Fall 12-2011

Compressive Sensing for Speech Signals in Mobile Systems

Sabir Ahmed

COMPRESSIVE SENSING FOR SPEECH SIGNALS IN MOBILE SYSTEMS

by

SABIR AHMED

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
Department of Electrical Engineering.

Dr. Hector A. Ochoa, Ph.D., Committee Chair

College of Engineering and Computer Science

The University of Texas at Tyler
December 2011

The University of Texas at Tyler
Tyler, Texas.

This is to certify that the Master's Thesis of

SABIR AHMED

has been approved for the thesis requirement on
July 26th, 2011
for the Master of Science in Electrical Engineering

Approvals:

Committee Chair: Hector A. Ochoa, Ph.D.

Member: Mukul V. Shirvaikar, Ph.D.

Member: Ron J. Pieper, Ph.D.

Chair and Graduate Coordinator: Mukul V. Shirvaikar, Ph.D.

Dr. James K. Nelson, Jr., Ph.D., P.E.,
Dean, College of Engineering and Computer Science

Acknowledgements

I would like to add a few heartfelt words for the people who guided me in numerous ways. First, I would like to thank my parents who supported me in every phase of life and encouraged me to achieve my goals. I express my profound gratitude to my advisor Dr. Hector A. Ochoa for his constant support, patience, guidance and allowing me to choose this topic. His esteemed guidance helped me to learn and enjoy the real essence of research. Without his guidance and assistance, this thesis would not have been possible.

I owe the greatest measure of magnitude to Dr. Mukul V. Shirvaikar for his support and encouragement, which fueled in me the confidence to work with dedication. I would honestly like to express my thanks to Dr. Ron Pieper for taking time to be part of my committee and reviewing my work.

I would like to thank the entire Electrical Engineering Department and The University of Texas at Tyler for supporting me throughout my Masters degree. Finally, I would like to convey my earnest thanks to my best friends Raheez Reppal, Kenza Amrani and Swetha Gazabare for their constant encouragement, which filled me with zeal and buoyancy for completing my research.

Table of Contents

List of Figures

List of Tables

Abstract

COMPRESSIVE SENSING FOR SPEECH SIGNALS IN MOBILE SYSTEMS

Sabir Ahmed

Thesis Chair: Hector A. Ochoa, Ph.D.

The University of Texas at Tyler
August 2011

Compressive sensing is an emerging and revolutionary technology that strongly relies on the sparsity of the signal. In compressive sensing the signal is sparsely compressively sampled by taking a small number of random projections of the signal, which contain most of the salient information. Compressive sensing has been previously applied in areas like: image processing, radar systems and sonar systems. This research work will discuss the potential implementation of compressive sensing in mobile communication systems and how it will influence their data rates.

In a typical mobile communication system, the signal of interest is sampled at least at the Nyquist rate. The Nyquist sampling theorem states that the frequency used to sample a signal should be at least twice the maximum frequency contained within the signal. However, this is not the most efficient way to compress the signal, as it places a lot of burden in sampling the entire signal while only a small percentage of the transform coefficients are needed to represent it. The recent results in compressive sampling (also known as compressive sensing) provide a new way to reconstruct the original signal with a minimal number of observations. In compressive sensing the significant information about the signal/image is directly acquired, rather than acquiring the significant information that will be eventually thrown away.

The goal of this research is to propose a new mobile communication system which employs compressive sampling to compress the speech signal at the transmitter and decompress it at the receiver. The expected results from the proposed system will be an increment in the data rates of these systems. In order to simulate how compressive sensing could be applied, a small speech signal was recorded in MATLAB. The signal at the transmitter is then multiplied by the measurement matrix which in this case is composed of randomly generated numbers. The measurement matrix is chosen in such a way that the sparse signal can be exactly recovered at the receiver using one of the different optimization techniques available. Once the signal has gone through the process of compressive sampling, it is ready to be transmitted through the mobile system. The transmitted signal is then reconstructed by the receiver from a significantly small number of samples by using any of the multiple optimization techniques available. The algorithm is simulated in MATLAB. The results show that if a threshold window is applied to the transmitted speech signal and the length of the signal is kept constant, the compression rate of the speech signal is increased.

Chapter One

Introduction

Mobile communications is one of the most important research areas in the field of telecommunications. It is predicted that within a few decades a considerable number of data and voice connections will become partially or completely wireless. One of the main challenges in wireless systems is to provide higher data rates in mobile environments. In order to achieve this goal a new sampling technique called compressive sampling can be used instead of the traditional sampling technique. Compressive sensing (CS) can be implemented in a mobile system because most of the signals in the real world have a sparse representation under certain domain transformations. In a typical communication system, the signal is sampled at least at twice the highest frequency contained in the signal. However, this limits efficient ways to compress the signal, as it places a huge burden on sampling the entire signal while only a small number of the transform coefficients are needed to represent the signal [1].

On the other hand, compressive sampling provides a new way to reconstruct the original signal from a minimal number of observations. CS is a sampling paradigm that allows us to go beyond the Shannon limit by exploiting the sparsity structure of the signal. It allows us to capture and represent the compressible signals at a rate significantly below the Nyquist rate. The sampling step is very fast because it employs non adaptive linear projections that preserve the structure of the signal. The signal is then reconstructed from these projections by using different optimization techniques. During compressive sampling only the important information about a signal is acquired, rather than acquiring the important information plus the information of a signal which will be eventually discarded at the receiver. By implementing this theory, a new mobile communication system has been proposed, in which compressive sampling or compressive sensing is used to compress and decompress the speech signal at the transmitter and at the receiver to increase the data rates.

In recent years, various signal sampling schemes have been developed in the market place. Vetterli et al [2] presented a method to uniformly sample continuous-time signals, such as non-uniform splines and stream of Dirac. However, such sampling methods are difficult to implement because of the need to have sufficient information about the reconstruction kernel before sampling the signal. In the meantime, the emerging compressive sensing theory [1, 3] shows that an unevenly sampled discrete signal can be perfectly reconstructed with high probability of success by using different optimization techniques and by considering fewer random projections or measurements compared to the Nyquist standard. The key elements that need to be addressed before using compressive sensing are the following: how to find the transform domain in which the signal has a sparse representation, how to effectively sample the sparsely signal in the time domain and finally, how to recover the original signal from the samples by using optimization techniques.

In summary, the large amount of data needed to sample at the Nyquist rate, especially for speech, image and video signals, motivates the study of compressive sensing as a feasible solution for future mobile communication systems. Sparse signals are defined as signals that can be represented by a limited number of data points in the transform domain. Many real-world signals can be classified into this category using an appropriate transform domain. For instance, if signal x is a sine, it is clearly not sparse, but its Fourier transform is extremely sparse. Another example is a piecewise constant image away from edges of finite length that has a sparse gradient. The consequences of acquiring large amounts of data, added to the overhead of compression can be improved by using compressive sensing. As a result, there are potential savings in terms of energy, memory and processing.

In the proposed approach, a speech signal was recorded and compressively sampled using a measurement matrix. The output of the compressive sensing algorithm is the observation vector which is transmitted to the receiver. At the receiver section signal is reconstructed from a significant small numbers of samples by using different optimization techniques such as l_1-norm or convex optimization. MATLAB simulations were performed to compress the speech signal below the Nyquist and reconstruct it using one of the multiple optimization techniques available without losing any important information.

Chapter Two

Background

2.1 Basic Theory of Compressive Sensing

The theory of compressive sensing was developed by Candes *et al* [3] and Donoho [1] in 2004. It involves taking random projections of the signal and recovering it from a small number of measurements using optimization techniques. In a traditional sampling theorem, the signal is sampled using Nyquist rate, whereas with the help of compressive sensing the signal is sampled below the Nyquist rate. This is possible because the signal is transformed into a domain in which it has a sparse representation. Then the signal is reconstructed from the samples using one of the different optimization techniques available. The basic block diagram of a compressive sensing system is shown in Figure 1.

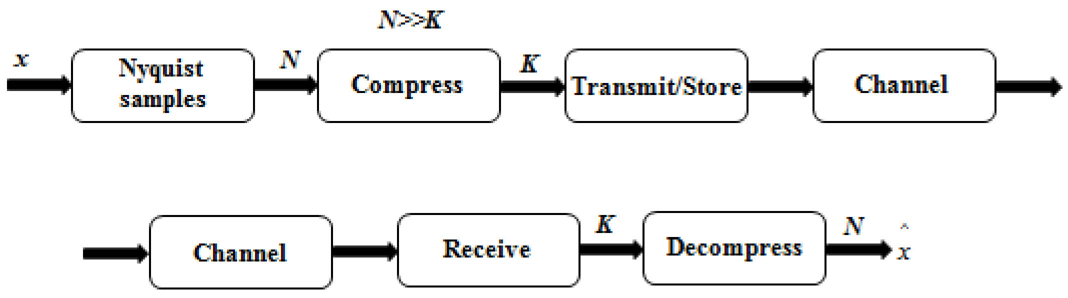


Figure 1. Basic compressive sensing block diagram [4]

2.1.1 Signal Representation and Sparsity

Signal representation and sparsity play a vital role in compressive sensing. Let $\mathbf{x} \in R^L$ represent a real signal, assuming that the signal $\mathbf{x}$ is sparse in the orthogonal basis $\boldsymbol{\psi} = \{\psi_1, \psi_2, \psi_3 ....\psi_N\}$ where N is the length of the signal, then $\mathbf{x}$ can be represented by a linear combination of K *(K<<N)* basis functions as

$$\mathbf{x} = \sum_{i=1}^{K} \boldsymbol{\theta}_{n_i} \boldsymbol{\psi}_{n_i} \tag{2.1}$$

where $\boldsymbol{\psi}_{n_i} \in \boldsymbol{\psi}$, $n_i \in \{1, 2, 3......, N\}$. Let $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3 ....\theta_N]^T$ be the vector of coefficients for the signal $\mathbf{x}$ in $\boldsymbol{\psi}$. The random measurement of the signal $\mathbf{x}$ can be represented as

$$\mathbf{y} = \boldsymbol{\phi\theta}, \ \boldsymbol{\phi} : M \times N , \quad K < M << N \tag{2.2}$$

where $\boldsymbol{\phi}$ is the uniform random measurement matrix, **y** is the measurement vector of the signal **x**, $\boldsymbol{\theta}$ is the vector of coefficients for the signal **x** and $M = cK(c < 1)$ represents the number of measurements required for perfect reconstruction. If all the entries of $\boldsymbol{\phi}$ are drawn from a Gaussian distribution, the signal can be exactly reconstructed with high probability of success when the constant '*c*' is between three and five [5]. The procedure used to ensure the sparsity of the signal is called transform coding, which is performed by the following four steps [6]

i. Obtain the full N-points signal **x** using the Nyquist rate
ii. Compute the complete set of transform coefficients (e.g. DFT)
iii. Locate the *K* largest coefficients and throw away the smallest coefficients
iv. Multiply signal by the measurement matrix to obtain the observation vector of length M.

Figure 2 shows an example of how compressive sensing can be used to compress a signal below Nyquist rate [7]. In this example, the original sampled signal is composed of 300 samples. The intent is to reconstruct the signal using only 30 samples. Figure 2(a) shows the time domain representation of the sampled signal. From this figure, it is evident that by selecting only 30 samples (red dots) from the 300 samples it will be impossible to reconstruct the original signal perfectly. On the other hand, by applying compressive sensing to the frequency representation of the signal it is possible to perfectly reconstruct it from a significant small number of samples. In order to achieve this goal it is necessary to implement an optimization techniques. However, not every optimization technique can be used for this purpose. For example, Figure 2(c) represents the reconstructed spectrum using the l_2 minimization. Clearly, there are significant differences between the signal in Figure 2(b) and the signal from Figure 2(c).

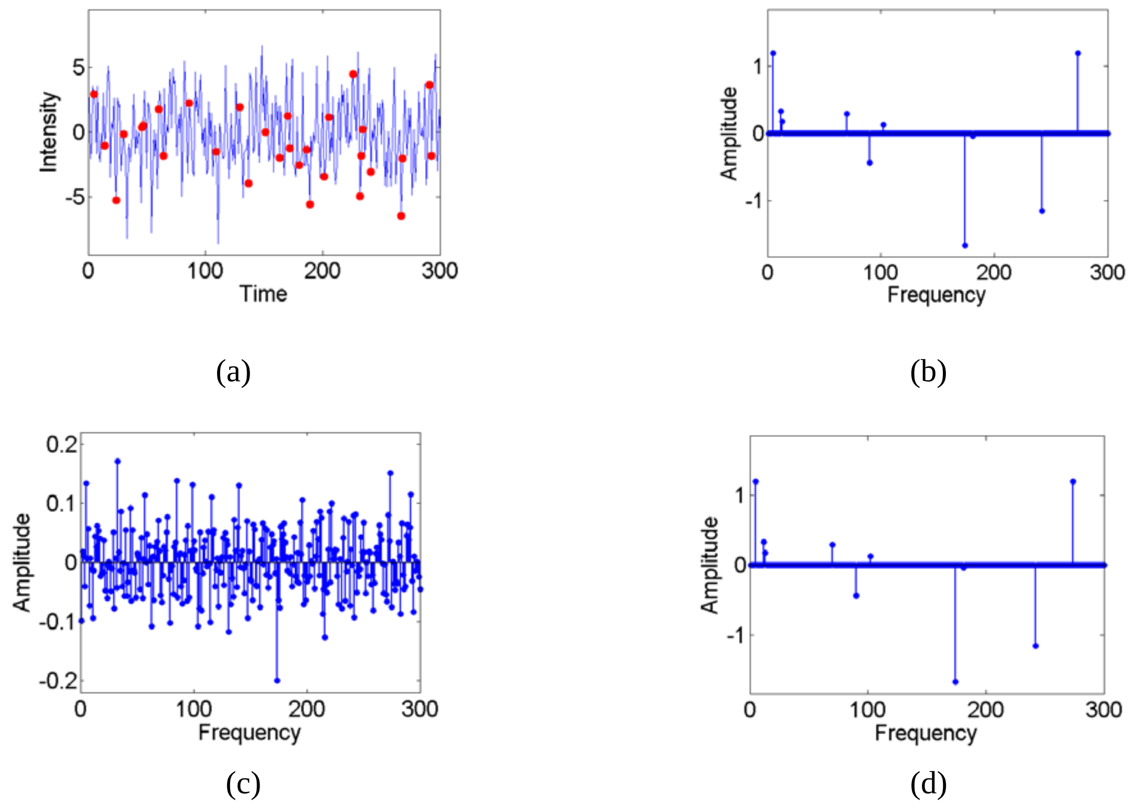


Figure 2. Sampling using compressive sensing (a) time domain representation of the signal composed of 300 samples, (b) Fourier spectrum of the signal to be encoded, (c) reconstruction of the Fourier spectrum via l_2 minimization, (d) reconstruction of the Fourier spectrum via l_1 minimization

In contrast, the reconstruction using the l_1 minimization gives as a result a perfect reconstruction. This can be clearly seen by comparing Figure 2(b) and Figure 2(d). In summary, optimization techniques based on l_1 minimization are desired when compressive sensing is used.

2.1.2 Measurement Matrix

In this section, special emphasis is given to represent the signal with an incoherent basis. Let us consider the linear measurement process depicted in Figure 3 that computes M < N inner products between **x** and the collection of vectors $\{\boldsymbol{\phi}_j\}_{j=1}^{M}$ via $\mathbf{y}_j = \langle \mathbf{x}, \boldsymbol{\phi}_j \rangle$, where $j=1,\ldots,M$, $\boldsymbol{\phi}_j^T$ denotes the transpose of ϕ_j and $\langle\cdot,\cdot\rangle$ denotes the inner products. Let a M×1 vector $\boldsymbol{y}$ be the measurements array y_j. In matrix notation the vector $\boldsymbol{y}$ is obtained using the following expression

$$\boldsymbol{y} = \mathbf{\Phi x} = \mathbf{\Phi\Psi\alpha} \tag{2.3}$$

where $\mathbf{\Phi}$ is a $M \times N$ measurement matrix with each row been a measurement vector $\boldsymbol{\phi}_j^T$ and $\boldsymbol{\alpha}$ is the coefficient vector with K non zeroes element. It has been seen that some of the measurement matrices can be used in any scenario, in the sense that they are incoherent with any fixed basis $\mathbf{\Psi}$ such as Gabor, spikes, sinusoidal and wavelets. The compressive sensing measurement process with K-sparse coefficient vector **x** is depicted in Figure 3.

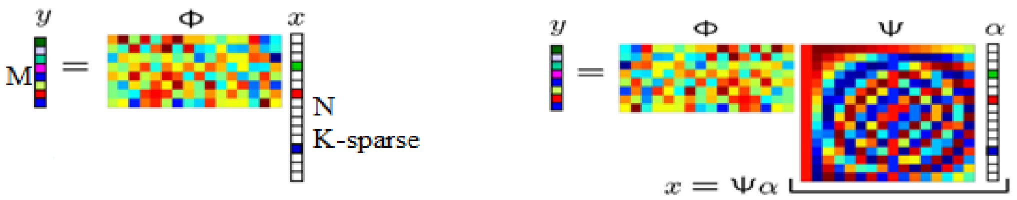


Figure 3. Compressive sensing measurement process [4]

The measurement matrix plays a vital role in the process of recovering the original signal. This poses an interesting problem: "How should one design a measurement matrix $\mathbf{\Phi}$ that is basically a collection of N vectors in K dimensions?" There are two types of measurements matrices that can be used in compressive sensing: The Random measurement matrix and the predefined measurement matrix. The fundamental revelation is that, if a signal **x** composed of N samples is sparse then the actual signal can be reconstructed using $M \geq O(K\log(N/K)) \ll N$ linear projection of **x** onto another basis. Furthermore, **x** can be perfectly reconstructed using different optimization techniques. If $\mathbf{\Phi}$ is a structurally random matrix, its rows are not stochastically independent because they are randomized from the same random seed vector. The random matrix is transposed and then orthogonalized. This will have the effect of creating a matrix that represents an orthonormal basis. In a predefined measurement matrix, the matrix is created by using function like the Dirac functions and

Sine functions. In this case, the signal is multiplied by several Dirac functions centered at different locations to obtain the observation vector. Then the speech signal can be reconstructed using the l_1 normalization method by using the observation vector and the predefined measurement matrix.

Linear programming is another procedure that plays a vital role in reconstructing the original signal. It is a mathematical approach designed to get the best outcome in a given mathematical model, which is a special case of mathematical programming. Linear programming can be expressed in the following canonical from:

$$\begin{aligned} &\text{maximize } \mathbf{c}^T\mathbf{x} \\ &\text{subject to } A\mathbf{x} \le b \end{aligned} \tag{2.4}$$

where $\boldsymbol{x}$ represents the variable that is to be determined, c and b are vectors of coefficients and A is a matrix of coefficients. The above expression which has to be maximized or minimized is called the objective function and the equation $A\mathbf{x} \le b$ defines the constraints over which the objective function has to be optimized. At the end the reconstruction of the speech signal depends upon the observation vector and the measurement matrix.

2.1.3 Signal Reconstruction in Compressive Sensing

Recent developments in signal theory have shown that a sparse signal is a useful model in areas such as: communications, radar and image processing. Therefore the assumption that every signal can be represented in a sparse form has helped in the compression of the signal of interest. The perfect reconstruction of a signal **x** depends on the measurement matrix **Φ** and the measurement vector **y**. The compressive sensing theory tells us that when the matrix **ΦΨ** has the Restricted Isometric Property (RIP) which are nearly orthonormal then it is possible to recover the K largest significant coefficients from a similar size set of $M = O(K\log(N/K))$ measurements of **y**. As a result, the sparse signal can be reconstructed by different optimization techniques such as l_1 norm and convex optimization. The first minimization technique which has been used to reconstruct the signal is the l_1 minimization

$$(P1)\min \| \mathbf{x} \|_{l1} \text{ Subject to } \mathbf{\Phi x = y} \tag{2.5}$$

which is also known as *basis pursuit* (P1). The goal of this technique is to find the vectors with the smallest l_1-norm

$$\| \mathbf{x} \|_1 = \sum_{i=1}^{n} | \mathbf{x}_i | \tag{2.6}$$

It is also known as Taxicab norm Manhattan norm. The name relates to the distance a taxi has to drive in rectangular street grid to get from the origin to the point **x**. The distance obtained from this norm is called the Manhattan distance or l_1 distance. The results obtained in [8, 9] shows that, if a signal **x** is sufficiently sparse, the signal can be reconstructed by using *basis pursuit* (P1). The other optimization technique known as convex optimization (*cvx*) will solve many medium and small scale problems. By using *cvx* the signal is minimized in order to reconstruct the original signal [10]

2.2 Compressive Classification

This section will discuss the classification of compressive sensing based on its application. It deals with the effect of compressive sensing and its performance. The compressive sensing framework is useful in a wide range of statistical inference tasks e.g. in finding the solution to detection and estimation problems in which it is able to reconstruct the original signal from the measurements.

2.2.1 Classical Detector

The classical detector is one of the classifications in compressive sensing. Let us say that there are two hypotheses concerning the signal; that it is present in the measurement or it is not. The classical Neymon-Pearson (NP) detector uses a likelihood ratio test where the sufficient statistic $\mathbf{t} \equiv \langle \mathbf{y}, \mathbf{x} \rangle$ is compared against a threshold γ. Here **y** represents the measurements, **x** is the original signal and γ is set to achieve a certain probability of false alarm $P_F \leq \alpha$ for $0 \geq \alpha \leq 1$. It is shown that (for example, see [11]):

$$P_D(\alpha) = Q(Q^{-1}(\alpha) - \sqrt{SNR}) \tag{2.7}$$

where Q(.) is the flipped version of standard Gaussian cumulative distribution function.

2.2.2 Compressed Detector

The theory described in [12] can be easily extended when the measurements are made using a compressed sampler. For this section, the following hypotheses are considered:

$$H_0 : \mathbf{y} = \boldsymbol{\phi} n \tag{2.8}$$

$$H_1 : \mathbf{y} = \boldsymbol{\phi}(\mathbf{x} + n) \tag{2.9}$$

where $n \sim N(0, \sigma^2)$ is white Gaussian noise with mean zero with a standard variance. It is easy to show that the sufficient value is given by $\tilde{\mathbf{t}} \equiv \langle \mathbf{y}, \boldsymbol{\phi}\mathbf{x} \rangle$. The probability of false alarm is approximately given by the following equation [12]:

$$P_D(\alpha) \approx Q(Q^{-1}(\alpha) - \sqrt{M / N}\sqrt{SNR}) \tag{2.10}$$

where M is the number of random projections and N is the sparsity of the signal. By comparing equation (2.7) and (2.10) it can be seen that the performance of the detector will be deteriorated with the decrease of M and the performance also depends upon the rate of degradation of the SNR.

2.2.3 Compressive Classifier

Let us consider a set $X = \{\mathbf{x}_i\}$ of signals for a multi-class classifier for which the result has been generalized. The detection rule is given by $\hat{\mathbf{x}} = \min_{\mathbf{x}_i \in \mathbf{X}} \left\| \mathbf{y} - \phi \mathbf{x}_i \right\|_2$. In this classification the distances are not preserved, rather they are uniformly shrunken when compared to the matrix $\boldsymbol{\phi}$ where the distance is preserved. For any signal **x**, and typical value of ε , with probability of at least $1-\delta$, the following expression holds for all $\mathbf{x}_i$, [12]

$$(1-\varepsilon)\sqrt{M / N} \le \frac{\| \phi(\mathbf{x} - \mathbf{x}_i) \|_2}{\| (\mathbf{x} - \mathbf{x}_i) \|_2} \le (1+\varepsilon)\sqrt{M / N} \tag{2.11}$$

It can been seen that the effect of noise can be amplified with the transformation as it is dependent on the signal to noise ratio (SNR).

2.3 Optimization Techniques

Signal reconstruction plays an important role in compressive sensing theory where the signal is reconstructed or recovered from a minimum number of measurements. By using optimization techniques it is possible to recover the signal without losing the information at the receiver. There has been a series of papers related to the theory of signal recovery from highly incomplete information.

2.3.1 l_1 Minimization

A recent series of papers have developed a theory of signal recovery from highly incomplete information. The results states that a sparse vector $\mathbf{x} \in R^N$ can be recovered from a small number of linear measurement $b = A\mathbf{x} \in R^K$, K<< N by solving a convex program. l_1 minimization is used to solve the under determined linear equations or sparsely corrupted solution to an over determined equations. Recently, l_1 minimization has been proposed as a convex alternative to the combinatorial norm l_1, which simply counts the number of nonzero entries in a vector, for synthesizing the signal as a sparse superposition of waveforms.

(a) Min-l_1 with Equality Constraints

The program $P1 \min \|\mathbf{x}\|_1$ subject to $A\mathbf{x} = b$, is also known as *basis pursuit.* The goal of this program is to find the smallest l_1 norm, using the following equation (2.6)

$$\|\mathbf{x}\|_1 := \sum_i |\mathbf{x}_i|$$

This algorithm search is for a vector **x**, that will explain the observation b. If the signal **x** is sufficiently sparse then (P1) will find the norm of **x** by using the values of *A* and *b* [8, 9]. When **x**, *A*, and *b* have real valued entries, (P1) can be recast as a linear program.

(b) Minimum l_1 Error Approximation

Let A be a full rank $M \times N$ matrix. Given that $\mathbf{y} \in R^M$ then where P_A is the basis

$$(P_A) \min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 \qquad (2.12)$$

pursuit which will find the vector $\mathbf{x} \in R^N$ in such a way that the error $(\mathbf{y} - A\mathbf{x})$ has the minimum l_1 norm [8, 9]. As an example, let us apply this algorithm to the field of channel coding. Channel coding produces a code word $c = A\mathbf{x}$ that represents the message **x**. The message is transmitted over a channel from the transmitter to the receiver. During this process the signal is corrupted by external sources. The decoders detect the signal as $\mathbf{y} = c + e$, where e is the corruption. If this error is sparse enough then this program can be used to reconstruct the signal **x** perfectly and eliminate the interference introduced by the external source.

2.3.2 l_2 Minimization

It defines the l_p norm of the vector **s** as $(\|\mathbf{s}\|_p)^p = \sum_{i=1}^{N} |\mathbf{s}_i|^p$. The classical approach to inverse problems of this type is to find the vector in the translated null space with the smallest l2 norm (energy) by solving

$$\mathbf{s}^* = \text{argmin } \|\mathbf{s}`\|_2 \text{ such that } \boldsymbol{\theta}\mathbf{s}` = \mathbf{y} \tag{2.13}$$

This optimization technique has the closed form solution $\mathbf{s}^* = \boldsymbol{\theta}^{\mathbf{T}}(\boldsymbol{\theta}\ \boldsymbol{\theta}^{\mathbf{T}})^{-1}\ \mathbf{y}$. Unfortunately, l_2 minimization will almost never find a K-sparse solution, returning instead a nonsparse **s*** with many nonzero elements.

2.3.3 Convex Program

cvx is a *disciplined convex programming (DCP) algorithm*. *Disciplined convex programming* is a set of conventions used for constructing convex optimization proposed by Michael Grant, Stephen Boyd, and Yinyu Ye [10]. There are convex optimization problems that are used to analyze and solve problems such as Linear programs (LPs), Second Order Cone Programs (SOCPs) and Semi-Definite Program (SDPs). They can also be used to solve other problems which involve non-differentiable functions, such as the l_1 norm [10]. These set of rules are called the *DCP rule set* and problems which disobey these set of rules are rejected even if the problem is convex. The problem which cannot be solved using DCP has to be rewritten in such a way that it follows the DCP rule set. First, let us consider the most basic convex optimization problem, “least squares”. In a least-squares problem, $\mathbf{x} \in R^n$ is the vector that minimizes $\| A\mathbf{x} - b \|_2$, where $A \in R^{m\times n}$ is skinny and full rank (i.e. $m \geq n$ and Rank (A) = n). Let us create a test problem data for *m*, *n*, *A* and *b* in MATLAB.

```
m=16; n=8;
A=randn (m, n);
 b=randn (m, 1);
```

Then the least square solution for *x* is given by $x = (A^T A)^{-1} A^T b$. The easiest way to calculate the least square is by using the back slash operator which is given by *x*= *A**b*. The same result is obtained in MATLAB using *cvx*.

```
cvx_begin
      variable x (n);
      minimize (norm (A*x-b));
 cvx_end                                                                    (2.13)
```

The first line of code creates a place holder for the new cvx specification and prepares MATLAB to accept the declaration of variables, constraints and objective functions. The *variable x(n)* declares **x** as a optimization variable. The variable must be declared before it is been used as a constraint or as an objective function. The next line of code specifies that the objective function is to be minimized; in this case it is l_1 norm of $A\mathbf{x} - b$. The *cvx_end* specifies the end of the *cvx* specification and cause the problem to be solved.

2.3.4 Matching Pursuit

Orthogonal matching pursuit (OMP) is a canonical greedy algorithm for a sparse approximation. Let $\boldsymbol{\phi}$ represent a matrix of size $M \times N$ (where typically $M < N$) and $\mathbf{y}$ denotes a vector in R^M, the goal of OMP is to recover a coefficient vector **x**`∈ R^N with roughly $K < M$ non-zero terms so that $\boldsymbol{\phi}\hat{\mathbf{x}}$ equals **y** exactly or approximately. OMP is frequently used to find a sparse representations of the signal $\mathbf{y} \in R^M$ in settings where $\boldsymbol{\phi}$ represents a dictionary for the signal space. It is also commonly used in compressive sensing, where $\mathbf{y} = \boldsymbol{\phi}\mathbf{x}$ represents compressive measurements of a sparse signal $\mathbf{x} \in R^N$ to be recovered. One of the attractive features of OMP is its simplicity. OMP is empirically competitive in terms of approximation performance.

2.4 Applications of Compressive Sensing

2.4.1 Compressive Imaging

The common approach in digital image system is to capture as many pixels as possible and later compress the captured image by digital means [13]. Compression is desired to increase the storage capacity and enhance the communication process. Compression techniques exploit the visual redundancy typical to human intelligible images. After capturing the optical image and applying data compression, the image is represented by a smaller number of pixels than the original image. The decompressed image should satisfy some desired visual quality. This type of imaging evokes a question: is it necessary to capture all the image samples and then compress them? The answer to the question is the recent development in compressive sensing theory. The basic idea for implementing compressive sensing is that the reconstruction of an image is possible even with fewer measurements than the nominal number of pixels.

2.4.2 Medical Imaging

One of the most promising applications for compressive sensing is in medical imaging. MRI scanners sample the lines within the “k-space”. Sampling each line take time and inject energy into the patient [14]. As MRI technology has advanced, there has been an increasing desire to use higher field strengths to analyze larger sets of data, such as 3D imaging and dynamic imaging. A number of reduced sampling techniques have been used such as K-t BLAST, K-t SENSE and K-t vDUST which exploits the linear spatiotemporal correlation within the image sequences. In contrast, Lusting et al [14]. have produced preliminary results showing that a compressive sensing strategy that exploits the sparsity of the spatiotemporal data is also possible.

2.4.3 Analog to Information Conversion

Analog to-digital converters (ADC) have been used in sensing and communications due to the advancement in digital signal processing. The process of ADC is based on the Nyquist sampling theorem which uniformly samples the signal with a rate of at least twice its bandwidth in order to reconstruct the signal perfectly. Emerging applications like radar detection and ultra-wideband communication are pushing the limit of ADC. The recent developments in the field of compressive sensing (CS), has helped in the design of Analog to-Information converters (AIC) that are able to acquire samples at a lower sampling rate.

2.4.4 Compressive Radar

The new theory of compressive sensing can be used in radar imaging systems which are designed to determine the range, altitude, direction and speed of moving and fixed objects [6]. The received radar signal can be reconstructed with fewer measurements by solving an inverse problem through a linear program or a Greedy Algorithm. With the implementation of compressive sensing in radar systems, the need for a pulse compression matched filter at the receiver side and the analog to-digital conversion operating at high Nyquist rates can be eliminated. As a result, the complexity and the cost of the receiver hardware is going to be greatly reduced.

2.5 Compressive Sensing in Mobile Communication System

The purpose of this research effort is to implement compressive sensing in a mobile system. By using compressive sensing techniques, the speech signal is precoded at the transmitter side which is being sent to the receiver through a wireless channel. As a result, a small number of samples are being transmitted, and this will increase the transmission data rates when compared to the current communication systems. In the proposed communication system, first the speech signal is modeled in such a way that the input signal is sparse enough before applying compressive sensing. The sparse signal is multiplied by the predefined measurement matrix. The output of the compressive sensing module is then transmitted to the receiver. At the receiver, the signal is perfectly reconstructed from a significant small number of measurements by using different optimization techniques such as l_1-norm or convex optimization. The basic block diagram of the proposed systems is shown in Figure 4.

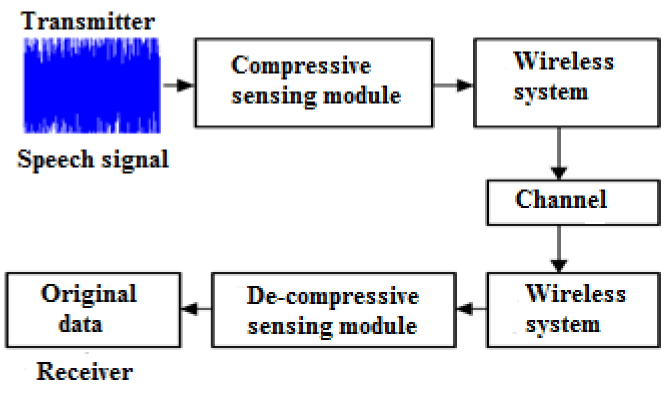


Figure 4. Compressive sensing in a mobile communication system

Chapter Three

Analysis and Design

In this chapter, the use of compressive sensing in a mobile communication system is proposed in order to increase the data rates. The objective is to increase the data rates of current and possibly future generation mobile systems. In the proposed system the speech signal is sampled below the Nyquist rate by using compressive sensing. The compressed spectrum is then transmitted over the wireless system and successfully reconstructed at the receiver without losing any significant information. In the first stage of the project, a speech signal was modeled using a Laplace random number generator shown in Figure 5. It was decided to use a Laplace number generator to model the speech signal, because these types of signals typically have a Laplacian distribution [15]. The modeled speech signal was mapped into the discrete frequency domain using the FFT. The results obtained from this transformation are shown in Figure 6. In the second stage, before compressive sensing was applied to the signal, a threshold window was used to eliminate the coefficients that are significant to the signal. In other words, all the coefficients with small amplitude were multiplied by zero. In Figure 7 it can be seen how the FFT spectrum looks after the threshold has been applied. The purpose of the threshold is to ensure that the FFT spectrum is sparse.

In the third stage, the threshold spectrum was multiplied by the measurement matrix, which is a matrix composed of random numbers. The output of the compressive sensing algorithm is converted into a digital signal using an Analog- to-Digital converter in order to be transmitted by the mobile system. At the receiver section, an initial guess was made using the measurement matrix and the observation vector (vector signal), which is close to the input speech signal. Finally, the speech signal was reconstructed from a significant small number of observations by using one of the optimization techniques available. The reconstructed signal at the output of the optimization module is shown in Figure 8. The difference between the actual signal and the reconstructed signal was calculated in order to observe the error between both signals. This error is shown in Figure 9.

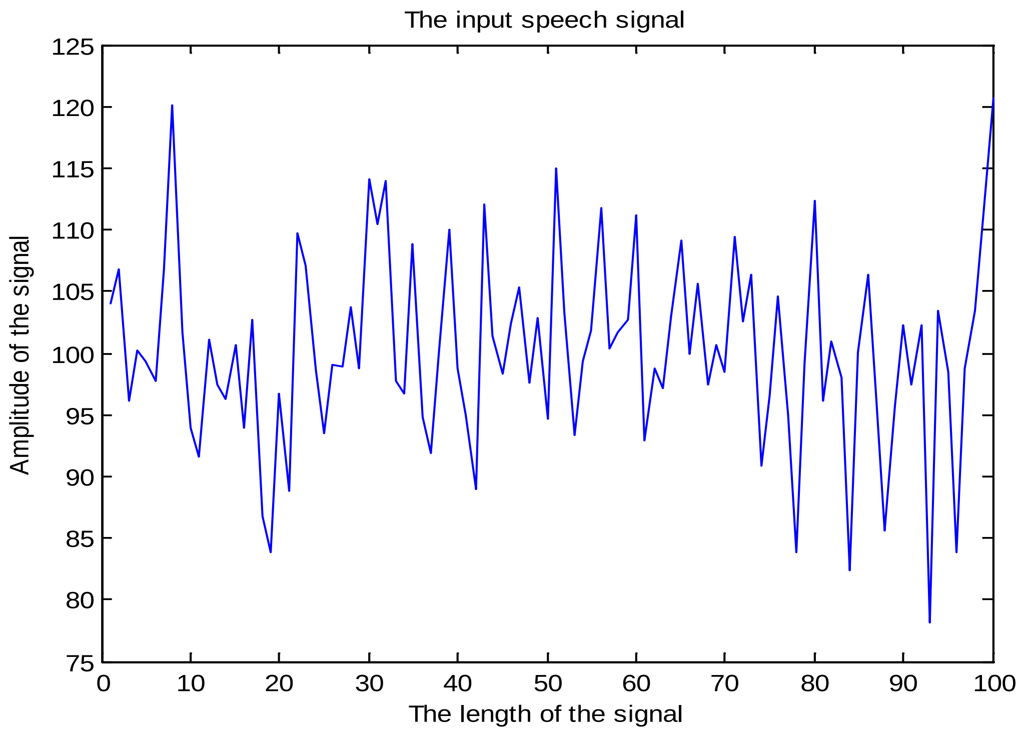


Figure 5. Input speech signal

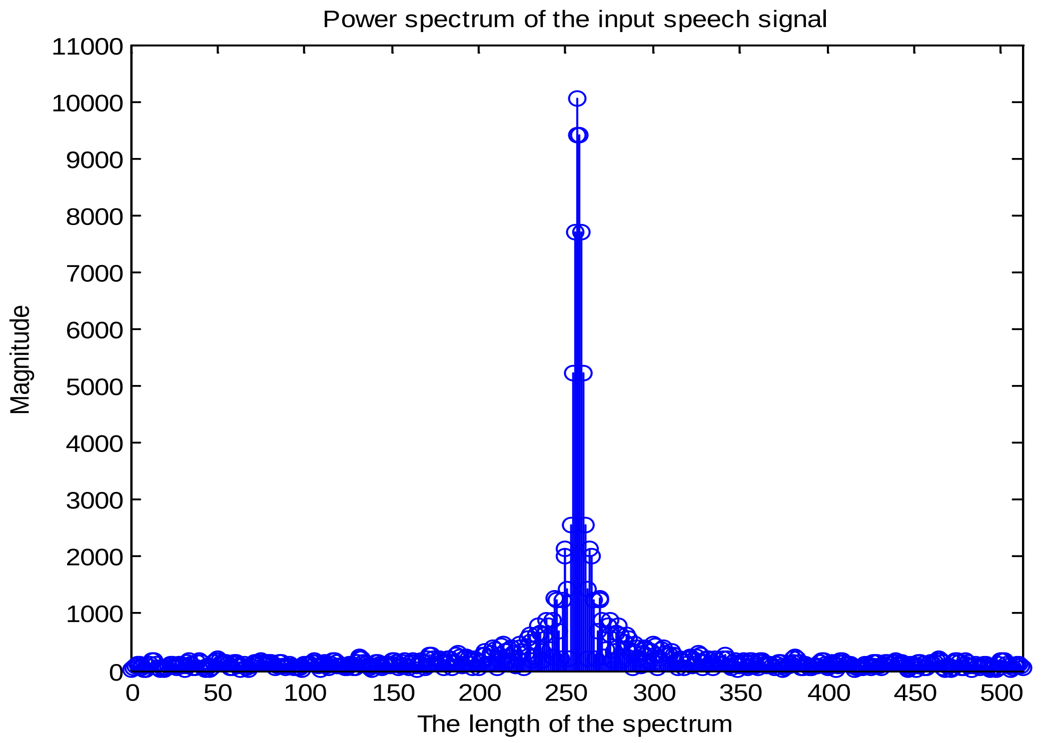


Figure 6. Power spectrum of the input speech signal

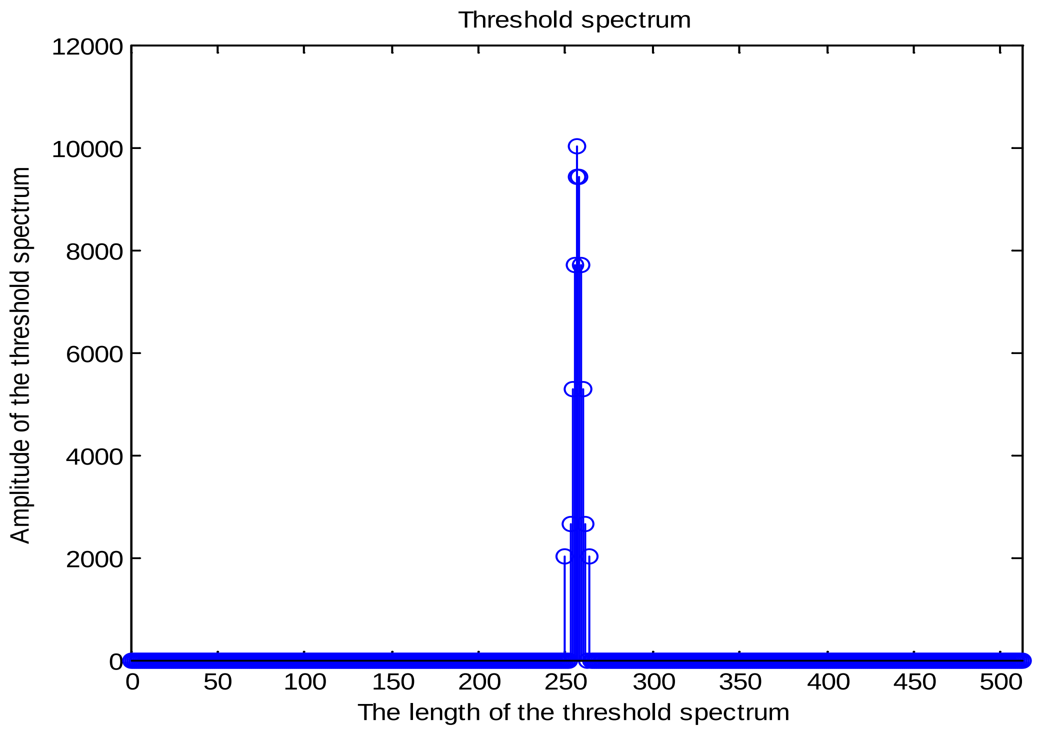


Figure 7. Threshold spectrum of FFT spectrum

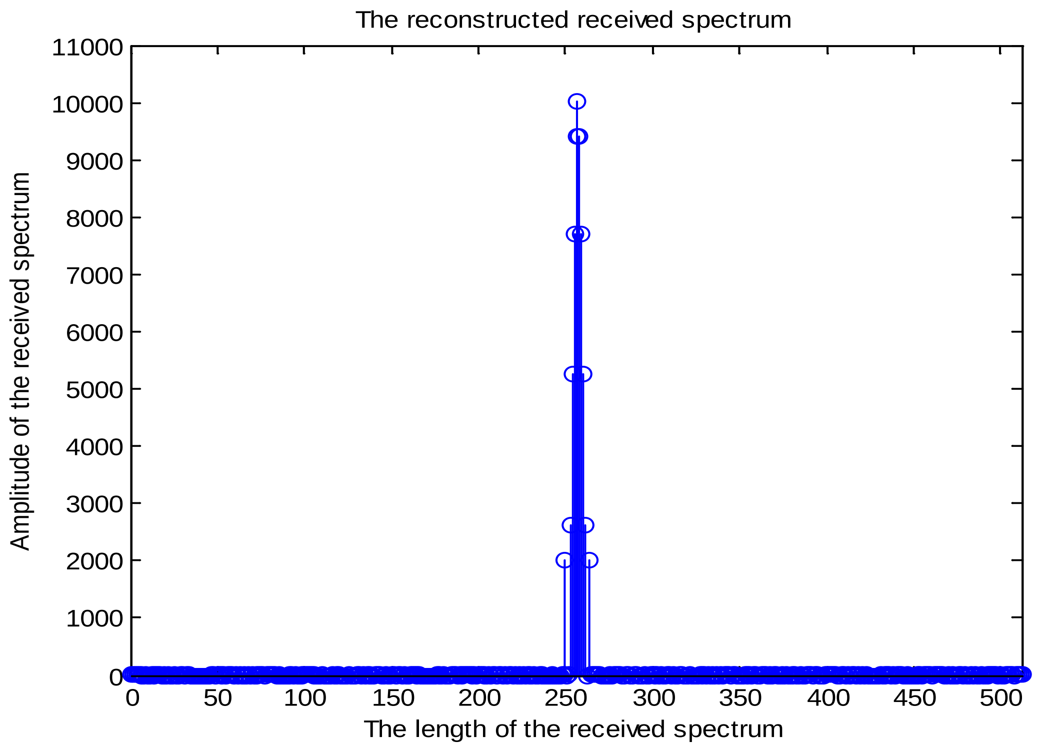


Figure 8. Reconstructed FFT spectrum using l_1-minimization

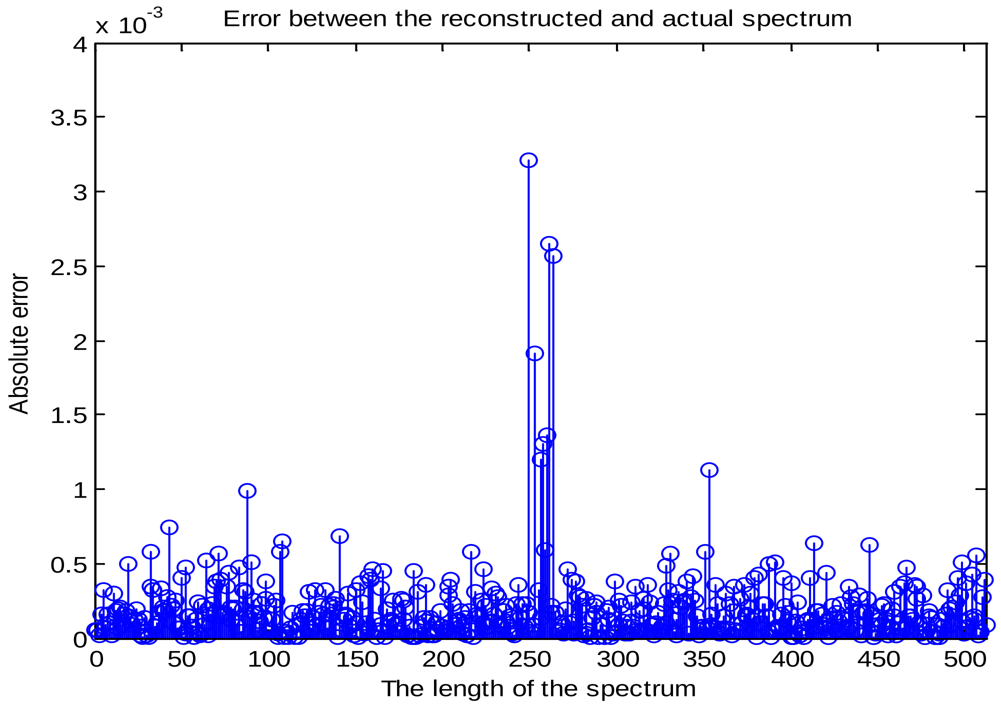


Figure 9. Error between the reconstructed and the actual spectrum

The FFT was a revolutionary algorithm that made Fourier analysis and processing of digital signals fairly easy. The FFT algorithm employs a few tricks and can compute the Fourier transform of a discrete signal in $N\log_2(N)$ operations [16]. There are two factors that need to be considered when the FFT is implemented in MATLAB

(i) FFT uses complexes numbers and

(ii) It computes both positive and negative frequencies.

This is where it becomes difficult to implement FFT in compressive sensing. The main problem is that applying compressive sensing to a complex number is a tedious and complex process, and is being researched at Michigan State University using a hybrid compressive sensing model (Complex and Real) [17].

In order to overcome caused by the FFT, instead a Discrete Cosine Transform (DCT) is implemented. The DCT is conceptually the same as DFT except that it does a better job in concentrating the energy into lower order coefficients than the DFT. Another important property of the DCT is that all the spectral coefficients are purely real. Assuming that the input signal is periodic, the magnitude of the DFT is spatially invariant (phase of the input does not matter) which is not true for DCT. The DCT does not introduce discontinuities while imposing periodicity into the time signal, whereas in the DFT, the time signal is truncated and assumed to be periodic. As a result, discontinuities are introduced in the time domain and corresponding artifacts are introduced into the frequency domain. However, as an even symmetry is assumed while truncating the time signal, no discontinuities or related artifacts are introduced in the DCT. The comparison between the FFT and the DCT is shown in Figure 10 [18].

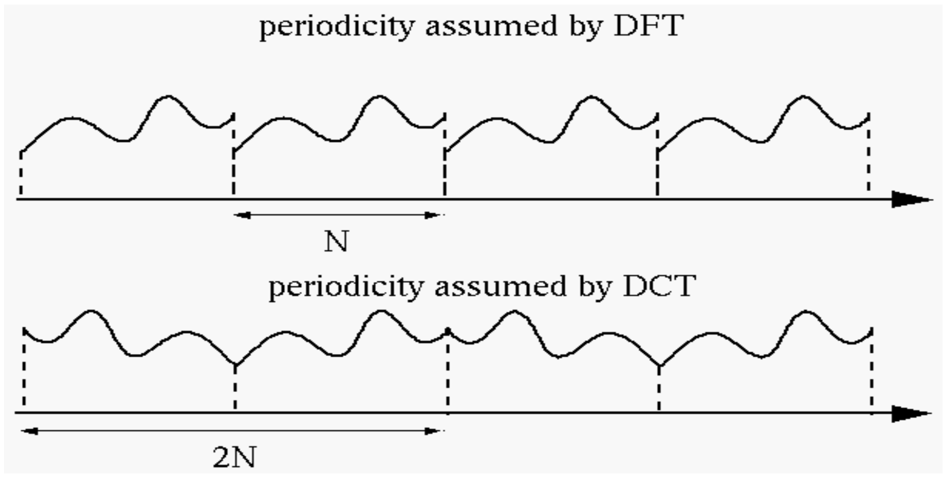


Figure 10. Periodicity comparison of DFT and DCT [18]

Chapter Four

Results

In order to test the algorithm on a real speech signal, a wave recorder was used to record a short piece of speech. The MATLAB function "*wavrecord*" allows the user to record *n* samples of an audio signal at a specific sample rate. For instance, Figure 11 shows a speech signal recorded using the "*wavrecord*" function composed of 2000 samples. The recorded speech signal then goes through the DCT which transforms a sequence of real data points into its real spectrum. The transformed speech signal is shown in Figure 12. Before compressive sensing is applied to the DCT spectrum a threshold window is used to eliminate the small coefficients. The rationale under this process is that the small coefficient does not contribute to the overall signal compared to the large coefficient. This is used to ensure that the DCT spectrum is sparse before applying compressive sensing. The result from this process is shown in Figure 13.The threshold spectrum is then multiplied by the measurement matrix which in this case is composed of randomly generated numbers which is shown in Figure 14.

The output of compressive sensing is the observation vector which is sent to the mobile communication module in order to be transmitted which is shown in Figure 15. At the receiver the compressed DCT coefficient is decompressed and reconstructed from a significant small numbers of observations using one of the different optimization techniques such as the l_1 normalization shown in Figure 16. The error is calculated by taking the difference between the received and the transmitted threshold spectrum, which is shown in Figure 17. Finally, the received reconstructed spectrum is passed through the IDCT in order to recover the speech signal. The recovered speech signal can be observed in Figure 18.

During the design of the proposed system multiple tests were performed in order to analyze the efficiency of this system. Table 1 shows how the length of the signal, the threshold value and the sparsity of the signal affect the compression rates. It is also observed that by keeping the length of the signal (L) constant and by varying the

Threshold window (Th) it is plausible to achieve a desired compression. This is because by modifying the threshold window the compressed sample (K) is also modified.

Table 1. Amount of compression by varying signal parameter.

Length of Signal (L)	*Threshold window (Th)* *UL*	*LL*	*Compressed samples (K)*	*Error (err)*	*Compression (%) (K/L)*
2000	0.04	-0.06	1000	1.5×10^{-5}	50
2000	0.04	-0.06	900	1.8×10^{-5}	55
2000	0.04	-0.06	800	7×10^{-5}	60
2000	0.04	-0.06	700	0.7	65
2000	0.04	-0.06	600	0.18	70
2000	0.04	-0.06	500	0. 3	75

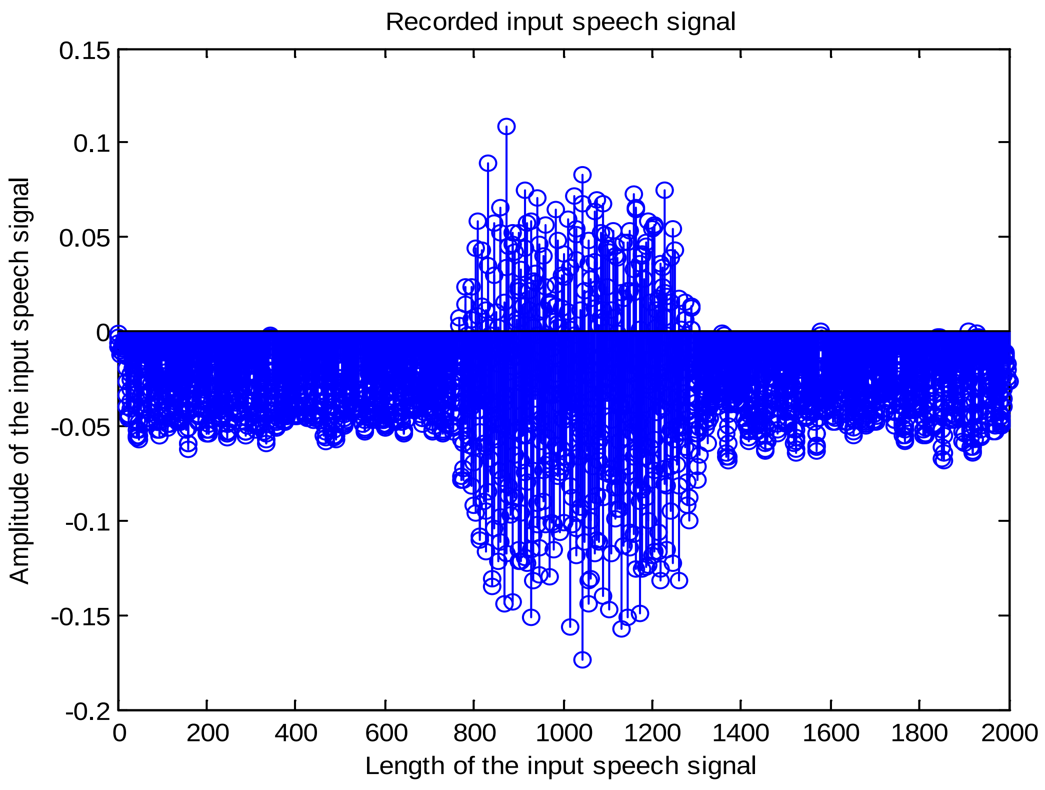


Figure 11. Recorded input speech signal

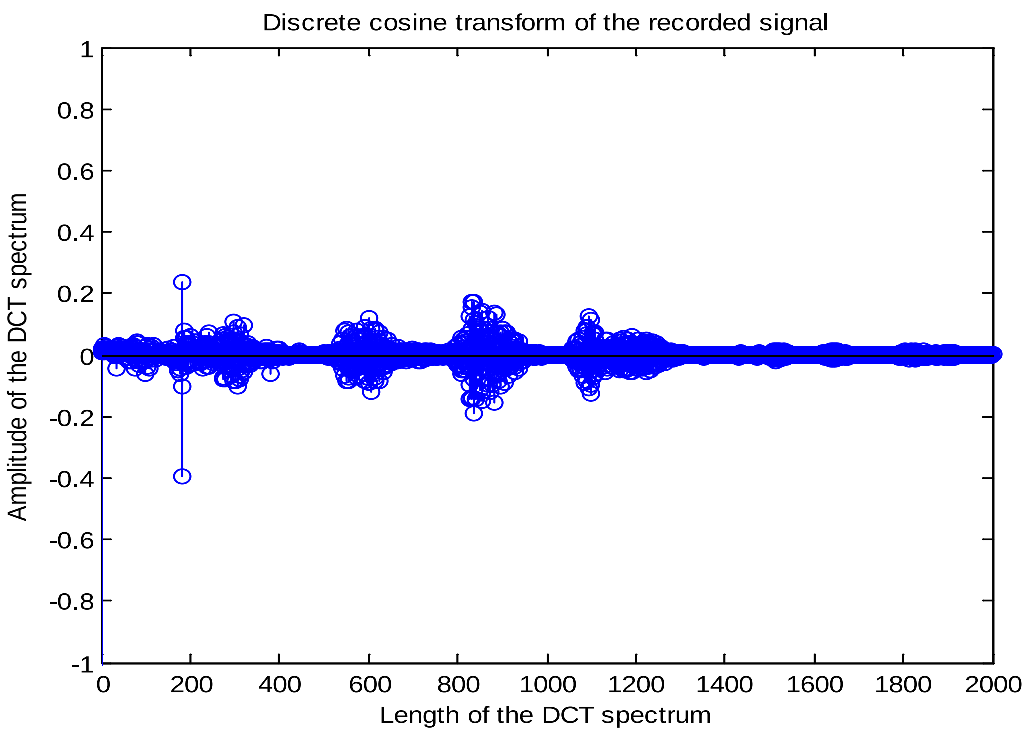


Figure 12. DCT of recorded speech signal

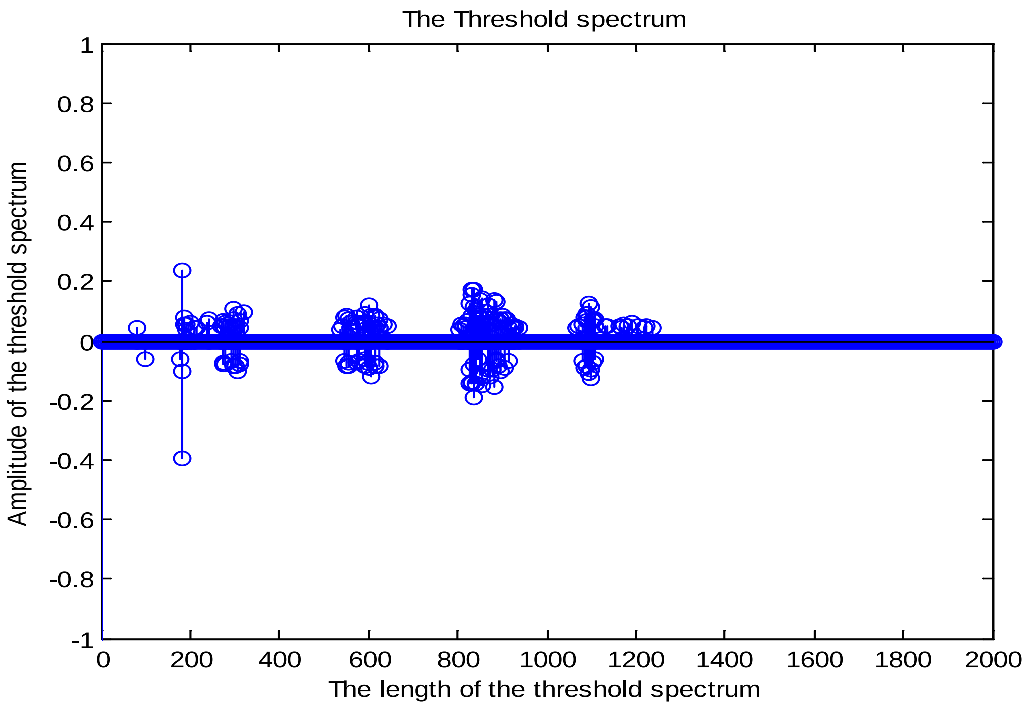


Figure 13. Threshold spectrum

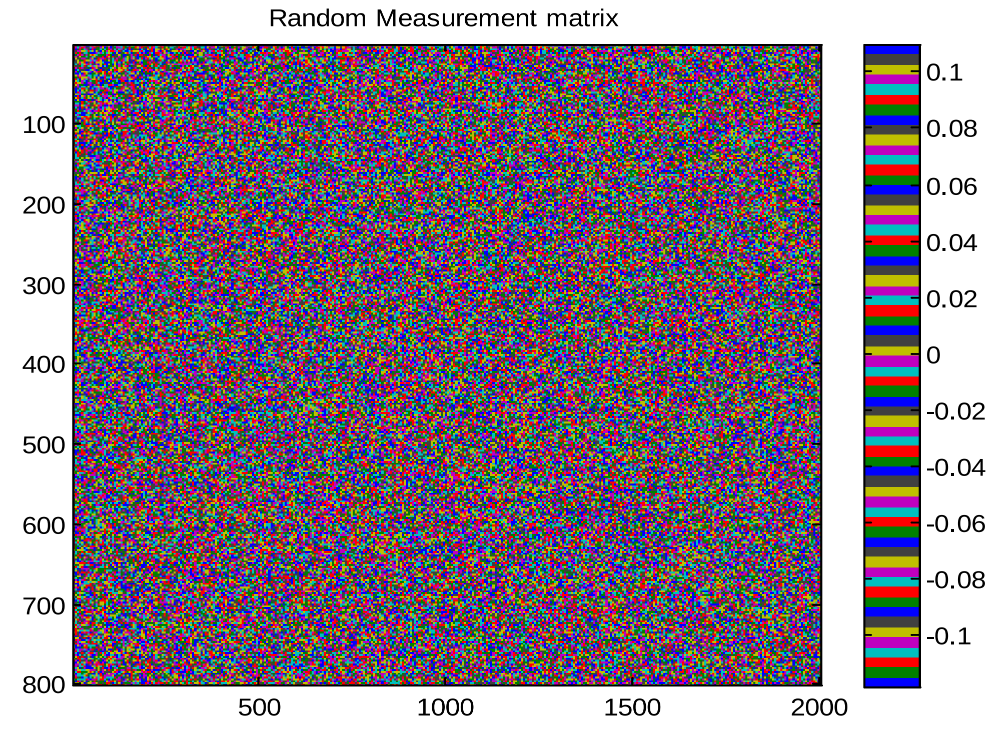


Figure 14. Random measurement matrix

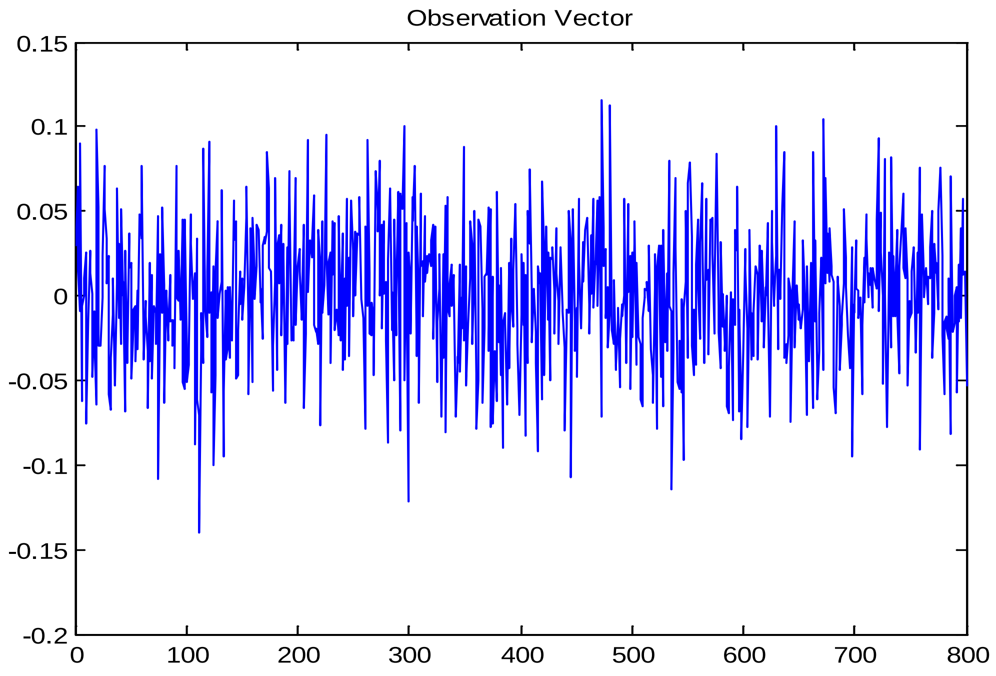


Figure 15. Observation vector

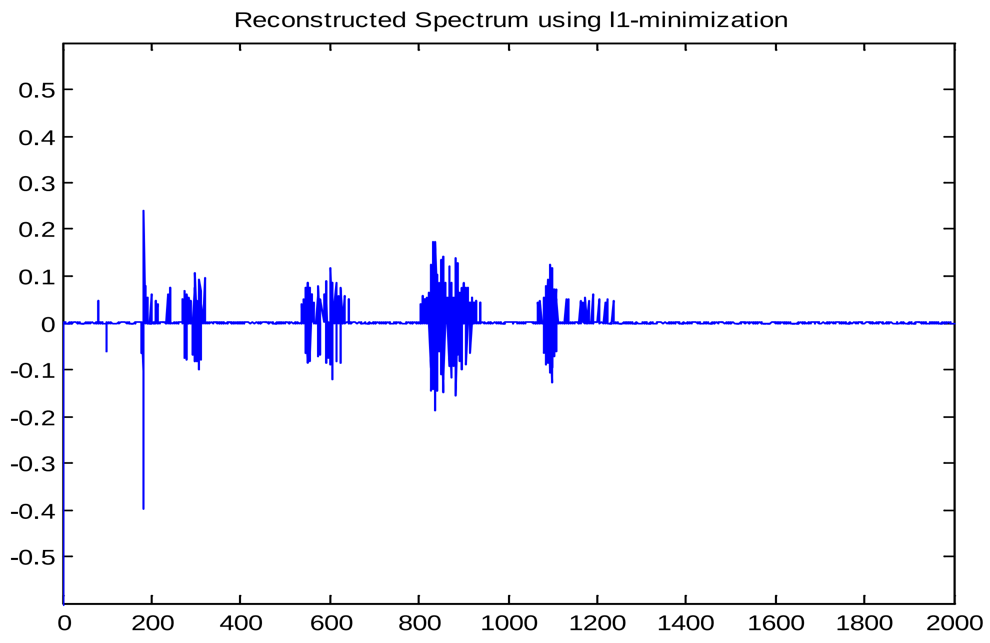


Figure 16. Reconstructed spectrum using l_1 minimization

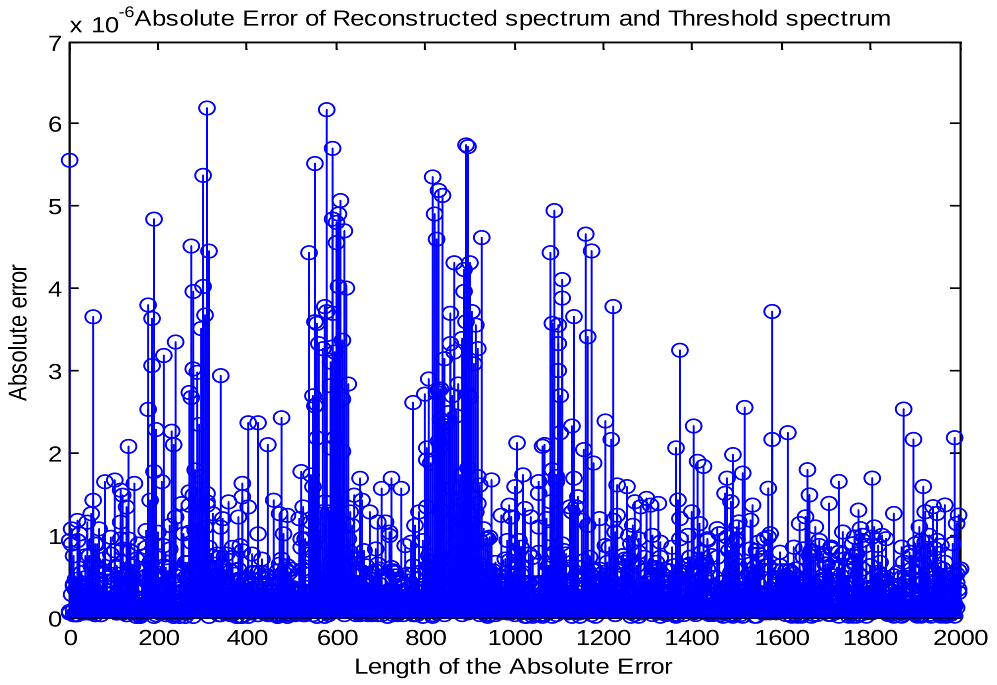


Figure 17. Absolute error between the received reconstructed spectrum and original spectrum

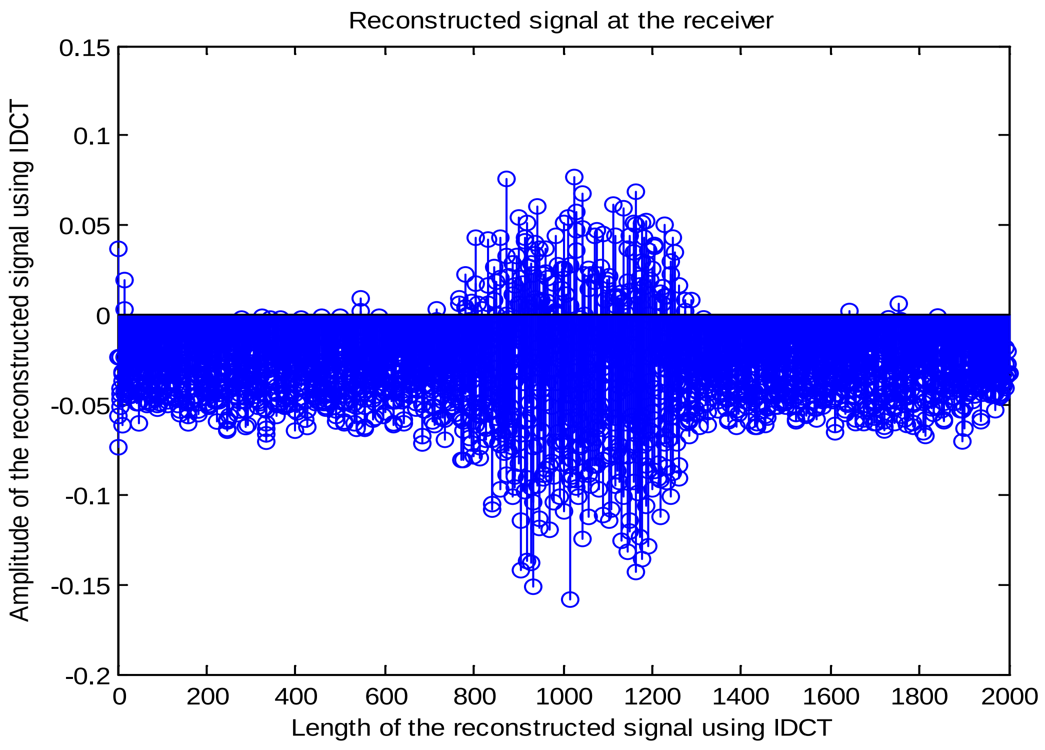


Figure 18. Reconstructed speech signal at the receiver

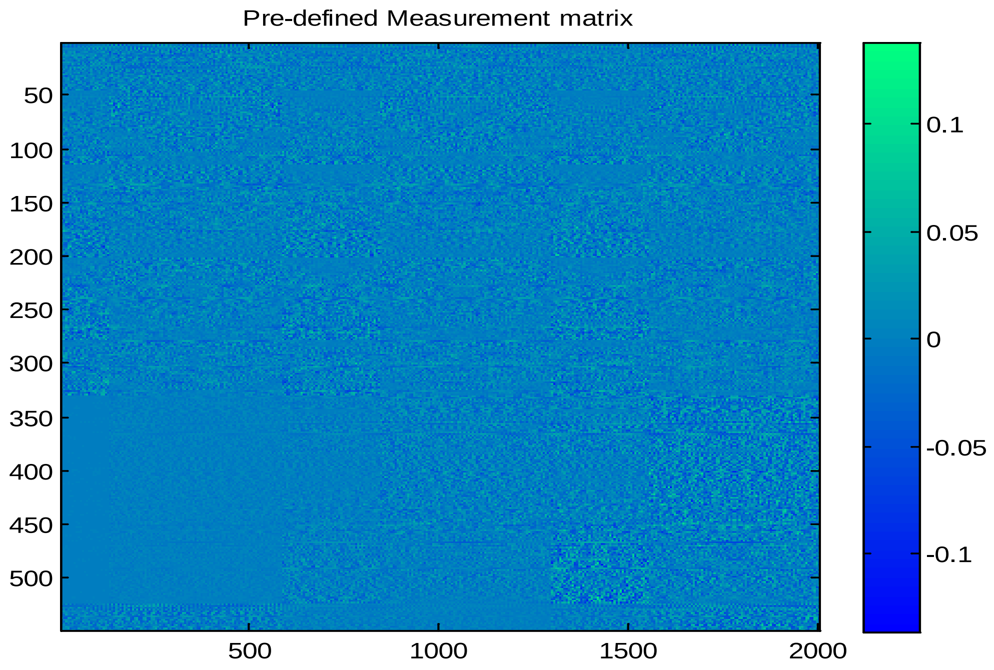


Figure 19. Pre-defined measurement matrix

4.1 Comparison with Wavelet Transform

The Wavelet transform is a powerful mathematical tool in many areas of science and engineering, in particular in the fields of audio and data compression. A wavelet is defined as a small wave that has its energy concentrated in time to provide a tool for the analysis of transient, non-stationary, or time-varying phenomena. A signal or function *f(t)* often can be better analyzed, described, or processed if it is expressed as a linear decomposition

$$f(t) = \sum_{l} a_l \psi_l(t) \tag{4.1}$$

where l is an integer index for the sum, a_l is the expansion coefficients and Ψ_l is the set of real-valued functions of t called the expansion set. If the expansion is unique, the set is called a basis for the functions that could be represented. If the basis is orthogonal, then the coefficients can be calculated by the *inner product.*

$$a_k = \langle f(t), \psi_k(t) \rangle = \int f(t)\psi_k(t)dt \tag{4.2}$$

A single coefficient a_k is obtained by substituting equation (4.1) into equation (4.2) and therefore for the *wavelet expansion,* a two-parameter system is constructed such that equation (4.1) becomes

$$f(t) = \sum_{k}\sum_{j} a_{j,k}\psi_{j,k}(t) \tag{4.3}$$

Where both j and k are integer indices and $\Psi_{j,k}$ *(t)* is the wavelet expansion that usually forms an orthogonal basis. The set of expansion coefficients $\boldsymbol{a}_{\boldsymbol{j},\,\boldsymbol{k}}$ are called the discrete wavelet transform of *f (t)* and equation 4.3 is its inverse. All wavelet systems are generated from a single scaling function or wavelet by simple scaling and translation. This two-dimensional representation is achieved from the function Ψ *(t),* also called the mother wavelet

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \qquad j,k \in Z \tag{4.4}$$

Wavelet systems also satisfy multi-resolution conditions. In effect, this means that a set of scaling functions can be determined in terms of integer translates of the basic scaling function by

$$\varphi_k(t) = \varphi(t-k) \quad k \in Z \quad \varphi \in L^2 \tag{4.5}$$

It can be seen that if a set of signals can be represented by *φ(t-k)*; a larger set can represented by *φ(2t-k)*, giving a better approximation of any signal. Hence, due to the spanning of the space of *φ(2t)* by *φ(t)*, *φ(t)* can be expressed in terms of the weighted sum of the shifted *φ(2t)* as

$$\varphi(t) = \sum_n h(n)\sqrt{2}\varphi(2t-n), \quad n \in Z \tag{4.6}$$

Where the coefficients *h(n)* may be real or complex numbers called the scaling function coefficients. However, the important features of a signal can better be described, not by $\boldsymbol{\varphi_{j,k}(t)}$, but by defining a slightly different set of functions $\Psi_{j,k}(t)$ that span the differences between the spaces spanned by the various scales of the scaling function. These functions are the wavelets and, they can be represented by a weighted sum of shifted scaling *φ(2t)* function defined in equation 4.6 by

$$\psi(t) = \sum_n h_1(n)\sqrt{2}\varphi(2t-n), \quad n \in Z \tag{4.7}$$

The function generated by equation 4.7 gives the prototype or mother wavelet *Ψ(t)* for a class of expansion functions of the form given by equation 4.4

$$f(t) = \sum_{k=-\infty}^{\infty} c(k)\varphi_k(t) + \sum_{j=0}^{\infty}\sum_{k=-\infty}^{\infty} d(j,k)\psi_{j,k}(t) \tag{4.8}$$

The coefficients in this wavelet expansion are called the discrete wavelet transform (DWT), of the signal ***f(t)***. For a large class of signals, the wavelet expansion coefficients drop off rapidly as *j* and *k* increase. As a result, the DWT is efficient for image and audio compression.

The demand for compression technology increases every year in parallel with the increase in aggregate bandwidth for the transmission of audio and video signals. As a result, the wavelet-based approach plays an important role in the scheme of things as Perceptual coding of audio signals found its way to a growing number of consumer applications. The foremost criterion for audio compression technology is to achieve a certain signal quality at a given bit-rate as this directly translates to cost savings by getting a higher compression ratio at the same quality of service. Wavelet-based compression is claimed to be more efficient at low bit rates but are actually less successful than discrete cosine transform (DCT) -based systems in achieving good efficiency at near-transparent compression ratios. Computational complexity also limits the algorithmic implementation of a codec. As a result, algorithmic delay becomes an important constraint especially for two-way communications applications. In that respect, it is notable that Wavelet compression does require more computational power than DCT-based compression.

Let us consider a test signal which has to compress using compressive sensing and wavelets transformation in order to evaluate the performance. Figure 20 is the test signal which has to be compressed using wavelet transform.

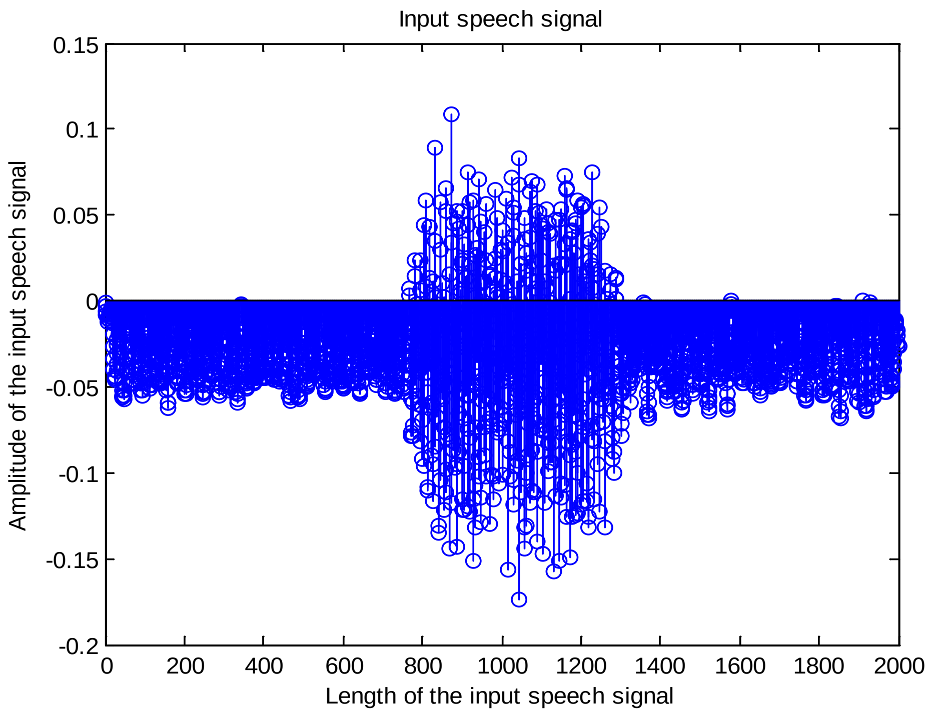
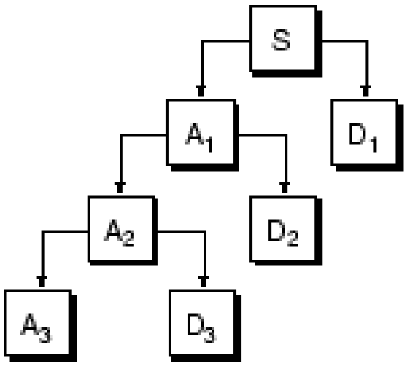


Figure 20. Input test signal

Figure 22 depicts the 1-level decomposition of the test signal using a wavelet transform. This process involves two aspects: breaking up a signal to obtain the wavelet coefficients, and reassembling the signal from the coefficients. In wavelet analysis, a signal is split into an approximation and a detail. The approximation is then itself split into a second-level approximation and detail, and the process is repeated. For an n-level decomposition, there are n+1 possible ways to decompose or encode the signal.



$$
\begin{aligned}
S &= A_1 + D_1 \\
&= A_2 + D_2 + D_1 \\
&= A_3 + D_3 + D_2 + D_1
\end{aligned}
$$

Figure 21. Approximations and details of original signal

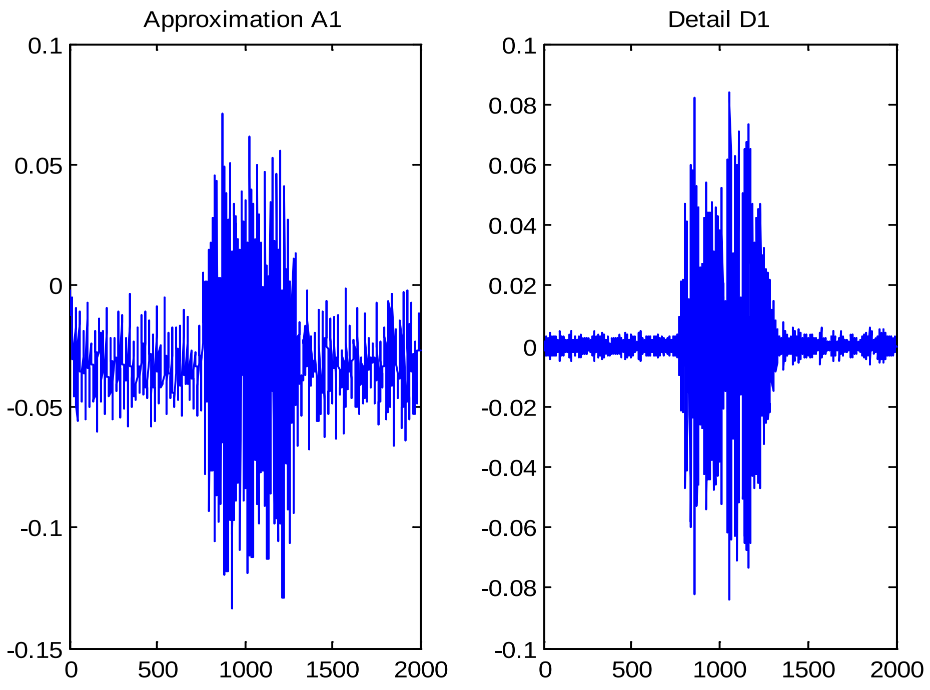


Figure 22. 1-Level decomposition of the test signal

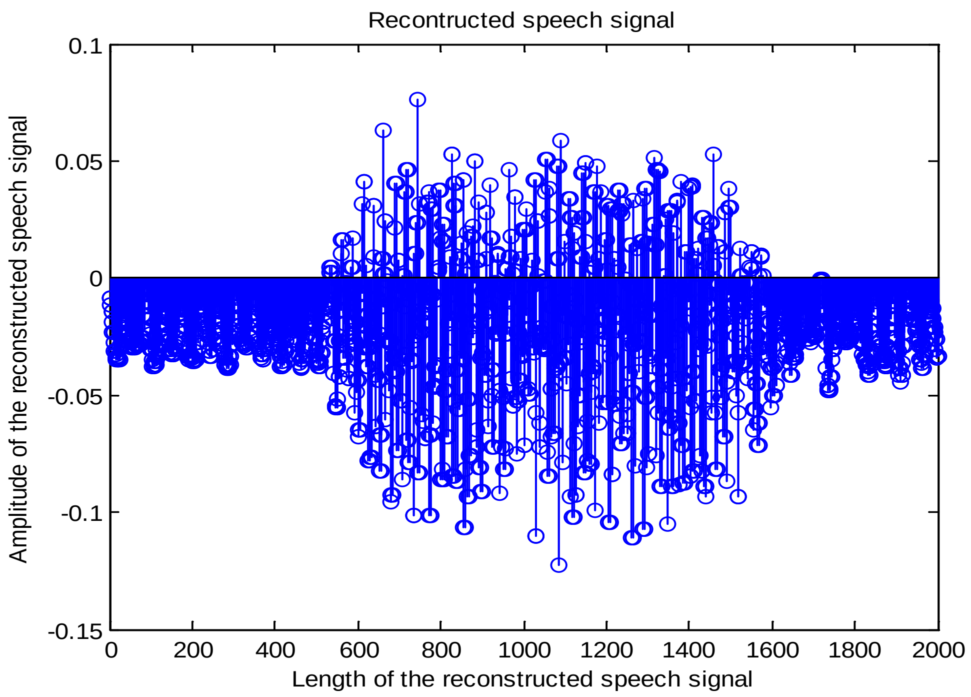


Figure 23. Reconstructed signal using wavelets

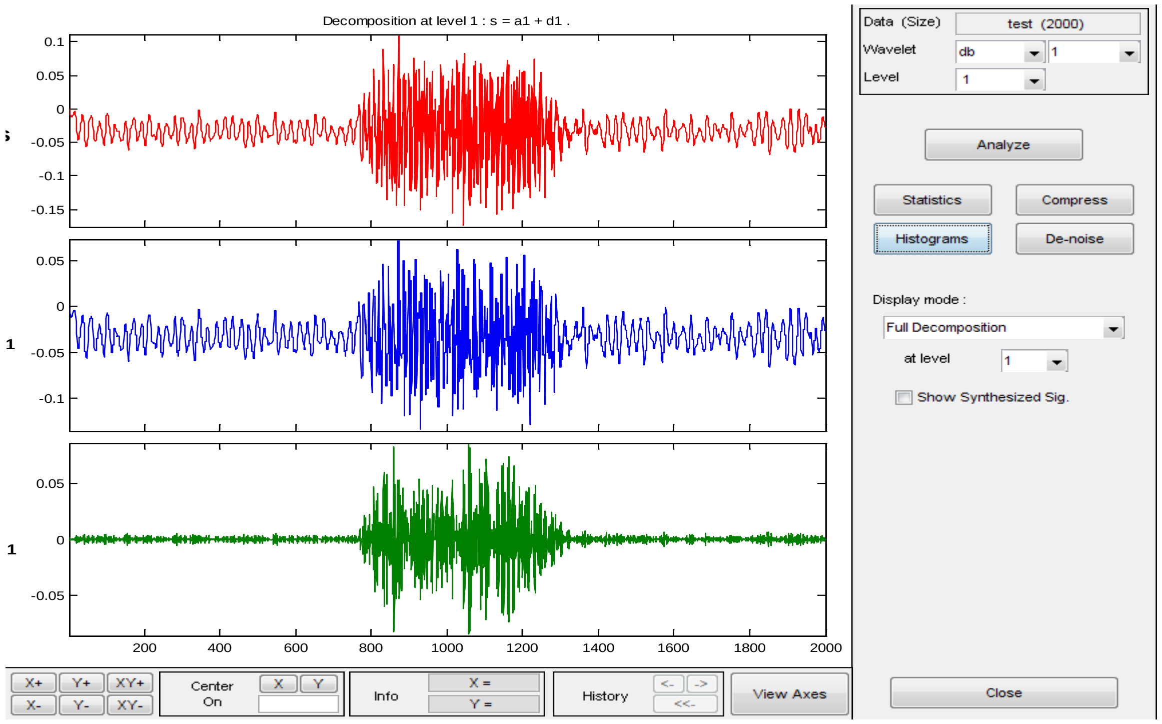


Figure 24. Test signal with 1-level decomposition using wavelets GUI toolbox

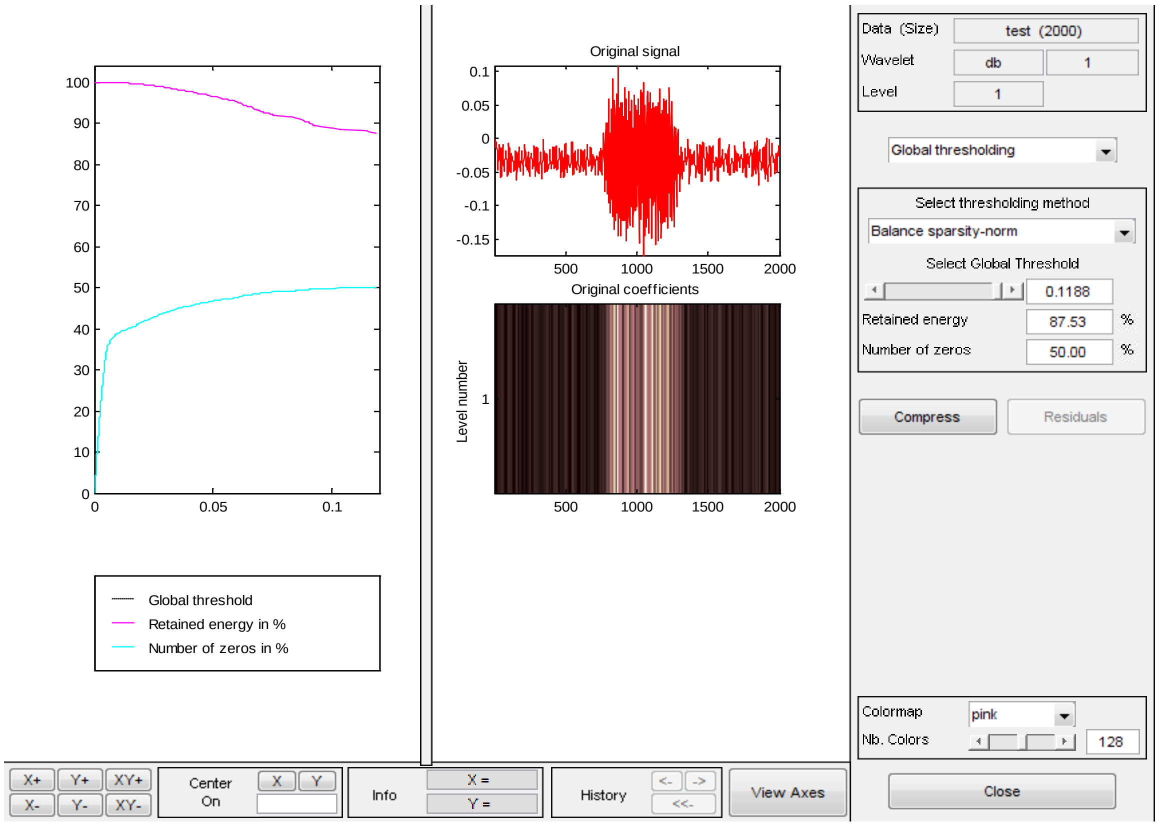

Data (Size): test (2000)
Wavelet: db 1
Level: 1
Global thresholding
Select thresholding method: Balance sparsity-norm
Select Global Threshold: 0.1188
Retained energy: 87.53 %
Number of zeros: 50.00 %
Colormap: pink
Nb. Colors: 128

Figure 25. Test signal with 1-level of coefficients

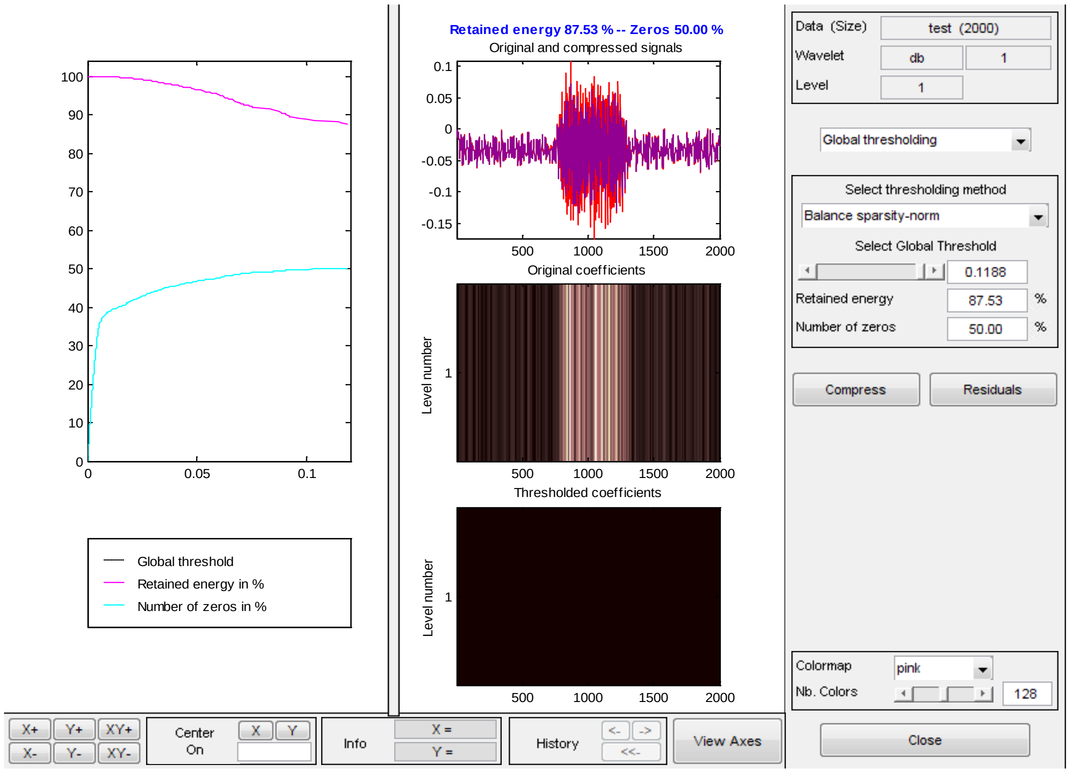

Retained energy 87.53 % -- Zeros 50.00 %
Data (Size): test (2000)
Wavelet: db 1
Level: 1
Global thresholding
Select thresholding method: Balance sparsity-norm
Select Global Threshold: 0.1188
Retained energy: 87.53 %
Number of zeros: 50.00 %
Colormap: pink
Nb. Colors: 128

Figure 26. Original and compressed signal using wavelets GUI toolbox

Table 2. shows the comparison of compression of speech signal using wavelets and compressive sensing using different parameters.

Table 2. Comparison of compressive sensing and wavelet compression

Parameters	Length of signal	Threshold window	Number of zeroes	Non-zeroes coefficient	Maximum absolute Error	Compression (%)
Compressive sensing	2000	0.0074 to -0.0045	1000	1000	0.0160	50%
Wavelet	2000	0.1188	1000	1000	0.1840	50%

Chapter Five

Conclusion and Future Work

In this research the design of a new mobile communication system using compressive sensing has been studied. The proposed system should fulfill the following specifications:

- Low power consumption
- Accurate reconstruction of the speech signal
- Increased data rates

During the design process, this module went through different tests and analysis in order to find the most adequate optimization technique to reconstruct the speech signal with few random measurements without losing the information. For simulation purposes, code was created in order to compress and transmit the speech signal below the Nyquist rate by taking only a few measurements of the signal. The result shows that by keeping the length of the signal (L) and threshold window (Th) constant one can achieve the desired compression of the signal by making the signal sparse (K) to a certain amount which in turn increases the data rates. Two different types of measurement matrices: predefined and random measurement matrices were studied and tested using MATLAB. The speech signal was reconstructed without losing important information in order to achieve an increase in the data rates. After multiple simulations, it was found that the system worked as expected and the speech signal was reconstructed efficiently with a minimum error. However, the system is still not perfect and more research still required. Performance of compressive sensing is better when compared to wavelet compression as there is a minimum error with same compression rate using different parameters.

The following list points out some of the future work that needs to be done which will improve the advancement of mobile communication systems.

- Implementing compressive sensing in 3G and LTE mobile communication systems
- Parameters like noise, multipath effects and shadowing needed to be considered while measuring the output

- Different transformations need to be tested in order to find the most efficient one for this application
- Design a measurement matrix that will be optimum for speech signals.

References

[1] D.L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory,* vol. 52, pp.1289-1306, 2006.

[2] M. Vetterli, P. Marziliano, and T. Blu, “Sampling Signals with Finite Rate of Innovation,” *IEEE Transaction on Signal Process*, vol. 50, no. 6, pp.1417–1428, 2002.

[3] E. Candes, J. Romberg, and T. Tao, “*Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information,” IEEE Transaction on Information Theory*, vol. 52, pp. 489–509, 2006.

[4] “Compressive Sensing, a New Frame for Imaging”, http://www.cs.jhu.edu/~misha/ Reading Seminar/Papers/Baraniuk06.pdf.

[5] E. Candes and J. Romberg , “Practical Signal Recovery from Random Projections,” *Processing SPIE International Symposium Electronic Imaging*, pp. 76–86, vol. 5674, 2005.

[6] R. Baraniuk and P. Steeghs, "Compressive Radar Imaging," *Radar Conference, 2007 IEEE*, doi:10.1109/RADAR.2007, pp.128-133, 2007.

[7] Compressive Sensing, http://www.ricam.oeaw.ac.at/people/page/fornasier /CSFornasier Rauhut.pdf.

[8] E. Candes and T. Tao. “Near-optimal Signal Recovery from Random Projections and universal encoding strategies,” *IEEE*, vol. 52, pp. 5406 – 5425, 2004.

[9] Disciplined convex programming, http://cvxr.com/cvx/cvx_usrguide.pdf, 2010.

[10] S.M. Kay. *Fundamentals of statistical signal processing*. Prentice Hall, 1998.

[11] M.A. Davenport, M.B.Wakin, and R.G. Baraniuk. Detection and Estimation with Compressive Measurements. Technical report, Department of Electrical and Computer Engineering, Rice University, 2006.

[12] Compressive Imaging, http://citeseerx.ist.psu.edu/.

[13] Compressive Sensing, http://www.see.ed.ac.uk/~mdavies4/Research/CS/.

[14] MATLAB CENTRAL, http://www.mathworks.com/matlabcentral/ fileexchange/7309.

[15] FFT Tutorial, http://www.phys.nsu.ru/cherk/fft.pdf.

[16] A. A. Moghadam, H, Radha, "Hybrid Compressed Sensing of Images," *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*, vol. 99, pp. 99-104, 4-6, 2010.

[17] DCT, http://fourier.eng.hmc.edu/e161/lectures/dct/node1.html.

Appendices

Appendix A: MATLAB Code

MATLAB code for sampling the speech signal in mobile system using compressive sensing (Random measurement matrix)

```
clc;
clear all;

%Fs Hz (samples per second) is the rate at the speech signal is sampled
Fs=2000;
x=wavread('test.wav');
figure(1)
stem(x)
title('Recorded input speech signal');
xlabel('Length of the input speech signal');
ylabel('Amplitude of the input speech signal');

%Discrete cosine transform of the recorded signal
a0=dct(x)
figure(2)
stem(a0)
axis([0 2000 -1 1]);
title('Discrete cosine transform of the recorded signal');
xlabel('Length of the DCT spectrum');
ylabel('Amplitude of the DCT spectrum');

% Thresholding the spectrum to make it sparse
for i=1:1:2000;
if  a0(i,1)<=0.04 && a0(i,1)>=-0.06
        a0(i,1)=0;
else
        a0(i,1)=a0(i,1);
end
end

a0;
figure(3)
stem(a0)
axis([0 2000 -1 1]);
title('The Threshold spectrum');
xlabel('The length of the threshold spectrum');
ylabel('Amplitude of the threshold spectrum');
```

```
% Sparsity of the spectrum(K)and Length of the signal (N)
K=800
N=2000

% Random measurement matrix
disp('Creating measurment matrix...');
A = randn(K,N);
A = orth(A')';
figure(4)
imagesc(A)
colorbar;
colormap('lines');
title('Random Measurement matrix');
disp('Done.');

%  observations vector
y = A*a0;
figure(5)
plot(y)
title('Observation Vector');

%initial guess = min energy
x0 = A'*y;

%solve the LP
tic
xp = l1eq_pd(x0, A, [], y, 1e-2);
toc
figure(6)
plot(xp)
axis([0 2000 -0.6 0.6]);
title(' Reconstructed Spectrum using l1-minimization');

% Inverse dicrete cosine transform of reconstructed signal (IDCT)
Xrec=idct(xp)
wavplay(Xrec,Fs)
figure(7)
stem(Xrec)
title('Reconstructed signal at the receiver');
xlabel('Length of the reconstructed signal using IDCT');
ylabel('Amplitude of the reconstructed signal using IDCT');

% Calculating Absolute error between the reconstructed and actual signal
err=(max(abs(Xrec-x)))
```

```
stem(err);
title(' Absolute Error of Reconstructed spectrum and Threshold spectrum ');
xlabel('Length of the Maximum Absolute Error');
ylabel('Maximum Absolute error')
```

Appendix B

MATLAB code for sampling the speech signal in mobile system using compressive sensing (Pre-defined measurement matrix)

```
clc;
clear all;

% Fs Hz (samples per second) is the rate at the speech signal is sampled
Fs = 2000;

% Recording the speech signal
rec = wavrecord(2000,Fs);


% Playing the recorded signal
wavplay(rec,Fs);
figure(1)
stem(rec)
title('Recorded input speech signal');
xlabel('Length of the input speech signal');
ylabel('Amplitude of the input speech signal');


% Discrete cosine transform of the recorded signal
a0=dct(rec)
figure(2)
stem(a0)
title('Discrete cosine transform of the recorded signal');
xlabel('Length of the DCT spectrum');
ylabel('Amplitude of the DCT spectrum');

% Thresholding the spectrum to make it sparse
for i=1:1:2000;
if  a0(i,1)<=0.05 && a0(i,1)>=-0.06
        a0(i,1)=0;
else
        a0(i,1)=a0(i,1);
end
end
a0;
figure(3)
stem(a0)
title('The Threshold spectrum');
xlabel('The length of the threshold spectrum');
```

```
ylabel('Amplitude of the threshold spectrum');

% Pre-defined measurement matrix
xx=sort(ceil(400*randn(1,1000)));
n=2000
for i=1:n
A(:,i)=sin(i*xx)';
end
A = orth(A')';
figure(4)
plot(A)
title('Pre-defined Measurement matrix');
disp('Done.');


%  Observations matrix
y = A*a0;
figure(5)
plot(y)
title('Observation Vector');


%initial guess = min energy
x0 = A'*y;

%solve the LP
tic
xp = l1eq_pd(x0, A, [], y, 1e-2);
toc
figure(6)
plot(xp)
title(' Reconstructed Spectrum using l1-minimization');

% Calculating the error between the Received reconstructed spectrum and actual
spectrum
figure(7)
stem(abs(xp-a0),'-')
title(' Absolute Error of Reconstructed spectrum and Threshold spectrum ');
xlabel('Length of the Absolute Error between the reconstructed spectrum and transmitted
threshold spectrum');
ylabel('Absolute error');
```

Appendix B (Continued)

```
% Inverse dicrete cosine transform of reconstructed signal

x0=idct(xp)
wavplay(x0,Fs)
figure(8)
stem(x0)
title('Reconstructed signal at the receiver');
xlabel('Length of the reconstructed signal using IDCT');
ylabel('Amplitude of the reconstructed signal using IDCT');
```

Appendix C

MATLAB code for sampling the Laplacian speech signal generated by randraw function

```
clc;
clear all;
close all;

%Length of the signal
L=512;
x=randraw('laplace', [100, 6], 1e2 );
figure(1)
plot(x)
title('The input speech signal');
xlabel('The length of the signal');
ylabel('Amplitude of the signal');

%Fourier transform of the input speech signal
y0=fft(x,L);
z=abs(y0);
a0=fftshift(z);
figure(2)
stem(a0)
axis([0 512 0 11000]);
title('Power spectrum of input speech signal');
xlabel('The length of the spectrum');
ylabel('Magnitude');

%Making the signal sparse by threshholding
for i=1:1:L;
if a0(i,1)>=2000
        a0(i,1)=a0(i,1);
else
        a0(i,1)=0;

end
end
 a0;

% Making the signal sparse
figure(3)
stem(a0)
axis([0 512 0 12000]);
title('Threshold spectrum');
```

```
xlabel('The length of the threshold spectrum');
ylabel('Amplitude of the threshold spectrum');
K=200
N=512

% measurement matrix
disp('Creating measurment matrix...');
A = randn(K,N);
A = orth(A')';
disp('Done.');

% observations
y = A*a0;

% initial guess = min energy
x0 = A'*y;

% solve the LP
tic
xp = l1eq_pd(x0, A, [], y, 1e-3);
toc

% Reconstructed output
figure(4)
stem(xp)
axis([0 512 0 11000]);
title('The reconstructed received spectrum');
xlabel('The length of the received spectrum');
ylabel('Amplitude of the received spectrum');

%Absolute error between the reconstructed and original signal
figure(5)
stem(abs(xp-a0),'-')
axis([0 512 0 0.004]);
title(' Error between the reconstructed and actual spectrum ');
xlabel('The length of the spectrum');
ylabel('Absolute error');
```

Appendix D

MATLAB code for compressing the test signal using wavelet compression

```
% Load original one-dimensional signal.
Fs=2000
s=wavread('test.wav')';
figure(1)
stem(s)
title('Input speech signal');
xlabel('Length of the input speech signal');
ylabel('Amplitude of the input speech signal');
l_s = length(s);

% Wavelet transform of input signal
[cA1,cD1] = dwt(s,'db1');

%To extract the Level-1 Approximation and Detail coefficient
A1 = idwt(cA1,[],'db1',l_s);
D1 = idwt([],cD1,'db1',l_s);
figure(2)
subplot(1,2,1); plot(A1); title('Approximation A1')
subplot(1,2,2); plot(D1); title('Detail D1')

%Inverse Wavelet transform of Approximation and detail coefficient
A0 = idwt(A1,D1,'db1',l_s);
wavplay(A0,Fs)
figure(3)
stem(A0)
title('Recontructed speech signal');
xlabel('Length of the reconstructed speech signal');
ylabel('Amplitude of the reconstructed speech signal');
err = max(abs(s-A0))
```

Appendix E

MATLAB code for reconstructing the speech signal using l_1-minimization

```
 l1eq_pd.m
%
% Solve
% min_x ||x||_1  s.t.  Ax = b
%
% Recast as linear program
% min_{x,u} sum(u)  s.t.  -u <= x <= u,  Ax=b
% and use primal-dual interior point method
%
% Usage: xp = l1eq_pd(x0, A, At, b, pdtol, pdmaxiter, cgtol, cgmaxiter)
%
% x0 - Nx1 vector, initial point.
%
% A - Either a handle to a function that takes a N vector and returns a K
%     vector , or a KxN matrix.  If A is a function handle, the algorithm
%     operates in "largescale" mode, solving the Newton systems via the
%     Conjugate Gradients algorithm.
%
% At - Handle to a function that takes a K vector and returns an N vector.
%      If A is a KxN matrix, At is ignored.
%
% b - Kx1 vector of observations.
%
% pdtol - Tolerance for primal-dual algorithm (algorithm terminates if
%     the duality gap is less than pdtol).
%     Default = 1e-3.
%
% pdmaxiter - Maximum number of primal-dual iterations.
%     Default = 50.
%
% cgtol - Tolerance for Conjugate Gradients; ignored if A is a matrix.
%     Default = 1e-8.
%
% cgmaxiter - Maximum number of iterations for Conjugate Gradients; ignored
%     if A is a matrix.
%     Default = 200.
%
% Written by: Justin Romberg, Caltech
% Email: jrom@acm.caltech.edu
% Created: October 2005
```

```
function xp = l1eq_pd(x0, A, At, b, pdtol, pdmaxiter, cgtol, cgmaxiter)

largescale = isa(A,'function_handle');

if (nargin < 5), pdtol = 1e-3;  end
if (nargin < 6), pdmaxiter = 50;  end
if (nargin < 7), cgtol = 1e-8;  end
if (nargin < 8), cgmaxiter = 200;  end

N = length(x0);

alpha = 0.01;
beta = 0.5;
mu = 10;

gradf0 = [zeros(N,1); ones(N,1)];

% starting point --- make sure that it is feasible
if (largescale)
if (norm(A(x0)-b)/norm(b) > cgtol)
   disp('Starting point infeasible; using x0 = At*inv(AAt)*y.');
   AAt = @(z) A(At(z));
   [w, cgres, cgiter] = cgsolve(AAt, b, cgtol, cgmaxiter, 0);
if (cgres > 1/2)
     disp('A*At is ill-conditioned: cannot find starting point');
     xp = x0;
return;
end
   x0 = At(w);
end
else
if (norm(A*x0-b)/norm(b) > cgtol)
   disp('Starting point infeasible; using x0 = At*inv(AAt)*y.');
   opts.POSDEF = true; opts.SYM = true;
   [w, hcond] = linsolve(A*A', b, opts);
if (hcond < 1e-14)
     disp('A*At is ill-conditioned: cannot find starting point');
     xp = x0;
return;
end
   x0 = A'*w;
```

```
end
end
x = x0;
u = (0.95)*abs(x0) + (0.10)*max(abs(x0));

% set up for the first iteration
fu1 = x - u;
fu2 = -x - u;
lamu1 = -1./fu1;
lamu2 = -1./fu2;
if (largescale)
  v = -A(lamu1-lamu2);
  Atv = At(v);
  rpri = A(x) - b;
else
  v = -A*(lamu1-lamu2);
  Atv = A'*v;
  rpri = A*x - b;
end

sdg = -(fu1'*lamu1 + fu2'*lamu2);
tau = mu*2*N/sdg;

rcent = [-lamu1.*fu1; -lamu2.*fu2] - (1/tau);
rdual = gradf0 + [lamu1-lamu2; -lamu1-lamu2] + [Atv; zeros(N,1)];
resnorm = norm([rdual; rcent; rpri]);

pditer = 0;
done = (sdg < pdtol) | (pditer >= pdmaxiter);
while (~done)

pditer = pditer + 1;

w1 = -1/tau*(-1./fu1 + 1./fu2) - Atv;
w2 = -1 - 1/tau*(1./fu1 + 1./fu2);
w3 = -rpri;

sig1 = -lamu1./fu1 - lamu2./fu2;
sig2 = lamu1./fu1 - lamu2./fu2;
sigx = sig1 - sig2.^2./sig1;

 if (largescale)
    w1p = w3 - A(w1./sigx - w2.*sig2./(sigx.*sig1));
    h11pfun = @(z) -A(1./sigx.*At(z));
```

```
[dv, cgres, cgiter] = cgsolve(h11pfun, w1p, cgtol, cgmaxiter, 0);
if (cgres > 1/2)
      disp('Cannot solve system.  Returning previous iterate.  (See Section 4 of notes for
more information.)');
      xp = x;
return
end
    dx = (w1 - w2.*sig2./sig1 - At(dv))./sigx;
    Adx = A(dx);
    Atdv = At(dv);
else
    w1p = -(w3 - A*(w1./sigx - w2.*sig2./(sigx.*sig1)));
    H11p = A*(sparse(diag(1./sigx))*A');
    opts.POSDEF = true; opts.SYM = true;
    [dv,hcond] = linsolve(H11p, w1p, opts);
if (hcond < 1e-14)
      disp('Matrix ill-conditioned.  Returning previous iterate.  (See Section 4 of notes for
more information.)');
      xp = x;
return
end
    dx = (w1 - w2.*sig2./sig1 - A'*dv)./sigx;
    Adx = A*dx;
    Atdv = A'*dv;
end

  du = (w2 - sig2.*dx)./sig1;

  dlamu1 = (lamu1./fu1).*(-dx+du) - lamu1 - (1/tau)*1./fu1;
  dlamu2 = (lamu2./fu2).*(dx+du) - lamu2 - 1/tau*1./fu2;

  % make sure that the step is feasible: keeps lamu1,lamu2 > 0, fu1,fu2 < 0
  indp = find(dlamu1 < 0);  indn = find(dlamu2 < 0);
  s = min([1; -lamu1(indp)./dlamu1(indp); -lamu2(indn)./dlamu2(indn)]);
  indp = find((dx-du) > 0);  indn = find((-dx-du) > 0);
  s = (0.99)*min([s; -fu1(indp)./(dx(indp)-du(indp)); -fu2(indn)./(-dx(indn)-du(indn))]);

  % backtracking line search
  suffdec = 0;
  backiter = 0;
while (~suffdec)
    xp = x + s*dx;  up = u + s*du;
    vp = v + s*dv;  Atvp = Atv + s*Atdv;
    lamu1p = lamu1 + s*dlamu1;  lamu2p = lamu2 + s*dlamu2;
    fu1p = xp - up;  fu2p = -xp - up;
    rdp = gradf0 + [lamu1p-lamu2p; -lamu1p-lamu2p] + [Atvp; zeros(N,1)];
    rcp = [-lamu1p.*fu1p; -lamu2p.*fu2p] - (1/tau);
    rpp = rpri + s*Adx;
    suffdec = (norm([rdp; rcp; rpp]) <= (1-alpha*s)*resnorm);
    s = beta*s;
    backiter = backiter + 1;
if (backiter > 32)
      disp('Stuck backtracking, returning last iterate.  (See Section 4 of notes for more
information.)')
      xp = x;
return
end
end


  % next iteration
  x = xp;  u = up;
  v = vp;  Atv = Atvp;
  lamu1 = lamu1p;  lamu2 = lamu2p;
  fu1 = fu1p;  fu2 = fu2p;

  % surrogate duality gap
  sdg = -(fu1'*lamu1 + fu2'*lamu2);
  tau = mu*2*N/sdg;
  rpri = rpp;
  rcent = [-lamu1.*fu1; -lamu2.*fu2] - (1/tau);
  rdual = gradf0 + [lamu1-lamu2; -lamu1-lamu2] + [Atv; zeros(N,1)];
  resnorm = norm([rdual; rcent; rpri]);
  done = (sdg < pdtol) | (pditer >= pdmaxiter);
  disp(sprintf('Iteration = %d, tau = %8.3e, Primal = %8.3e, PDGap = %8.3e, Dual res =
%8.3e, Primal res = %8.3e',...
    pditer, tau, sum(u), sdg, norm(rdual), norm(rpri)));
if (largescale)
    disp(sprintf('CG Res = %8.3e, CG Iter = %d', cgres, cgiter));
else
    disp(sprintf('H11p condition number = %8.3e', hcond));
end
end
```