moving through the Earth's magnetic field has a noise voltage generated in it as it cuts the lines of flux.

The main sources of electromagnetic interference in mobile communications systems are the radiations from the antennas of other mobile phones and base stations. The electromagnetic interference by mobile users and base stations can be reduced by the use of narrow-beam adaptive antennas, the so-called 'smart antennas'.

2.8 Channel Distortions

On propagating through a channel, signals are shaped, delayed and distorted by the frequency response and the attenuating (fading) characteristics of the channel. There are two main manifestations of channel distortions: magnitude distortion and phase distortion. In addition, in radio communication, we have the multipath effect, in which the transmitted signal may take several different routes to the receiver, with the effect that multiple versions of the signal with different delay and attenuation arrive at the receiver. Channel distortions can degrade or even severely disrupt a communication process, and hence channel modelling and equalization are essential components of modern digital communications systems. Channel equalization is particularly important in modern cellular communications systems, since the variations of channel characteristics and propagation attenuation in cellular radio systems are far greater than those of the landline systems.

Figure 2.8 illustrates the frequency response of a channel with one invertible and two noninvertible regions. In the noninvertible regions, the signal frequencies are heavily attenuated and lost to the channel noise. In the invertible region, the signal is distorted but recoverable.

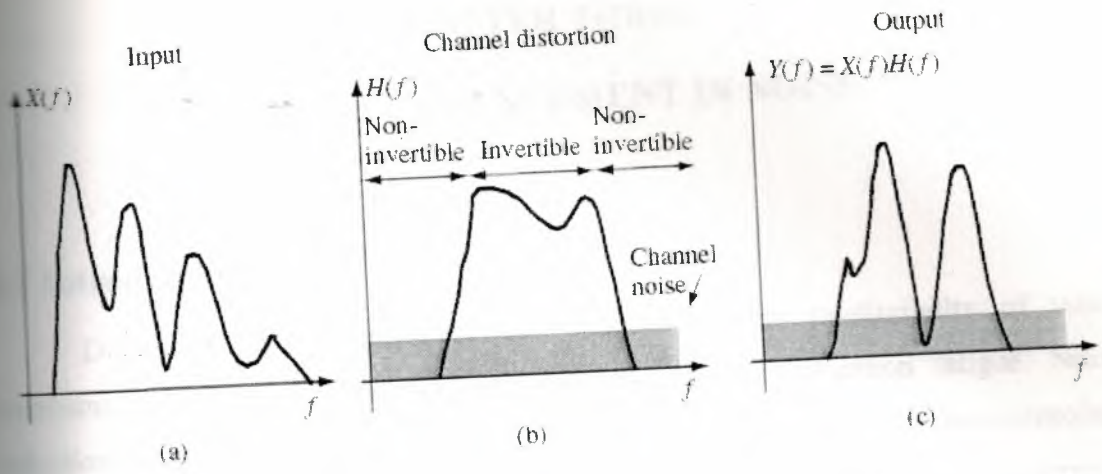


Figure 2.8 Illustration of channel distortion: (a) the input signal spectrum; (b) the channel frequency response; (c) the channel output.

This example illustrates that the channel inverse filter must be implemented with care in order to avoid undesirable results such as noise amplification at frequencies with a low signal-to-noise ratio.

CHAPTER THREE

SPEECH ENHANCEMENT IN NOISE

3.1 Introduction

De-noising speech improves the quality and the intelligibility of voice communication in noisy environments and reduces communication fatigue. Noise reduction benefits the users of hands-free phones, mobile phones and voice-controlled automated services used in noisy moving environments such as cars, trains, streets, conference halls and other public venues. Figure 3.1 illustrates a classification of the main signal processing methods for enhancement of noisy speech into two broad types:

(1) Single-input speech enhancement systems, where the only available signal is the noise contaminated speech picked up by a single microphone. Single input systems do not *cancel* noise, rather they *suppress* the noise using estimates of the signal-to-noise ratio of the frequency spectrum of the input signal. Single-input systems rely on the statistical models of speech and noise, which may be estimated from the speech-inactive periods or decoded from a set of pre-trained models of speech and noise. An example of a useful application of a single-input enhancement system is a mobile phone system used in noisy environments.

(2) Multiple-input speech enhancement systems, where a number of signals containing speech and noise are picked up by several microphones. Examples of multiple inputs systems are adaptive noise cancellation, adaptive beam-forming microphone arrays and multiple-input multiple-output (MIMO) acoustic echo cancellation systems. In multiple-input systems the microphones can be designed, spatially arranged and adapted for optimum performance. Multiple-input noise-reduction systems are useful for teleconferencing systems and for in-car cabin communication systems.

Figure 3.1 illustrates a categorization of the main noise reduction methods used for single-input and multiple-input scenarios. In order to achieve the best noise-reduction

performance, where possible, the advantages of the signal processing methods developed for single-input noise suppression and multiple-inputs noise cancellation systems are combined.

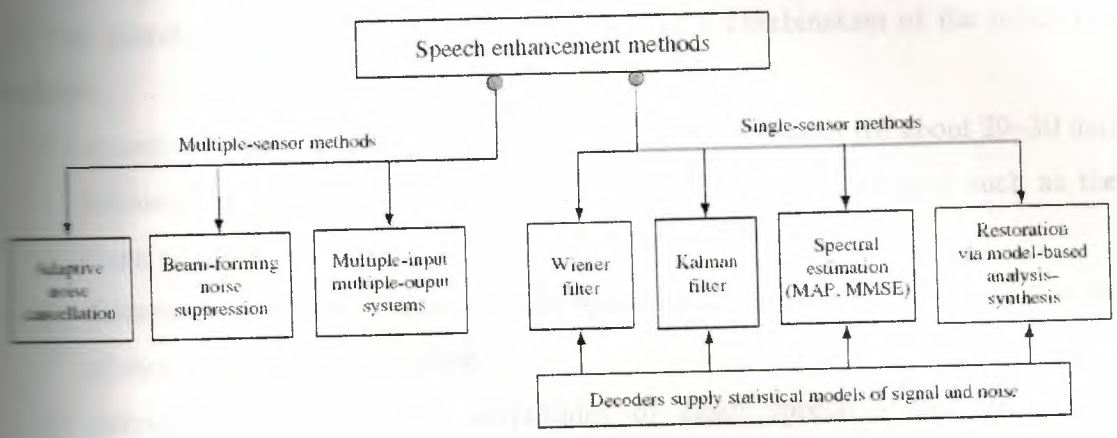


Figure 3.1 A categorization of speech enhancement methods. Note that statistical models can optionally provide single-input noise reduction methods with the additional information needed for improved performance.

3.2 Single-Input Speech-Enhancement Methods

In single-input systems the only available signal is the noisy speech; however, in applications where speech enhancement and recognition are performed on the same system, the results of speech recognition can provide the speech enhancement method with such information as the statistics of the power spectra or correlation matrices obtained from decoding the most likely speech and noise models. Single-input noise-reduction methods include Wiener filter, spectral subtraction, Kalman filter, MMSE method and speech restoration via model-based analysis and synthesis methods, as described in this section.

3.2.1 An Overview of a Speech-Enhancement System

Assuming that the speech signal, x(m), and the noise, n(m), are additive, the noisy speech, y(m), is modelled as

$$y(m) = x(m) + n(m) \tag{3.1}$$

where the integer variable m denotes the discrete-time index. It is generally assumed that the speech is not correlated with noise; this is a reasonable assumption in most cases when the signal and noise are generated by independent sources.

The general form of a typical speech-enhancement method is shown in Figure 3.2. The speech-enhancement system is composed of a combination of the following modules:

(1) speech segmentation into a sequence of overlapping frames (of about 20–30 ms) followed by windowing of each segment with a popular window such as the Hann window;

(2) discrete Fourier transformation of the speech samples within each frame to a set of short-time spectral samples;

(3) estimation of the spectral amplitudes of clean speech – this involves a modification of the magnitude spectrum of noisy speech according to an estimate of the signal to noise ratio at each frequency;

(4) an inter-frame signal smoothing method to utilise the temporal correlations of the spectral values across successive frames of speech;

(5) speech and noise models, and a speech and noise decoder, to supply the speech estimator with the required statistics (power spectra, correlation matrices, etc.) of speech and noise;

(6) voice activity detection, used to estimated and adapt noise models from the noise-only periods and also for applying extra attenuation to noise-only periods.

In the following, the elements of a speech-enhancement system are described in more detail.

3.2.1.1 Segmentation and Windowing of Speech

Speech processing systems divide the sampled speech signal into overlapping frames of about 20–30 ms duration. The N speech samples within each frame are processed and represented by a set of spectral features or by a linear prediction model of speech production.

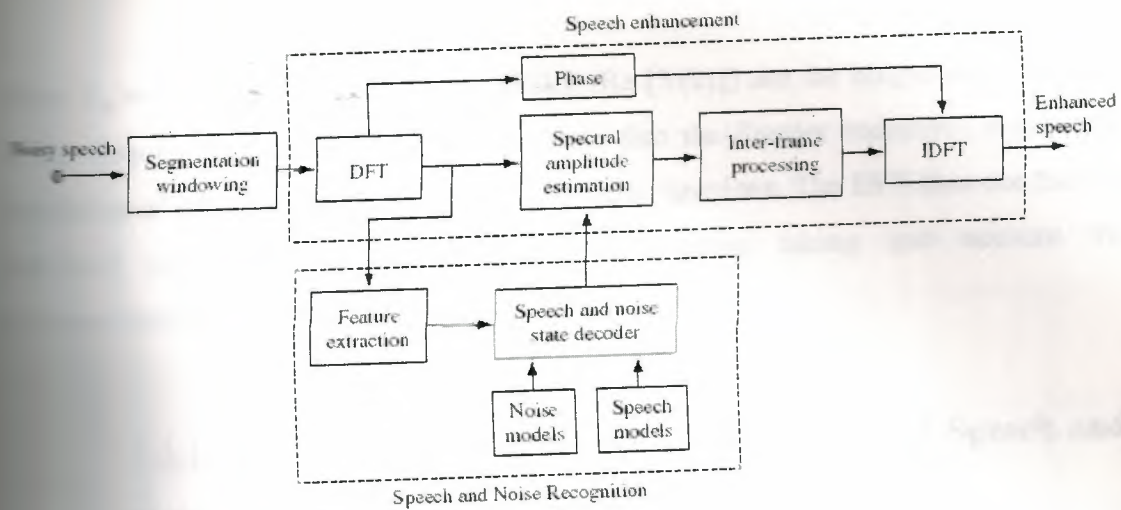


Figure 3.2 Block diagram illustration of a speech-enhancement system.

The signal within each frame is assumed to be a stationary process. The choice of the length of speech frames (typically set to between 20 and 30 ms) is constrained by the stationarity assumption of linear time-invariant signal processing methods such as Fourier transform or linear prediction model, and by the maximum allowable delay for real-time communication systems such as voice coders.

3.2.1.2 Spectral Representation of Speech and Noise

Speech is segmented into overlapping frames of N samples and transformed to frequency domain via discrete Fourier transform. In the frequency domain, the noisy speech can be represented as

$$Y(k) = X(k) + N(k) \quad k = 0, \dots N-1 \tag{3.2}$$

where X(k), N(k) and Y(k) are the short-time discrete Fourier transforms of speech, noise and noisy speech, respectively. The integer k represents the discrete frequency variable; it corresponds to an actual frequency of $2k\pi/N$ (rad/s) or kF_s/N (Hz) where F_s is the sampling frequency.

Equation (3.2) can be written in the complex polar form in terms of the magnitudes and the phases of the signal and noise at discrete frequency k as

$$Y_k e^{j\theta_{Y_k}} = X_k e^{j\theta_{X_k}} + N_k e^{j\theta_{N_k}} \quad k = 0, \dots N-1 \tag{3.3}$$

where $Y_k = |Y(k)|$ and $\theta_{yk} = \tan^{-1}\{\text{Im}[Y(k)]/\text{Re}[Y(k)]\}$ are the magnitude and phase of the frequency spectrum, respectively. Note that the Fourier transform models the correlation of speech samples with sinusoidal basis functions. The DFT bins can then be processed individually or in groups of frequencies, taking into account the psychoacoustics of hearing.

3.2.1.3 Linear Prediction Model Representation of Speech and Noise

The correlation of speech (or noise) samples can be modelled with a linear prediction (aka autoregressive) model. Using a linear prediction model of speech and noise, the noisy speech is expressed as

$$y(m) = \underbrace{\sum_{k=1}^{P} a_k x(m-k)}_{\text{Speech model}} + \underbrace{\sum_{k=1}^{q} b_k n(m-k) + v(m)}_{\text{Noise model}} \qquad (3.4)$$

where a_k and b_k are the coefficients of linear prediction models of speech and noise, respectively. Linear prediction models can be used in a variety of speech enhancement methods, including Wiener filters, Kalman filters and speech restoration via decomposition and re-synthesis.

3.2.1.4 Interframe and Intraframe Correlations

The two main issues in modelling noisy speech are:

(1) modelling and utilization of the probability distributions and the intraframe correlations of speech and noise samples *within* each noisy speech frame;

(2) modelling and utilization of the probability distributions and the interframe correlations of speech and noise features *across* successive frames of noisy speech.

Most speech-enhancement systems are based on estimates of the short-time amplitude spectrum or the linear prediction model of speech. The phase distortion of speech is ignored. In the case of DFT-based features, each spectral sample, X(k), at a discrete frequency k is the correlation of speech samples, x(m), with a sinusoidal basis function $e^{-j2\pi km/N)}$. The intraframe spectral correlation, that is the correlation of spectral samples within a frame of speech, is often ignored, as is the inter-frame temporal correlation of spectral samples across successive speech frames.

In the case of linear prediction models, the poles model the spectral correlations within each frame. However, the denoising of linear prediction model poles, or coefficients, is achieved through denoising the frequency response of clean speech and that ignores the correlation of spectral samples. The optimal utilization of the interframe and intraframe correlations of speech samples is a continuing research issue.

3.2.1.5 Speech Estimation Module

At the heart of a speech-enhancement system is the speech-estimation module. For speech enhancement, usually the spectral amplitude, or a linear prediction model, of speech is estimated and this estimate is subsequently used to reconstruct speech samples. A variety of methods have been proposed for the estimation of clean speech, including the Wiener filter, spectral subtraction, Kalman filters, the minimum mean squared error and the maximum *a posteriori* method. For proper functioning of the speech estimation module, knowledge of the statistics of speech and noise is required and this can be estimated from the noisy speech or it can be obtained from pre-trained models of speech and noise.

3.2.1.6 Probability Models of Speech and Noise

The implementation of a noise-reduction method such as the Wiener filter, Kalman filter, spectral subtraction or a Bayesian estimation method requires estimates of the statistics (and in particular the power spectra or equivalently the correlation matrices) of the speech and noise. An estimate of the noise statistics can be obtained from speech-inactive periods; however, for best results the speech and noise statistics are obtained from a network of probability models of speech and noise, and this

essentially implies that in an optimal speech processing system speech recognition and speech enhancement would need to be integrated. The most commonly used probability models for speech are hidden Markov models. Hidden Markov models, or alternatively Gaussian mixture models, can also be used for modelling nonstationary noise. To model different types of noise, a number of HMMs need to be trained, one HMM for each type of noise. Alternatively, one can use a GMM of noise with a large number of components, with each component effectively modelling a different type of noise.

3.2.2 Wiener Filter for De-Noising Speech

The Wiener filter theory, introduced in Chapter 6, forms the foundation of speech de-noising systems. The output of a Wiener filter is given by

$$\hat{x}(m) = \sum_{i=0}^{P} w(i)y(m-i) \tag{3.5}$$

where w(k) is the filter coefficient for de-noising the input speech y(m) and $\hat{x}$(m) is the estimate of clean speech x(m).

$$w = R_{yy}^{-1} r_{xy} \tag{3.6}$$

where $\boldsymbol{R}yy$ is the autocorrelation matrix of the noisy speech signal, $\boldsymbol{y}$, and $\boldsymbol{r}_{yx}$ is the cross correlation vector of the clean speech, $\boldsymbol{x}$, and noisy speech, $\boldsymbol{y}$.

For uncorrelated speech and noise, the Wiener filter Equation (3.6) can be written as

$$w = [R_{xx} + R_{nn}]^{-1} r_{xx} \tag{3.7}$$

where $\boldsymbol{R}$xx and $\boldsymbol{R}$nn are the autocorrelation matrices of the speech and noise, respectively, and $\boldsymbol{r}_{xx}$ is the autocorrelation vector of the speech. In the frequency domain, for additive noise uncorrelated with speech, the Wiener filter equation

$$W(k) = \frac{P_{XX}(k)}{P_{XX}(k)+P_{NN}(k)} \tag{3.8}$$

here W(k) is the frequency response of the Wiener filter, $P_{XX}(k)$ and $P_{NN}(k)$ are the ower spectra of speech and noise, respectively, and k is the discrete frequency variable. igure 3.3 outlines a block diagram implementation of a frequency domain Wiener ilter.

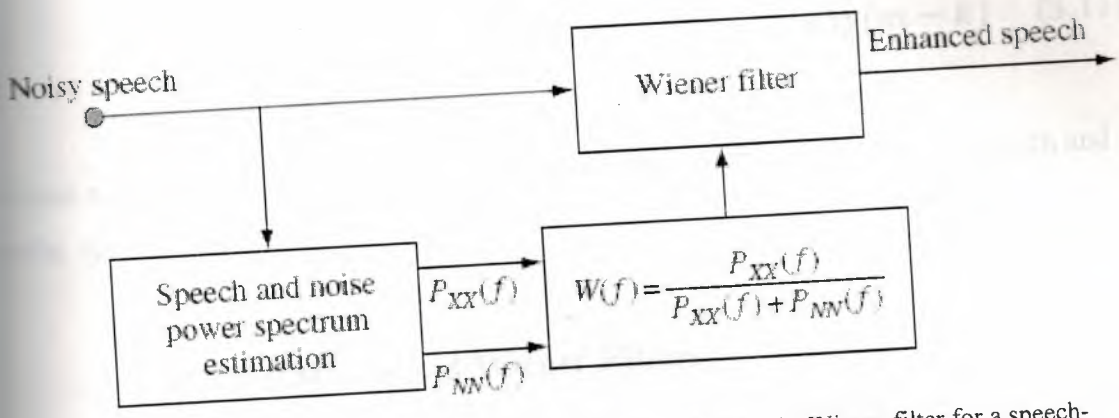


Figure 3.3 Block diagram overview of implementation of a Wiener filter for a speech-enhancement system.

By dividing the numerator and the denominator of Equation (3.8) by $P_{NN}(k)$, the Wiener filter can be expressed in terms of the signal-to-noise ratio as

$$W(k) = \frac{SNR(k)}{SNR(k)+1} \tag{3.9}$$

This equation reveals an important aspect of the general workings of the signal-input noise reduction system: *noise suppression methods effectively use a function of the estimates of the signal-to-noise ratios to modify the spectral amplitudes of the noisy signal.*

3.2.2.1 Wiener Filter Based on Linear Prediction Models

Wiener filters employing linear prediction models of speech and noise may be used for speech enhancement. The frequency response of Wiener filter can be expressed in terms of the ratio of power spectra of autoregressive (i.e. linear prediction) models of speech and noise as

$$W(f) = \frac{P_{XX}(f)}{P_{YY}(f)} = \frac{G_X^2/A_X^2(f)}{G_Y^2/A_X^2(f)} = \frac{G_X^2}{G_Y^2}\frac{A_Y^2(f)}{A_X^2(f)} \qquad (3.10)$$

e $G_x(f)/A_x(f)$ and $G_y(f)/A_y(f)$ are the frequency responses of linear prediction els of speech and noisy speech, respectively. In the time domain a square root ion of Wiener filter equation (3.10) can be implemented as

$$\hat{x}(m) = \sum_{k=1}^{P} a_k x(m-k) + \frac{G_X}{G_Y}\sum_{k=0}^{q} a_y(k)y(m-k) \qquad (3.11)$$

ere $a_x(k)$ and $a_y(k)$ are the coefficients of autoregressive models of clean speech and isy speech, respectively.

3.2.2.2 HMM-based Wiener Filters

The key to the successful implementation of a Wiener filter is the accurate stimation of the power spectra of speech and noise, $P_{XX}(k)$ and $P_{NN}(k)$, for the frequency domain Wiener filter of Equation (3.8) or equivalently the estimation of the correlation matrices of speech and noise, $\boldsymbol{R}_{xx}$ and $\boldsymbol{R}_{nn}$, for the time domain Wiener filter of Equation (3.7). This is not a trivial task as speech and most noise processes are nonstationary.

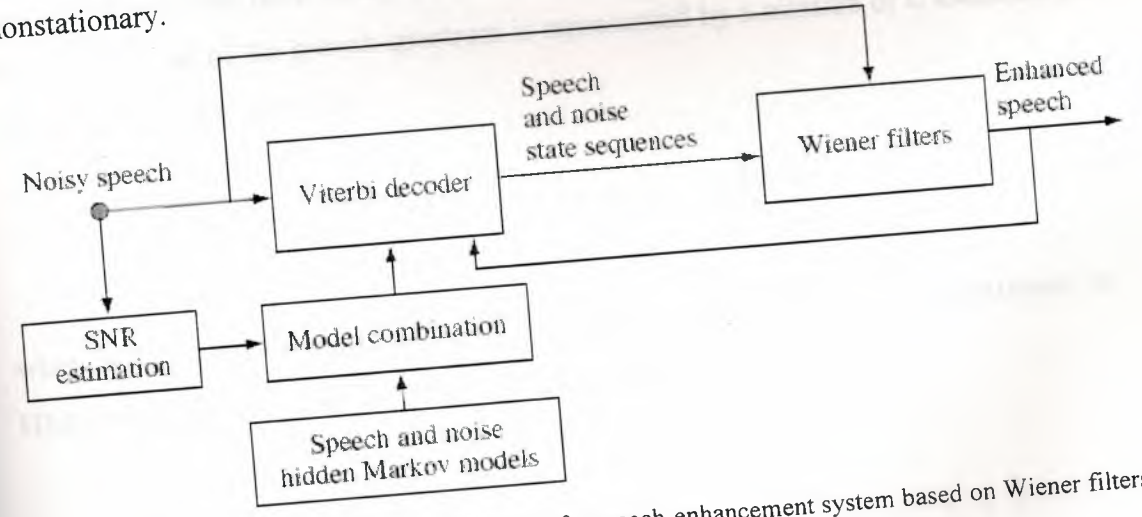


Figure 3.4 Block diagram illustration of a speech-enhancement system based on Wiener filters and hidden Markov models.

Given the noisy speech signal, the time-varying power spectra of speech and noise may be estimated from a set of pre-trained hidden Markov models, or Gaussian

nodels, of speech and noise using a Viterbi decoder (Figure 3.4). An HMM
d Wiener filter involves the following signal processing steps:

peech and noise decomposition – this involves the estimation of the most likely
ombination of speech and noise HMMs given the noisy speech signal. Using
Viterbi state decoders, the most likely combination of speech and noise states
ield the pdfs of the spectra of the most likely estimates of the clean speech and
noise.
The speech and noise power spectra from (1) are used to implement state-based
Wiener filters.

In HMM-based Wiener filtering, the choice of the speech features for training
s needs to be appropriate for both speech recognition and enhancement. Linear
tion-based cepstrum features provide a suitable choice as the cepstrum
cients obtained from HMM states can be mapped to the linear prediction model
cients and thereafter to the linear prediction model spectrum for use in the
mentation of the Wiener filter.

Assuming that for a noisy speech signal spectrum, Y(k), the Viterbi decoder
ns M different most likely state sequences, and that in each state the probability
ity function of the speech spectrum is represented by a mixture of L Gaussian pdfs,
Wiener filter is given by

$$\hat{X}(k) = \left[\sum_{\beta=1}^{M}\sum_{\gamma=1}^{L} p(\beta,\gamma) w_{\beta,\gamma}(k)\right] Y(k) \qquad (3.12)$$

ere $p(\beta,\gamma)$ is the estimated probability of speech and noise spectra from mixture of
MM state β and $W_{\beta,\gamma}(K)$ is the state Wiener filter.

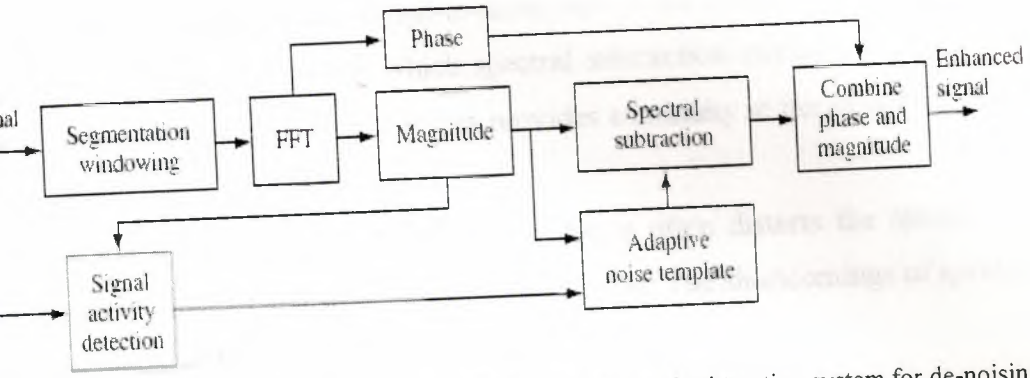


Figure 3.5 Block diagram illustration of an FFT-based spectral subtraction system for de-noising speech.

3.2.3 Spectral Subtraction of Noise

A simple and widely studied speech enhancement method is the spectral ction method (Figure 3.5). In spectral subtraction an estimate of the average itude spectrum of the noise is subtracted from the magnitude spectrum of noisy h. The spectral subtraction filter can be expressed as the product of the noisy h spectrum, Y(k), and a spectral gain function, $W_{SS}(k)$

$$\hat{X}(k) = W_{ss}(k)Y(k) \tag{3.13}$$

re the frequency response of the spectral subtraction filter, $W_{SS}(k)$, is

$$W_{ss}(k) = fn\left[1 - \frac{\alpha(k)\hat{N}(k)}{Y(k)}\right] \tag{3.14}$$

ere $\hat{N}(k)$ is an estimate of the noise average amplitude spectrum, $\alpha(k)$ is a frequency-endent subtraction factor and the function fn(·) is chosen to avoid negative values of ss(k) and provide a smoother frequency response when the signal-to-noise ratio drops relatively lower values. The form of the function fn(·) can be chosen as

$$_{ss}(k) = \begin{cases} 1 - \frac{\alpha(k)\hat{N}(k)}{Y(k)} & \text{if } SNR(k) < SNR_{\text{Thresh}} \\ \gamma \exp\{-\beta[SNR_{\text{Thresh}} - SNR(k)]\} & \text{else} \end{cases} \tag{3.15}$$

R(k) is an estimate of the signal-to-noise ratio at the discrete frequency k and is a threshold SNR below which spectral subtraction switches to a form of al attenuation, γ is a parameter that provides continuity at the switching point n attenuation control factor.

ne problem with spectral subtraction is that it often distorts the speech and the appearance of annoying short bursts of noise. The shortcomings of spectral on method can be summarized as follows:

The only statistics used in spectral subtraction is the mean of the magnitude spectrum of the noise. The mean and variance of the clean speech and the variance of the noise are not employed in the estimation process. Consequently noise variations about the mean are not suppressed and this results in more distortions than would be the case if the variance information were also used.

A hard decision needs to be employed to avoid the values of the estimates of the magnitude spectrum after subtraction going negative or below a noise floor value.

The spectral subtraction method is not speech-specific; the spectral trajectories of speech across time are not modelled and used in the de-noising process.

3.2.4 Speech Enhancement Via Linear Prediction Model Reconstruction

Speech can be enhanced through a process of decomposition, de-noising and reconstruction of the source-filter parameters of a linear prediction model of ch. An LP model of speech, may be expressed as

$$X(z) = E(z)V(z) \tag{3.16}$$

re E(z) is the z-transform of the excitation signal and V(z) is the z-transfer function a combined model of the vocal tract, glottal pulse and lip radiation; V(z) can be pressed by a cascade combination of a set of second-order resonators and a first-order del as

$$V(z,m) = G(m)\frac{1}{1+r_0 m)}\prod_{K=1}^{N}\frac{1}{1-2r_k(m)\cos(\varphi_k(m))z^{-1}+r_k^2(m)z^{-2}} \quad (3.17)$$

e $\Phi k(m)$ and $r_k(m)$ are the time-varying radii and the angular frequencies of the of the LP model, respectively, and G(m) is the overall gain of the LP model. In tion (3.17) the second-order sections model the resonance of speech spectrum and irst-order section models the slope of the spectrum of speech.

Formants are the resonances of speech. The poles of the LP model are the ant candidates, the raw data from which formants and their models are estimated. spectral resonance at a formant is characterized by a parameter vector comprising frequency F_k, bandwidth B_k and magnitude of the resonance M_k and their temporal e of variation (velocity), as

$$F_k = [F_k, B_k, M_k, \Delta F_k, \Delta B_k, \Delta M_k] \quad k = 1, \ldots., M \quad (3.18)$$

ere Δ denotes the slope of the trajectory of a feature vector over time, e.g. $\Delta F_k(t)$ resenting the velocity of the k-th formant at frame t is obtained as

$$\Delta F_k(t) = \frac{\sum_{m=1}^{N} m[F_k(t+m)-F_k(t+m)]}{\sum_{m=1}^{N} 2m^2} \quad k = 1, \ldots., M \quad (3.19)$$

uations (3.17)–(3.19) lead to the concept of formant-tracking LP models for speech storation. Format-synthesisers are well established in speech technology, many being velopments of the Klatt synthesiser.

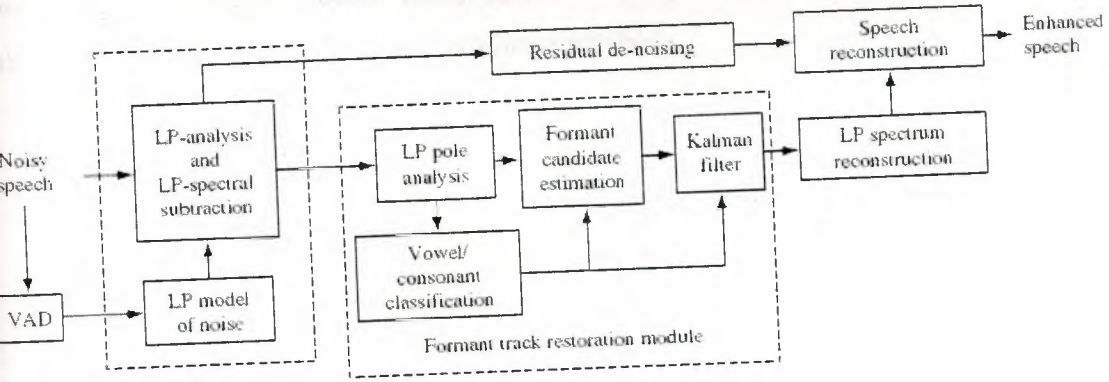


Figure 3.6 Overview of the speech-enhancement system.

3.2.4.1 Formant-tracking Speech Restoration System

The formant-tracking speech restoration system, illustrated in Figure 3.6, consists of the following sections:

(1) an LP model analysis module for estimation of the LP model parameters and the excitation parameters of noisy speech;

(2) a noise detection–estimation module for estimation of the time-varying parameters of noise model;

(3) a speech pre-cleaning module based either on the spectral subtraction or on the MMSE spectral amplitude estimation;

(4) a formant tracking method for tracking the formant parameters of pre-cleaned speech across successive speech frames;

(5) an excitation model for estimating the excitation parameters and providing an estimate of the cleaned excitation;

(6) speech enhancement via synthesis using an estimate of the LP model of speech combined with excitation.

The input to the LP pole analysis is the noisy speech spectrum with the mean of the LP spectrum of noise removed through spectral subtraction. An LP model is fitted to the speech spectrum and the poles of the model are obtained through a polynomial rooting function.

Figure 3.7 shows the reduction in formant tracking error from application of different noise reduction methods. Figure 3.8 shows a comparative illustration of the LP spectrograms of clean speech, noisy speech and speech restored with the spectral subtraction method and the formant synthesis method.

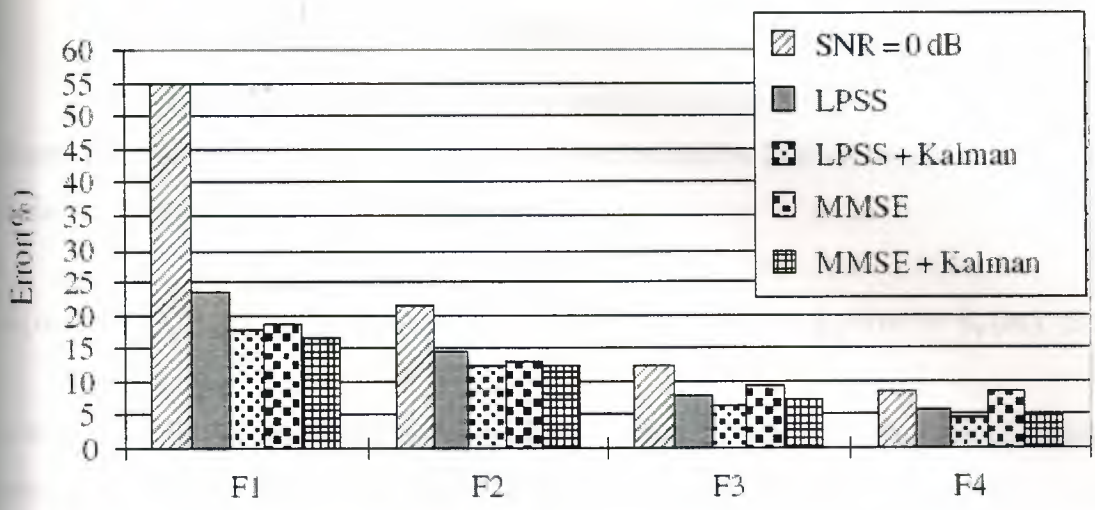


Figure 3.7 Average percentage error of formant tracks of speech in train noise and cleaned speech using LPSS, MMSE and Kalman filters; the results were averaged over five male speakers.

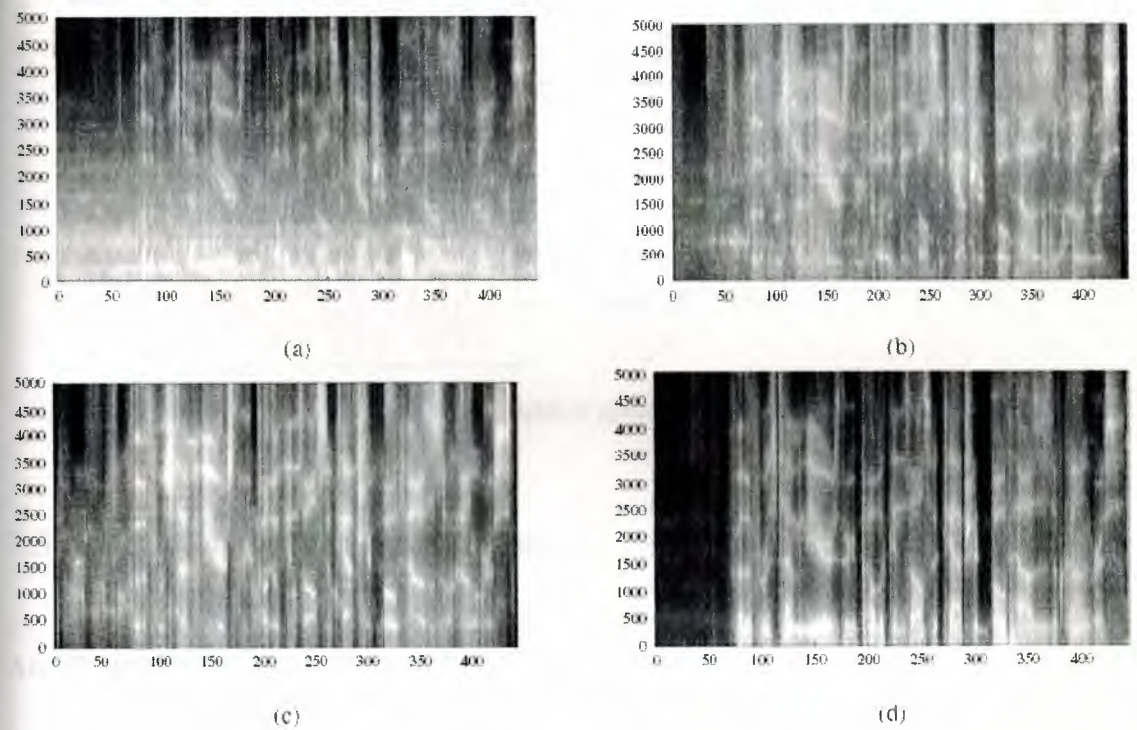


Figure 3.8 Comparison of LP spectrograms from: (a) speech in train noise; (b) pre-cleaned speech; (c) formant synthesized speech; and (d) clean speech.

3.2.4.2 De-noising of Speech Excitation Signal

The noisy speech excitation can be modeled, Figure 3.9, as a combination of the harmonic and the fricative noise content of the excitation and the residual background noise remaining after the inverse LP filtering as

$$e(m) = g_h \sum_{k=1}^{N} [a_k \sin(2\pi k F_0(m)) + b_k \cos(2\pi k F_0(m))] + g_n v(m) + n_r(m) \quad (3.20)$$

where $F_0(m)$ is the time-varying fundamental frequency of speech excitation, a_k and b_k are the amplitude of each excitation harmonic, v(m) is the noise-like excitation, g_h and g_n are the mixture weights and $n_r(m)$ is the background noise after inverse LP filtering.

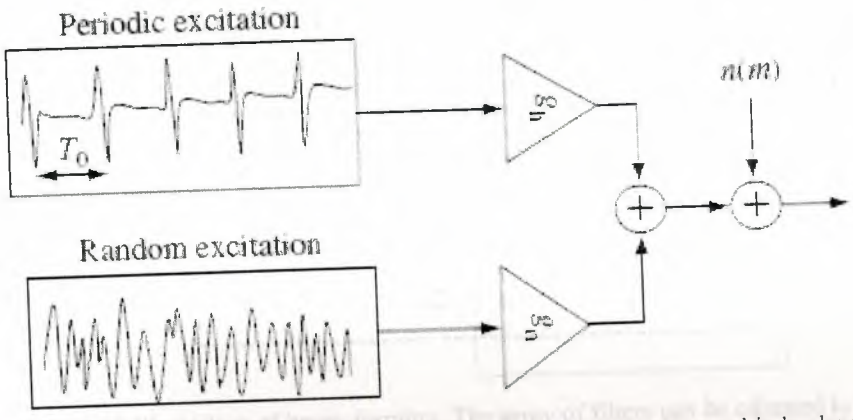


Figure 3.9 A harmonic plus noise model of speech excitation observed in background noise.

3.3 Speech Distortion Measurements

The most commonly used measure for quality of speech is signal-to-noise ratio. An average SNR measure is defined as

$$SNR = 10 log_{10} \left(\frac{P_{\text{Signal}}}{P_{\text{Noise}}} \right) \text{dB} \quad (3.21)$$

where $x_k(m)$ and $\hat{x}_m(m)$ are the clean signal and restored signal at frame m, N is the total number of frames and K is the number of samples in each frame. The segmental SNR of speech signals can fluctuate widely, as illustrated in Figure 3.10, which shows the variation of segmental SNR at average SNRs of 0, 5 and 10 dB.

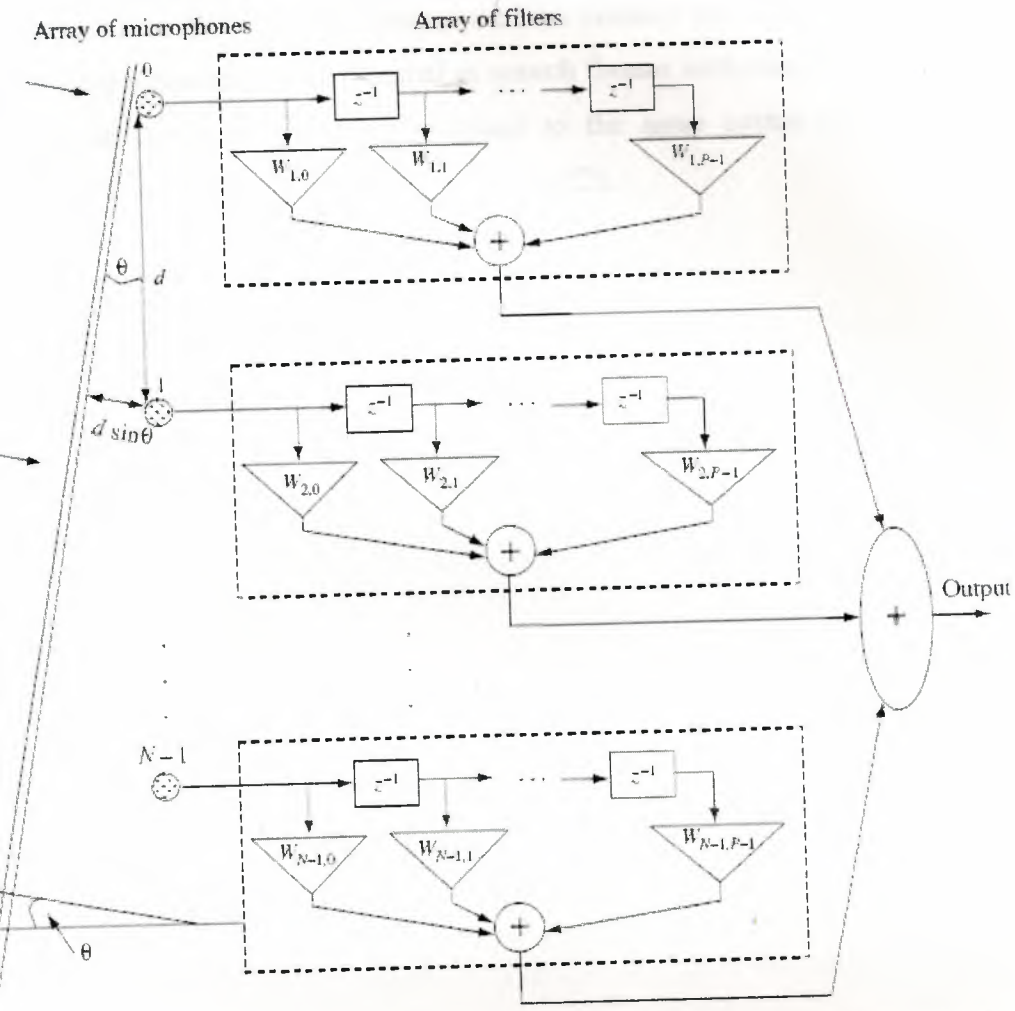


Figure 3.10 Illustration of beam-forming. The array of filters can be adjusted to change the 'looking' direction of the beam.

The signal-to-noise ratio is not the best measure of speech quality as it does not nto account the structure of the speech or the psychoacoustics of hearing. The a–Saito distance (ISD) measure is defined as

$$\mathrm{ISD}_{12} = \frac{1}{N}\sum_{j=1}^{N} \frac{[a_1(j)-a_2(j)]R_1(j)[a_1(j)-a_2(j)]^T}{a_1(j)R_1(j)a_1(j)^T} \qquad (3.22)$$

e $a_1(j)$ and $a_2(j)$ are the linear predication model coefficient vector calculated from and transformed speech at frame j and R1(j) is an autocorrelation matrix derived the clean speech. Owing to the asymmetry of the ISD measure (i.e. $ISD_{21} \neq ISD_{12}$), llowing segmental ISD measure is used:

$$\mathrm{ISD}_{sym} = (\mathrm{ISD}_{12}+\mathrm{ISD}_{21})/2 \qquad (3.23)$$

The ISD criterion is a more balanced measure of the distance between an original clean speech signal and a distorted speech signal as speech frames with relatively large SNRs do not dominate the overall distance measure to the same extent as in the more conventional SNR measures of Equations (3.21)–(3.22).

CHAPTER THREE

BUILDING THE USER INTERFACE

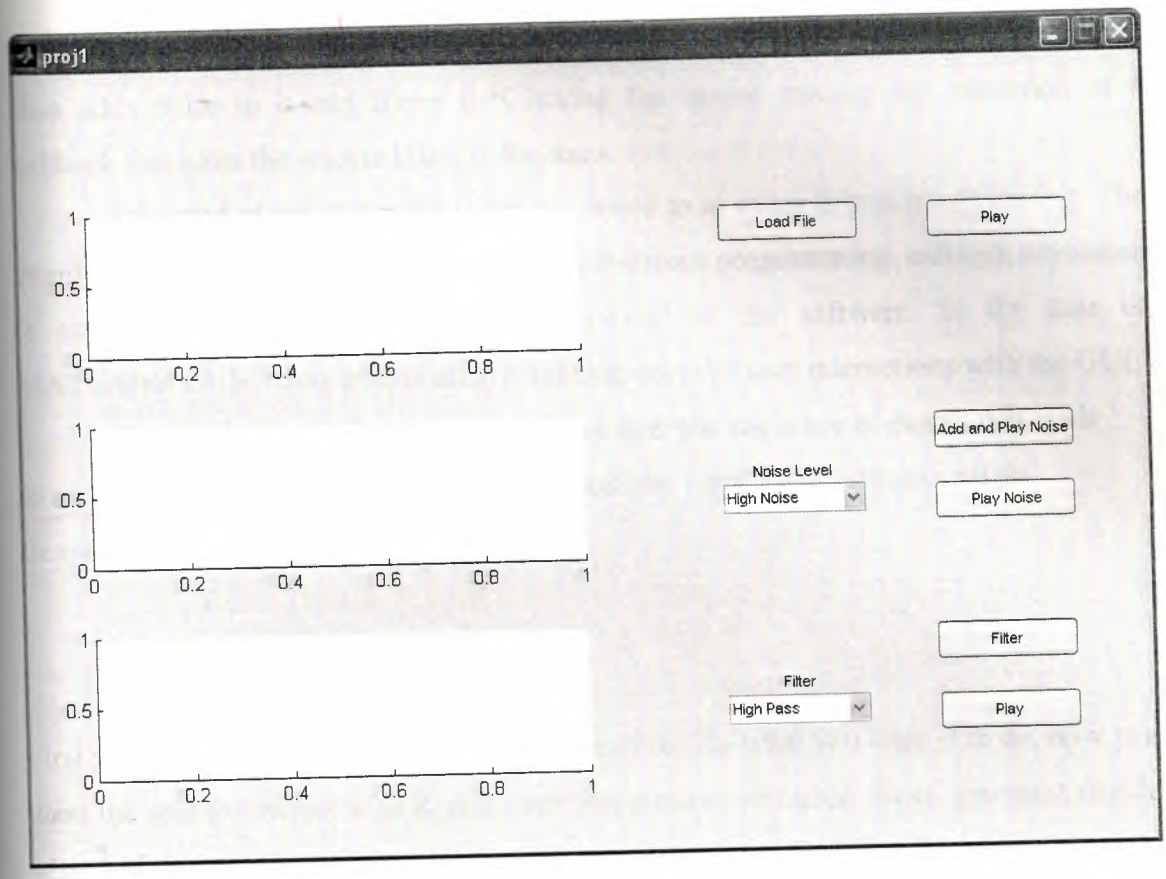
4.1 Introduction

The graphical user interface, or the GUI used in this project was done using Matlab. It is a simple program that shows how a crude working of a filter. As we know what a low-pass, high-pass or band-pass filter does is remove frequencies of certain values. This of course differs according to each filter. In this chapter we will see how a simple GUI is built.

4.1.1 What Is a GUI?

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand.

The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders—just to name a few. In MATLAB® software, a GUI can also display data in tabular form or as plots, and can group related components. The following figure illustrates a simple GUI.



The GUI contains

- An axes component
- A pop-up menu listing three data sets that correspond to MATLAB functions: peaks, membrane, and sinc
- A static text component to label the pop-up menu
- Three buttons that provide different kinds of plots: surface, mesh, and contour

When you click a push button, the axes component displays the selected data set using the specified plot.

4.1.2 How Does a GUI Work?

Each component, and the GUI itself, is associated with one or more user-written routines known as callbacks. The execution of each callback is triggered by a particular user action such as a button push, mouse click, selection of a menu item, or the cursor passing over a component. You, as the creator of the GUI, provide these callbacks.

In the GUI described in this project, the user loads a sound file of .wav format, then adds noise to it and filters it. Clicking the button triggers the execution of a callback that plots the selected data in the axes.

This kind of programming is often referred to as event-driven programming. The event in the example is a button click. In event-driven programming, callback execution is asynchronous, controlled by events external to the software. In the case of MATLAB® GUIs, these events usually take the form of user interactions with the GUI.

The writer of a callback has no control over the sequence of events that leads to its execution or, when the callback does execute, what other callbacks might be running simultaneously.

4.1.3 Where Do I Start?

First you have to design your GUI. You have to decide what you want it to do, how you want the user to interact with it, and what components you need. Next, you must decide what technique you want to use to create your GUI. MATLAB® software enables you to create GUIs programmatically or with GUIDE, an interactive GUI builder. It also provides functions that simplify the creation of standard dialog boxes. The technique you choose depends on your experience, your preferences, and the kind of GUI you want to create.

This table outlines some possibilities.

GUI	Technique
Dialog box	MATLAB software provides a selection of standard dialog boxes that you can create with a single function call. For links to these functions, see "Predefined Dialog Boxes" in the MATLAB Function Reference documentation.
GUI containing just a few components	It is often simpler to create GUIs that contain only a few components programmatically. Each component can be fully defined with a single function call.
Moderately complex GUIs	GUIDE simplifies the creation of such GUIs.
Complex GUIs with many components, and GUIs that require interaction with other GUIs	Creating such GUIs programmatically lets you control exact placement of the components and provides reproducibility.

4.2 Building the GUI

When we open Matlab at the toolbar we see a GUIDE button as you see on figure 4.1.

Figure 4.1

Clicking on it will open a window asking us if we want to create a new GUI or edit an existing one. We will open a new Default Blank GUI. This will open a window like that in Figure 4.2.

This is a blank GUI with no buttons or pop-menus. This is where we will input our various functions to our user interface (push buttons, slider etc.). As you see on the left side you see various things that can be added to the GUI.

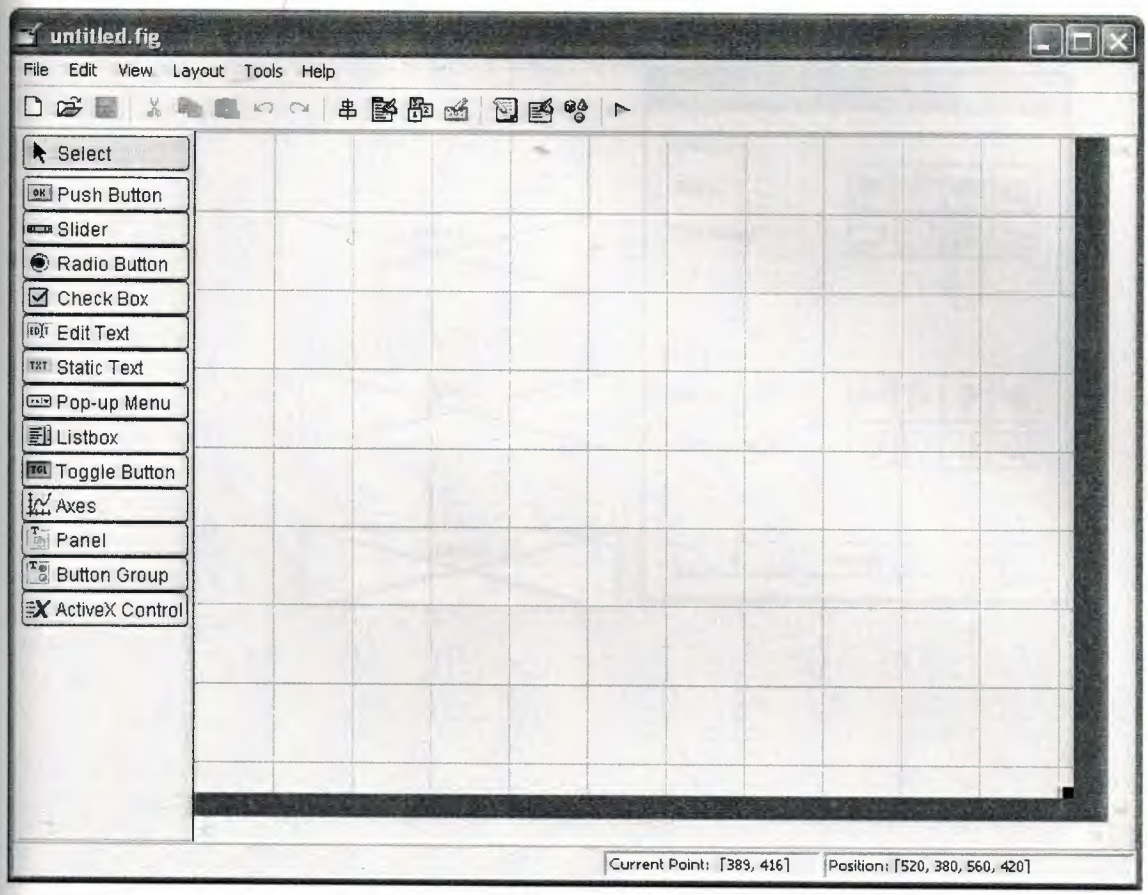


Figure 4.2

Let us try clicking in the Axes button and creating a couple of axe to our GUI. These axes will show us the different kinds of sound waves that we wish to see during our audio processing/speech enhancement. We can make the axes as big as we wish (according to the GUI window), but since we need three for this project let us try to keep them reasonable. If you see that the axes created are not aligned we can do so by choosing Align Objects from the Tools menu as seen in Figure 4.3. Once we have set and aligned our axes, we should get started on setting the push buttons that will work the different functions of our GUI.

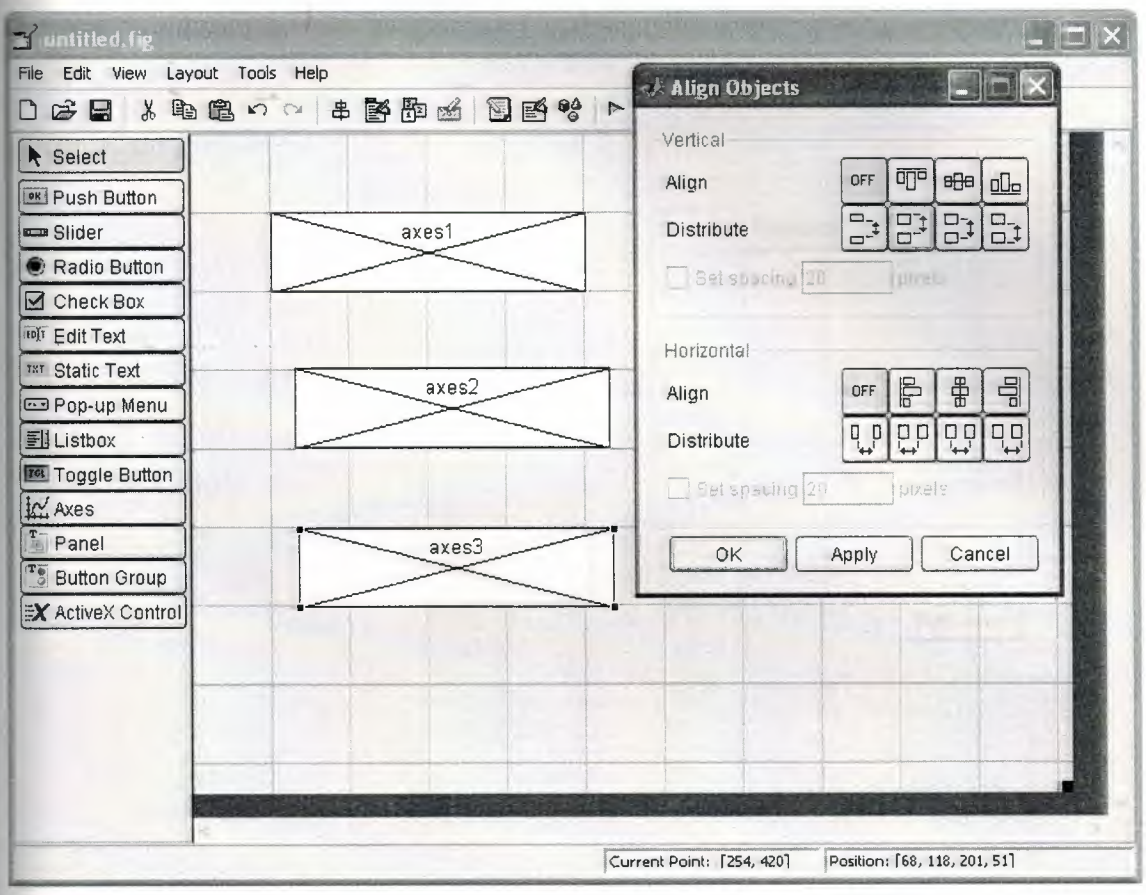


Figure 4.3

Next task in the building of our GUI is to add the push-buttons. Select the Push Button push from the component palette at the left of the Layout Editor and drag it into the layout area. Position them as shown in Figure 4.5. Use the Align Object tool from the menu if necessary.

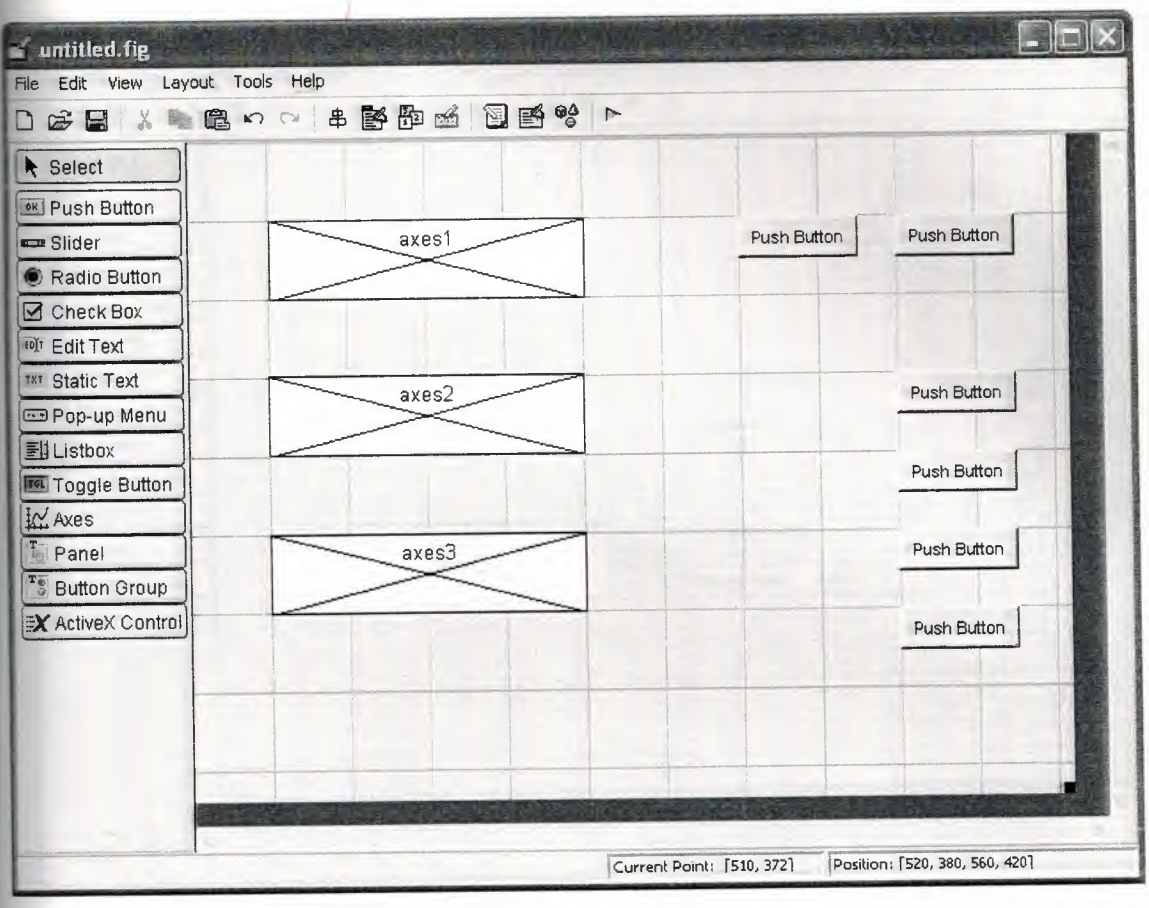


Figure 4.5

Once we have added our axes and the push buttons, we will add our pop-up menus. We do so by selecting pop-up menu and placing them in the layout area. Once we have placed them we should not forget that for the end user the pop-up menu should be explained as to what it is for. We just simply add some text on top of them by selecting static text and placing them on top of each menu respectively. Our GUI should look as in Figure 4.6.

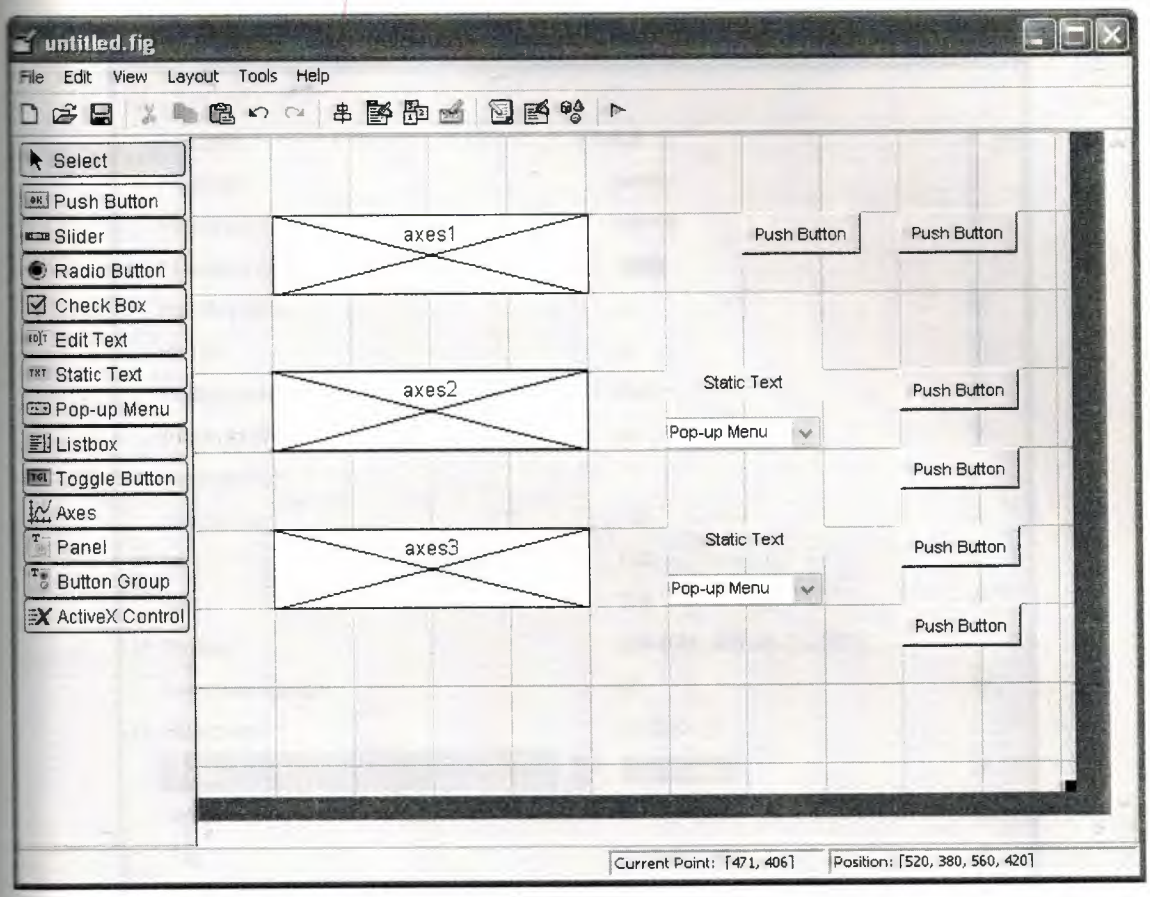


Figure 4.6

After we have placed all of our axes, push buttons and pop-up menus we are left with the task of naming them. To do this, select a push button. Then right click on it. A menu will appear. Select property inspector. A window will pop up as in Figure 4.7.

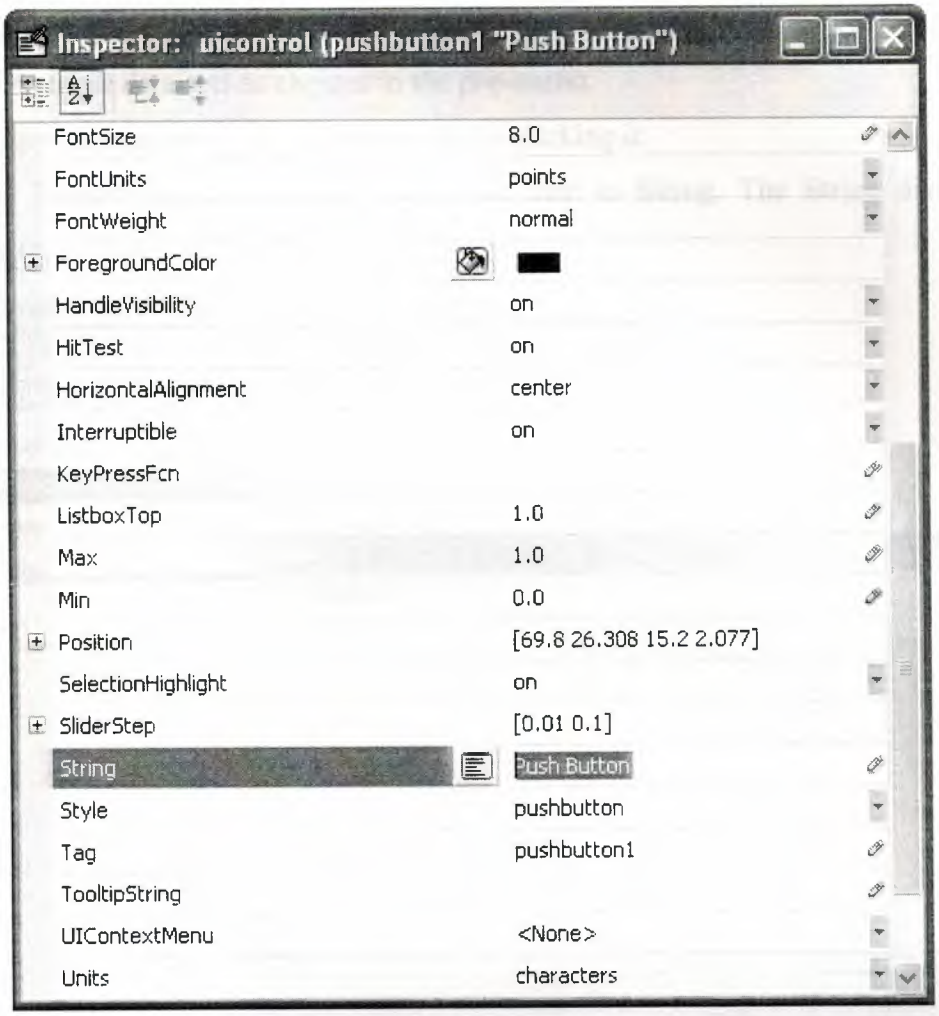


Figure 4.7

As shown in the highlighted area String, we will change the name of our button 'pushbutton1' to 'Load' as its function will be to load a sound of format .wav into our GUI program. Similarly we will change:

- Pushbutton2 to Play
- Pushbutton3 to Add Noise
- Pushbutton4 to Play Noise
- Pushbutton5 to Filter
- Pushbutton6 to Play Filtered

The pop-up menus each provide a choice of three data sets. Menu1: High Noise, Medium Noise, Low Noise. These correspond to the different forms of noise we will add to our input signal. Menu2: High Pass, Low Pass, Band Pass. These data sets

correspond to forms of filtering we will perform on our noisy signal. This shows you how to list those data sets as choices in the pop-menu.

1 In the layout area, select the pop-up menu by clicking it.

2 In the Property Inspector, click the button next to String. The String dialog box displays (as in Figure 4.8)

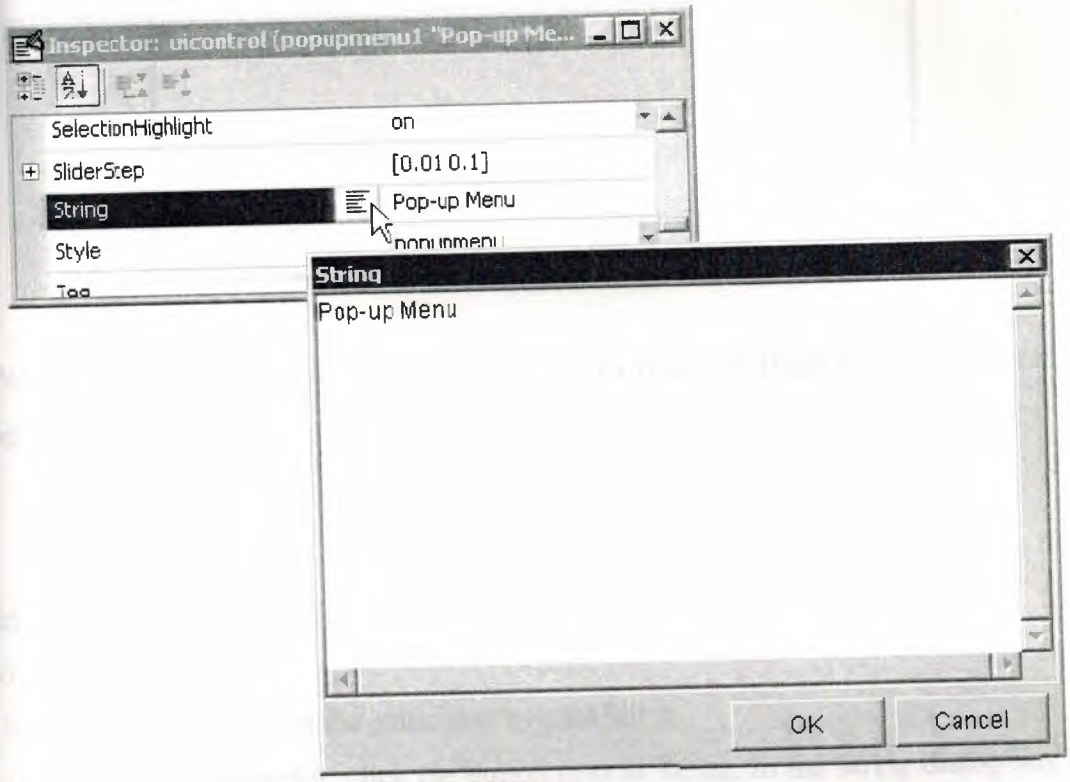
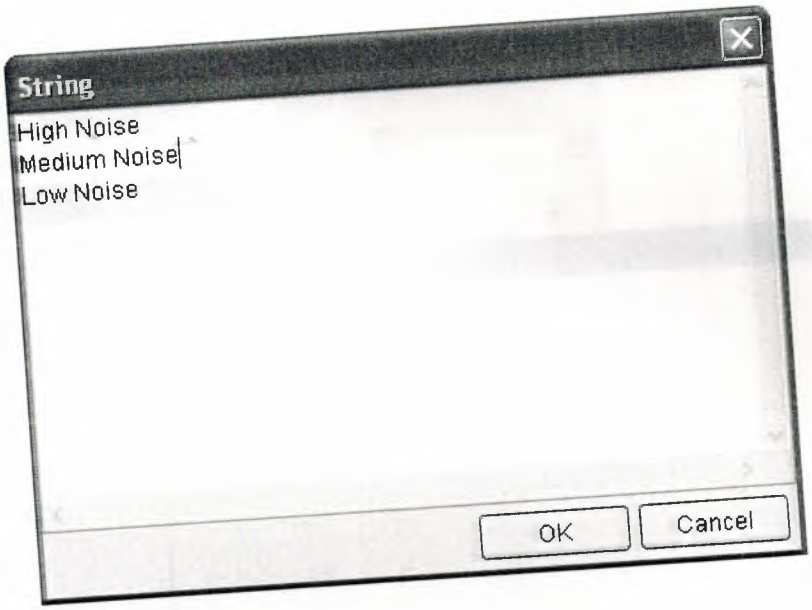
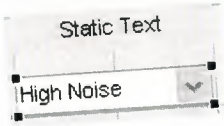


Figure 4.8

3 Replace the existing text with the names of the three data sets: High Noise, Low Noise and Medium Noise. Press **Enter** to move to the next line.



4 When you are done, click **OK**. The first item in your list, High Noise, appears in the pop-up menu in the layout area.

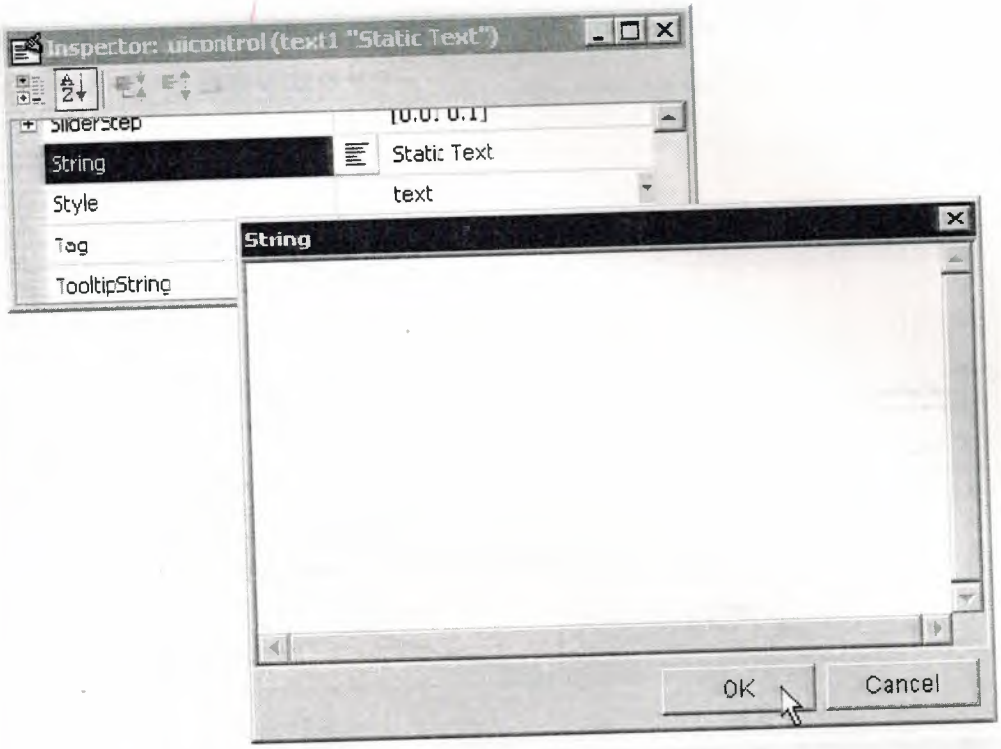


Similarly pop-up menu2 is done.

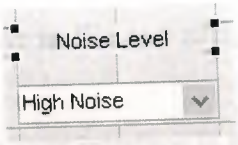
To edit the static text:

1 In the layout area, select the static text by clicking it.

2 In the Property Inspector, click the button next to String. In the String dialog box that displays, replace the existing text with the phrase Noise Level



3 Click **OK**. The phrase Select Data appears in the static text component above the pop-up menu.



Similarly we will perform the same tasks to our static text2.

4.3 Completed Layout

In the Layout Editor, your GUI now looks like Figure 4.9. This and the next step is to save the layout.

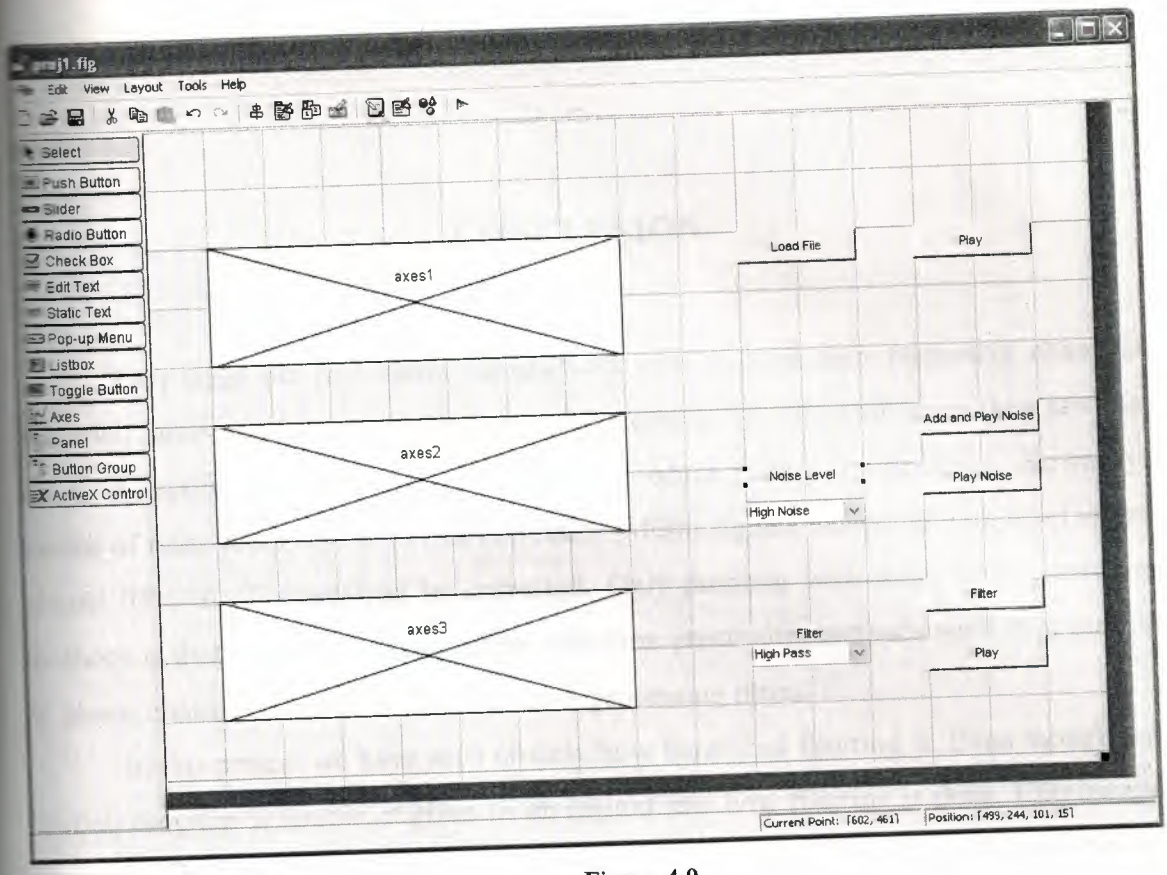


Figure 4.9

Next all we have to do is to edit the m-file and assign a callback for each button, axes and pop-up menu.

The m-file for this project's user interface is given in the appendix section.

REFERENCES

[1] Advanced Digital Signal Processing and Noise Reduction

[2] Creating Graphical User Interface in Matlab

[3] Wikipedia

APPENDIX

Matlab Code for Speech enhancement GUI

```
function varargout = proj1(varargin)
% PROJ1 M-file for proj1.fig
%      PROJ1, by itself, creates a new PROJ1 or raises the existing
%      singleton*.
%
%      H = PROJ1 returns the handle to a new PROJ1 or the handle to
%      the existing singleton*.
%
%      PROJ1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROJ1.M with the given input
arguments.
%
%      PROJ1('Property','Value',...) creates a new PROJ1 or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before proj1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to proj1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help proj1

% Last Modified by GUIDE v2.5 25-May-2008 13:30:25

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                   'gui_OpeningFcn', @proj1_OpeningFcn, ...
                   'gui_OutputFcn',  @proj1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end


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


% --- Executes just before proj1 is made visible.
function proj1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to proj1 (see VARARGIN)

% Choose default command line output for proj1
axes(handles.axes1); cla;
axes(handles.axes2); cla;
axes(handles.axes3); cla;


handles.a=0;
handles.x2=0;
handles.selectednoise=1;
handles.selectedfilter=1;
handles.fileLoaded = 0;
handles.fileNoisy = 0;
handles.fileFinal = 0;

handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes proj1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = proj1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

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


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[FileName,PathName] = uigetfile({'*.wav'},'Load Wav File');
[x,Fs] = wavread([PathName '/' FileName]);
handles.x = x ./ max(abs(x));
handles.Fs = Fs;
axes(handles.axes1);
time = 0:1/Fs:(length(handles.x)-1)/Fs;
handles.time = time;
plot(handles.time, handles.x);
xlabel('time (s)')
axis([0 max(time) -1 1]);
handles.fileLoaded = 1;

handles.fileNoisy = 0;
handles.fileFinal = 0;

guidata(hObject, handles);
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (handles.fileLoaded==1)
    sound(handles.x, handles.Fs);
end

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1
S = get(hObject,'Value');

 switch (S)
     case (1)
         handles.selectednoise=1;
     case (2)
         handles.selectednoise=2;
     case(3)
         handles.selectednoise=3;
 end

 guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

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


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.fileLoaded==1
    d = (length(handles.x)-1)/handles.Fs;
    done=0;

    switch (handles.selectednoise)
        case (1)
            %high noise
            noise=sqrt(0.1)*randn(1, length(handles.x));
            done=1;
        case (2)
            %medium noise
            noise=sqrt(0.01)*randn(1, length(handles.x));
            done=1;
        case (3)
            %low noise
            noise=sqrt(0.001)*randn(1, length(handles.x));
            done=1;
    end

    if (done==1)
        handles.fileNoisy = 1;
        handles.x2 = handles.x + noise.';
```

```
        axes(handles.axes2);
        time = 0:1/handles.Fs:(length(handles.x2)-1)/handles.Fs;
        plot (time, handles.x2);
        xlabel('time (s)')
    end
end

guidata(hObject, handles);


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu2
v = get(hObject,'Value');

switch (v)
    case (1)
        handles.selectedfilter=1;
    case (2)
        handles.selectedfilter=2;
    case(3)
        handles.selectedfilter=3;
end

guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

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


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Done=0;
filter=0;

if (handles.fileNoisy==1)

switch (handles.selectedfilter)
    case (1) % HIGHPASS
        filter = ones(1, length(handles.time));initializaiton by 1
        filter(1, 1 : (4000*length(handles.time)/handles.Fs)) = 0;
        filter(1, length(handles.time) -
(4000*length(handles.time)/handles.Fs) : length(handles.time)) = 0;
        Done=1;
    case (2) % LOWPASS
        filter = zeros(1, length(handles.time));
        filter(1, 1 : (2000*length(handles.time)/handles.Fs)) = 1;
        filter(1, length(handles.time) -
(2000*length(handles.time)/handles.Fs) : length(handles.time)) = 1;ber
```

```
        Done=1;
    case (3) %BANDPASS
        filter = zeros(1, length(handles.time));
        filter(1,
(2000*length(handles.time)/handles.Fs):(4000*length(handles.time)/hand
les.Fs)) = 1;
       filter(1, (length(handles.time) -
(4000*length(handles.time)/handles.Fs)):(length(handles.time) -
(2000*length(handles.time)/handles.Fs))) = 1;

        Done=1;
end
end
a = fft(handles.x2);                % FFT
a = a .* filter.';                  % filtering
a = ifft(a);                        % inverse FFT
a = real(a);
handles.a=a;
Done=1;

if (Done==1)
    handles.fileFinal=1;
    axes(handles.axes3);
    time = 0:1/handles.Fs:(length(handles.a)-1)/handles.Fs;
    %plot (time, handles.a);
    %specgram(handles.a)

    x = linspace(0, handles.Fs/2, length(handles.time)/2);
    t = fft(handles.a);
    t = t .* conj(t);
    size(x)
    size(t(1:(length(handles.time)/2))./max(t))

    semilogy(x', t(1:(length(handles.time)/2))./max(t));
    xlabel('Frequency (Hz.)')
    title('Spectrum of Filtered Sound')
end

guidata(hObject, handles);
```

```
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.fileFinal==1)
    sound(handles.a, handles.Fs);
end


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.fileLoaded==1)
    sound(handles.x2, handles.Fs);
end
```