USING HOMOTOPY ALGORITHM FOR BLIND SOURCE SEPARATION

A THESIS

submitted by

DHULIPALA PRANAV VAIDIK

for the award of the degree

of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL EGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2014

THESIS CERTIFICATE

This is to certify that the thesis titled **RESULTS OF USING HOMOTOPY ALGORITHM FOR SOLVING UNDERDETERMINED BLIND SOURCE SEPARATION PROBLEM**, submitted by **D Pranav Vaidik (EE10B008)**, to the Indian Institute of Technology Madras, Chennai for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. R Aravind
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 22nd May 2014

ACKNOWLEDGEMENTS

I sincerely thank Prof. R Aravind for giving me the opportunity to work and learn under him and for his invaluable suggestions during the course of the project

I am grateful for the help provided by Prof. Ananth Krishnan, whose guidance during the course “Computer Methods in EE” helped to overcome some of the major problems encountered during the project.

ABSTRACT

KEYWORDS: Blind Source Separation, Underdetermined Source Separation, Compressed Sensing, Sparse Signal Representation, l_1-minimization, Homotopy.

In this work we aim to solve the underdetermined Blind Source Separation [7] problem using sparse representation. For reconstruction of such sparse representations, we use Compressed Sensing [5], [6] framework, an emerging technique for efficient sparse data reconstruction. We use the two-staged approach proposed in [1], replacing the l_1-minimization algorithm in its second stage with a Homotopy based algorithm. The scope of this work is the separation of N audio source signals from M linear mixture signals each of which is a linear combination of the sources, while the proportions of mixing are unknown and the system is underdetermined, i.e. M<N. We restrict ourselves to the case where there are only two mixture signals i.e. M=2. This document demonstrates the results of using a Homotopy algorithm, a Compressed Sensing (CS) (see [5] and [6]) based approach towards the above mentioned problem and compares it with that of l_1-norm minimization approach against various data sets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

BSS	Blind Source Separation
CS	Compressed Sensing
ICA	Independent Component Analysis
PCA	Principal Component Analysis
LASSO	Least Absolute Shrinkage and Selection Operator
LARS	Least Angle Regression
SNR	Signal to Noise Ratio

CHAPTER 1

INTRODUCTION

Human acoustic systems are very efficient in distinguishing the sources from a mixture of audio signals, classically referred to as the cocktail party problem. On the other hand, it is quite difficult task for a machine to solve the same problem with very limited prior information about the sources and the mixing environment. Blind Source Separation (BSS) [7] is one of the widely used techniques to address this problem. In practice, however, we encounter the cases in which the number of mixtures obtained is less than the number of sources, i.e. the underdetermined case.

In this document, we present the results of an approach to a problem which aims to separate N sources from a set of M mixtures each of which is a linear combination of the sources, while the mixing matrix is unknown. In the case of Underdetermined Blind Source Separation [7] problem, the number of mixtures M is smaller than the number of sources N i.e. $M<N$, considering the noiseless case. The Independent Component Analysis (ICA) and Principle Component Analysis (PCA) (see [8], [9]) are some of the widely used techniques to approach the BSS problem, however solutions involving sparse signal representations (see [1],[2]) have been shown to be efficient for the solution of Underdetermined BSS problem.

When based on sparse representation, a two-staged approach is usually employed to recover the source signals, assuming that the sources are sparse signals. The first stage involves estimation of the unknown mixing matrix and the second involves the use of a non-linear optimization algorithm to recover the source signals. Following a similar approach, we propose to use a Compressed Sensing (CS) [5], [6] based algorithm for the second stage of the solution.

In this document, we propose to introduce the Homotopy algorithm to replace the l_1-minimization algorithm in the solution described in [1] to solve the above mentioned problem and discuss the results obtained from this alteration.

1.1. The Basic Problem

Let us consider M distinct sensors which give a corresponding output $\boldsymbol{x^t}$, an M-dimensional column vector at a given discrete time instant t, and let $\mathbf{X}$ be an $M \times T$ matrix corresponding to the sensor data up to time instant T i.e., t=1, 2,…, T (i^{th} row of $\mathbf{X}$ denotes the i^{th} mixture signal acquired by the i^{th} sensor and j^{th} column of $\mathbf{X}$ is $\boldsymbol{x^j}$ for every $j<T$). Similarly, let $\mathbf{S}$ be the $N \times T$ matrix of underlying source signals with each row representing one of the source signals and let A be the mixing matrix of the order $M \times N$. In the noiseless case, the problem of blind source separation [7] consists of finding the solution to the following system of equations:

$$\boldsymbol{X = AS} \tag{1}$$

Where, $\mathbf{X}$ is the only known quantity in the above equation in the case of solving BSS [7] Problem. This equation can be further split into numerous equations as

$$\boldsymbol{x^t = As^t} \tag{2}$$

Where, $\boldsymbol{s^t}$ is the N-dimensional column vector corresponding to the source signals at a given instant t. When we further decompose the matrix $\mathbf{A}$ into a set of column vectors $\boldsymbol{a^j}$ for j = 1, 2…, N, we can rewrite Eq. (2) as

$$\boldsymbol{x^t} = \sum_{j=1}^{N} \boldsymbol{a^j}\, s_j^t \tag{3}$$

Where, s_j^t is the corresponding element of the matrix $\mathbf{S}$. Without loss of generality, we normalize all the $\boldsymbol{a^j}$'s to unit length. The solution to the system of equations (3) is demonstrated in two steps. The first step involves the estimation of the column vectors $\boldsymbol{a^j}$'s and the estimate of the mixing matrix $\widehat{\boldsymbol{A}}$ is obtained. In the second step, we proceed towards solving the system of equations (2) for each discrete time instant t and thus obtain the estimation of the source matrix $\widehat{\boldsymbol{S}}$. For the second step, for solving the set of equations (2) in of M<N, we employ the Compressed Sensing (CS) [5], [6] based methods to obtain the solution.

CHAPTER 2

APPROACH TO THE SOLUTION

2.1. Step 1: Estimation of Mixing Matrix

Following equation (3), if we assume that, at every time instant *t*, only one of the s_j^t's is significant compared to the other components, i.e. all the other s_j^t's are zeroes or close to zero, then the corresponding $\boldsymbol{x^t}$'s are aligned along one of the directions $\boldsymbol{a^j}$'s. Thus, given that the source matrix is sparse enough, we can find that in the scatter plot of $\boldsymbol{x^t}$, a density of data points demonstrate a tendency to cluster themselves along the directions of $\boldsymbol{a^j}$'s. Hence, if we could obtain the vectors along which the scatter plot aligns, we can obtain the estimate of the mixing matrix. However, in most cases we encounter, the data in the time domain is not sparse. When a scatter plot of such data is plotted, we see that they don't necessarily align themselves along any particular direction. Hence the data sets in time domain is often not sparse enough for the above approach. Figure 2.1 shows an example of scatter plot of a data set in time domain. In the case of this data, M=2 and the scatter plot of row vectors $\mathbf{x_1}$ and $\mathbf{x_2}$, the row vectors obtained from the rows of the mixture matrix **X**, plotted against each other. From the plot, the points seemed to spread uniformly to a large extent and hence making it a very difficult task to estimate the column vector directions $\boldsymbol{a^j}$'s. Since we need a sparser data in order to estimate the mixing matrix, a possible approach is to look for a linear transform T in the domain of which, the data can be sparser. Since the transform is linear, the equation (1) on transforming to the domain of T becomes

$$T(\boldsymbol{X}) = \boldsymbol{A}T(\boldsymbol{S}) \tag{4}$$

Due to the transform T being linear, the mixing matrix is preserved. Transformation to frequency domain satisfies the criteria that are specified above as frequency domain is sparser than the time domain.

For transformation to frequency domain, each signal is processed into frames of length *L* with a hopping distance *d* as the distance between the starting points of successive frames. Each frame is multiplied with a Hanning Window of length *L* and transformed

with a standard FFT of length L.All the transformed frames are concatenated into a single vector. In our work, L is taken as $\frac{1}{10^{th}}$ of the sampling rate and d is approximately $0.3L$.

The scatter plot obtained in the transform domain was much more encouraging as the data points can be clearly seen to cluster along certain directions. Figure 2.2 shows the scatter plot of the data set plotted in Figure 2.1 in the frequency domain.

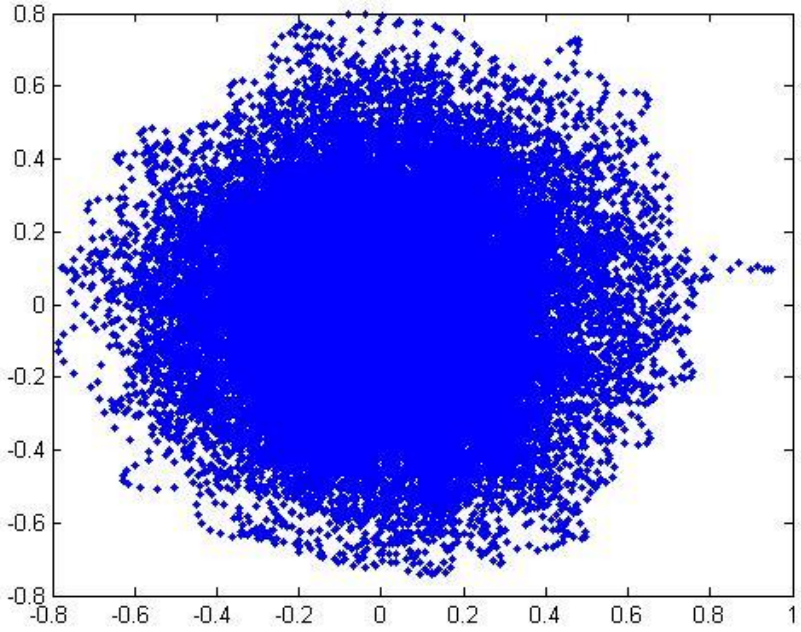


Figure 2.1: Scatter plot of row vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ for the data in the SixFlutes dataset in time domain.

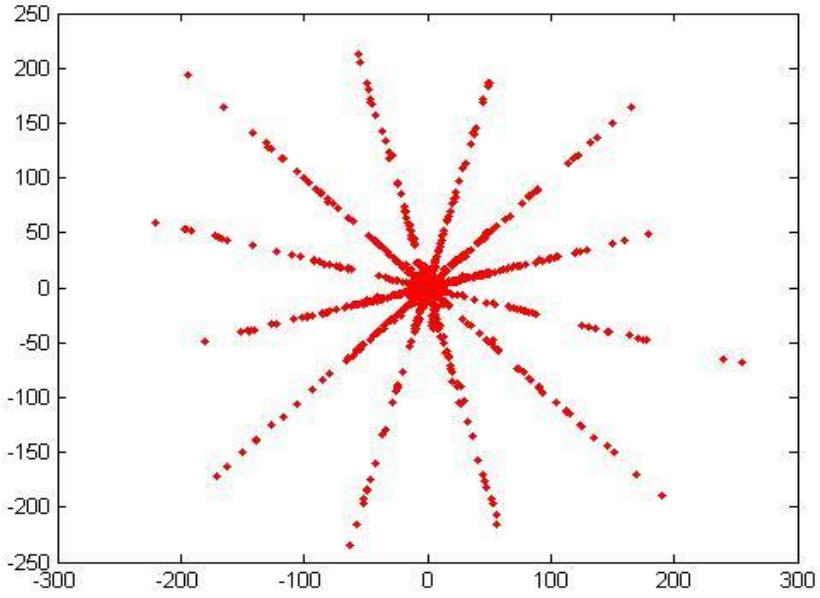


Figure 2.2: Scatter plot of row vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ for the data in the SixFlutes dataset in frequency domain

Now, since we have a scatter plot whose points clearly cluster along some of the directions, we need to establish a mathematical method to obtain the directional vectors in those specific directions. From the scatter plot, we can observe that the points that are farther from the origin tend to be closer to the directional vectors, thus making them the points that can contribute more towards finding the vectors. The approach we take is to define a potential function that is a function of the distance of the data points from the origin and also their inclination. In the case of M=2, we made use of the potential function described in [1] and the idea is to find the angles at which the potential function attains its peak values on a polar plot. The calculation of potential function is described in Appendix A. Once we obtain these angles, the directional vectors $\boldsymbol{a^j}$'s are estimated to be the unit vectors along these angles.

It is to be noted that the number of peak values in the polar plot is equal to the number of $\boldsymbol{a^j}$'s we acquire and thus equal to the number of source signals we estimate. Thus proper care should be taken in finding the location of the peaks of the polar plot. A reliable method is to down sample the data from potential function and to use polynomial interpolation to up sample it. This ensures that the polar plot is smoother and helps in

getting the exact number of peaks we intend to get. It is also recommended to define a certain lower threshold in finding the peaks in order to avoid the insignificant peaks, but it usually depends on the mixture signal data set and also we may miss a significant peak when the threshold is not properly set.

After the estimated column vectors are obtained, they are concatenated column wise to get the estimated mixing matrix $\widehat{\boldsymbol{A}}$. However, the order of the concatenation is of no importance as it only effects the row wise ordering of sources in the source matrix, but nevertheless, the individual source signals obtained are unaffected by this.

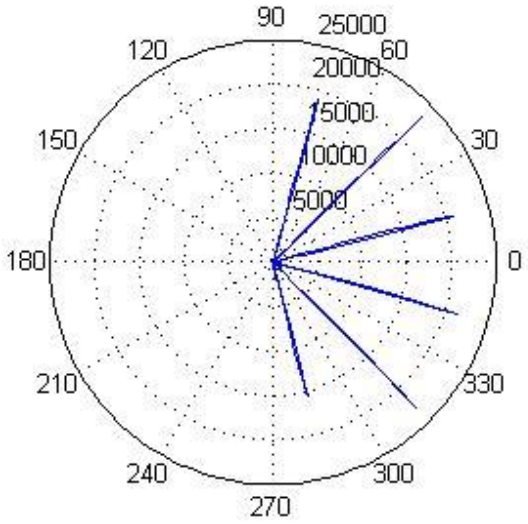


Figure 2.3: Polar plot of the potential function obtained for the data in the SixFlutes dataset

2.2. Step 2: Estimation of the Source Signals

After successfully obtaining the estimated mixing matrix $\widehat{\boldsymbol{A}}$, we proceed towards solving the equation (2). Even though $\widehat{\boldsymbol{A}}$ is known, this system of equations is underdetermined. Since the source signals are sparse in the transform domain, Compressed Sensing [5], [6] based techniques can be employed to solve the set of equations. In this document, we

have used the Homotopy based algorithm to solve the set of equations, explained in detail in the next chapter.

CHAPTER 3

THE HOMOTOPY APPROACH

3.1. The Algorithm

In the Homotopy based approach (see [3], [4]), we initialize the solution as a zero vector and construct a sparse solution in an iterative basis until the solution is under required tolerance. Homotopy is an approach to the solution to the Least Absolute Shrinkage and Selection Operator (LASSO) (see [10]) problem described by the unconstrained optimization problem

$$\text{argmin}_{\boldsymbol{h}\in\mathbb{C}^N}\|\boldsymbol{Y}-A\boldsymbol{h}\|_2^2/2+\lambda\|\boldsymbol{h}\|_1 \tag{5}$$

Where $\boldsymbol{Y}$ is a column vector of frequency domain representation of the mixture signal matrix $\boldsymbol{X}$, $\boldsymbol{h}$ is the corresponding temporary solution that is updated with every iteration. λ is a parameter that is also updated by the end of every iteration. As $\boldsymbol{h}$ moves closer to the solution, λ moves closer to zero. In a geometric sense, Homotopy iteratively selects the column vectors and moves in a direction that is equiangular with all the selected column vectors, and eventually forms a linear combination of the all the vectors that is closest to the solution.

Since we are presently dealing with vectors in with complex entries, we use a complex-valued Homotopy based algorithm to arrive to a generalized solution for the above problem. In the approach via Homotopy, we start with a large $\lambda \in \mathbb{R}$ and with $\boldsymbol{h}$ =0 as the initial solution to the problem. The algorithm terminates as $\lambda\rightarrow 0$, when $\boldsymbol{h}$ converges to the solution to the 'noiseless' sparse recovery problem

$$\min_{\boldsymbol{h}\in\mathbb{C}^N}\|\boldsymbol{h}\|_1 \text{ s.t } \boldsymbol{Y}=A\boldsymbol{h} \tag{6}$$

which is the also known as the l_1-norm minimization problem. In case of 'noisy' sparse recovery problem, we should change the stopping condition from $\lambda\rightarrow 0$ to

$$\|\boldsymbol{Y}-A\boldsymbol{h}\|_2 \leq \boldsymbol{\sigma_n} \tag{7}$$

Where, $\boldsymbol{\sigma_n}$ denotes the tolerance of error in reconstruction. Now, let us define a function f_λ such that,

$$f_\lambda(\boldsymbol{h}) = \frac{\|\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{h}\|_2^2}{2} + \lambda\|\boldsymbol{h}\|_1 \tag{8}$$

If for a given λ, if $\boldsymbol{h}$ is the minimizer of f_λ, then its sub differential at $\boldsymbol{h}$ is zero. So from equation (8) we get

$$\partial f_\lambda(\boldsymbol{h}) = -A^H(\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{h}) + \lambda\|\boldsymbol{h}\|_1 = 0 \tag{9}$$

where the sub differential $\boldsymbol{\partial}\|\boldsymbol{h}\|_1$ of $\|\boldsymbol{h}\|_1$ is defined by

$$\boldsymbol{\partial}\|\boldsymbol{h}\|_1 = \begin{cases} \omega \in \mathbb{C}^L \;\middle|\; \begin{matrix} \omega(i) = \frac{\boldsymbol{h}(i)}{|\boldsymbol{h}(i)|}, \boldsymbol{h}(i) \neq 0 \\ \omega(i) = \{v \in \mathbb{C} \mid |v| \leq 1\}, \boldsymbol{h}(i) = 0 \end{matrix} \end{cases} \tag{10}$$

Where h(i) and ω(i) are the i^{th} components of $\boldsymbol{h}$ and ω respectively. Let $\boldsymbol{c}$ denote the correlations between the mixing matrix and the residue and I denote the support of $\boldsymbol{h}$, both of which are given by

$$\boldsymbol{c} = \boldsymbol{A}^H(\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{h})$$

$$\boldsymbol{I} = \{\boldsymbol{i}|\boldsymbol{h}(\boldsymbol{i}) \neq \boldsymbol{0}\}$$

Thus the condition $\boldsymbol{f_\lambda(h)} = \boldsymbol{0}$ can be equivalently written as

$$\boldsymbol{c} = \boldsymbol{\lambda}\boldsymbol{\partial}\|\boldsymbol{h}\|_1 \tag{11}$$

which can be further written as

$$\begin{cases} \boldsymbol{c}(\boldsymbol{I}) = \boldsymbol{\lambda}\frac{\boldsymbol{h}(\boldsymbol{I})}{|\boldsymbol{h}(\boldsymbol{I})|} \\ |\boldsymbol{c}(\boldsymbol{I}^c)| \leq \boldsymbol{\lambda} \end{cases} \tag{12}$$

where $\boldsymbol{I^c}$, $\boldsymbol{c(I)}$ and $\boldsymbol{h(I)}$ denote the complement of I, correlations on the support I and the components of $\boldsymbol{h}$ on I respectively. The conditions established in (11) are to be maintained by the algorithm throughout process of tracing the solution path. In every step l, the step size $\boldsymbol{\gamma_l} \in \mathbb{R}$ and the moving direction $\boldsymbol{d_l} \in \mathbb{C}^N$ have to be found in order to update the solution to

$$\boldsymbol{h_{l+1} = h_l + \gamma_l d_l} \tag{13}$$

After every iteration, we update the parameter λ_l as

$$\boldsymbol{\lambda_{l+1} = \lambda_l - \gamma_l} \tag{14}$$

Let us define the support of $\boldsymbol{h_l}$, $\boldsymbol{I_l = \{i | h_l(i) \neq 0\}}$ for the l^{th} step of the solution. Since the components of $\boldsymbol{h_l}$ corresponding to $\boldsymbol{I_l}$ always follow (11), we have

$$\boldsymbol{c_{l+1}(I_l) = (\lambda_l - \gamma_l)\frac{h_l^t(I_l)}{|h_l^t(I_l)|}} \tag{15}$$

On the other hand, we also have to update the remaining elements of $\boldsymbol{c_l}$, so we have

$$\boldsymbol{c_{l+1} = A^H(Y^t - Ah_{l+1})} \tag{16}$$

Using (12) in the above equation, we get

$$\boldsymbol{c_{l+1} = A^H(y - Ah_l) - \gamma_l A^H A d_l} \tag{17}$$

Now, we define

$$\boldsymbol{c_l(I_l^c) = A^H(I_l^c)(y - Ah_l)}$$

$$\boldsymbol{d\mu(I_l^c) = A^H(I_l^c)Ad_l}$$

Using the above definitions in (17), we get

$$\boldsymbol{c_{l+1}(I_l^c) = c_l(I_l^c) - \gamma_l d\mu(I_l^c)} \tag{18}$$

When we put equations (13) and (11) in equation (15), we get equation (19), from which we can solve for $\boldsymbol{d_l}$ entries corresponding to $\boldsymbol{I_l}$ and setting the remaining components of $\boldsymbol{d_l}$ to zero, i.e.,

$$\boldsymbol{A^H(I_l)A(I_l)d_l(I_l) = c_l(I_l)/\lambda_l} \tag{19}$$

$$\boldsymbol{d(I_l^c) = 0}$$

Thus, so far we have successfully transformed the conditions in (12) in to equations (15) and (18) which can be more easily applied during the algorithm. When we generally look at the conditions in (12), we see that the condition breaks either when one of the

non-zero components of $\boldsymbol{h_l}$ crosses zero or when one of the zero components of $\boldsymbol{h_l}$ becomes non-zero, whichever of them happens first while $\boldsymbol{h_l}$ moves in the direction $\boldsymbol{d_l}$. The algorithm updates the direction $\boldsymbol{d_l}$ towards which we move towards the solution when this condition breaks.

First, let us consider the case where one of the components of $\boldsymbol{h_l}$ corresponding to I_l crosses zero. Now we have to find the least value of the step size $\boldsymbol{\gamma_l} \in \mathbb{R}$ for which this happens. Now, we define

$$\boldsymbol{I_s = \left\{ i \epsilon I_l \middle| Im\left\{\frac{h_l(i)}{d_l(i)}\right\} = 0 \right\}} \tag{20}$$

Thus in the above set, we include all the indices in I_l for which the step size $\boldsymbol{\gamma_l}$ is real. We define the step size obtained in this step as

$$\boldsymbol{\gamma_l^- = \min_{i \epsilon I_s} \frac{-Re\{h_l(i)\}}{Re\{d_l(i)\}}} \tag{21}$$

Coming to the other case, where one of the components of $\boldsymbol{h_l}$ corresponding to I_l^c becomes non-zero. So the correlation corresponding to that component also satisfies (13), i.e. for $\boldsymbol{i \in I_l^c}$ we have

$$\boldsymbol{|c_{l+1}(i)| = \lambda_{l+1}}$$

which implies

$$\boldsymbol{|c_l(i) - \gamma_l d\mu(i)| = (\lambda_l - \gamma_l)} \tag{22}$$

For simplicity of solving the above equation, let us define

$$\boldsymbol{c_R = Re\{c_l(i)\}}$$

$$\boldsymbol{c_I = Im\{c_l(i)\}}$$

$$\boldsymbol{\mu_R = Re\{\mu_l(i)\}}$$

$$\boldsymbol{\mu_I = Im\{\mu_l(i)\}}$$

Now we can see that for solving (22) for $\boldsymbol{\gamma_l}$, we end up solving the quadratic equation

$$(\mu_R^2 + \mu_I^2 - 1)\gamma_l^2 + 2(\lambda_l - c_I\mu_I - c_R\mu_R)\,\gamma_l + (c_R^2 + c_I^2 - \lambda_l^2) = 0 \quad (23)$$

We solve the above equation for all $i \in I_l^c$ and consider only the smaller solution for each i. Thus, out of all the solutions obtained for each $i \in I_l^c$, we consider the smallest solution and define the step size obtained as

$$\gamma_l^+ = \min_{i \epsilon I_l^c} \gamma_l(i) \quad (24)$$

As the step size is the maximum real number for which (13) and (18) are maintained, the step size we decide should be the minimum of the step sizes obtained from both the cases above. Thus, we get

$$\gamma_l = min\{\gamma_l^+, \gamma_l^-\} \quad (25)$$

If the step size obtained is from γ_l^+, the corresponding index is added to the set I_l. Else, it is removed from the set and thus the set I_l is updated along with the other parameters before we go to the next step. These steps are repeated until (7) is satisfied.

At the beginning of the algorithm, $\boldsymbol{h_l}$ and $\boldsymbol{d_l}$ are initialized as zero vectors. I_l and λ_l are initialized as the position and absolute value of the largest entry of $\boldsymbol{c_0}$ respectively. In every iteration, $\boldsymbol{d_l}$ is determined from (19) and then γ_l^+ and γ_l^- are calculated as mentioned above. After determining γ_l from (25), $\boldsymbol{h_l}$, λ_l and I_l are updated accordingly. The algorithm continues until (7) is satisfied.

CHAPTER 4

SIMULATION AND RESULTS

4.1. Performing Measures

After the estimated source matrix $\hat{S}$ is obtained in the frequency domain, it is transformed back into time domain. Since the transformations in both ways involve multiplication of the signals with the Hanning window, the edges of the source matrix obtained are forced to zero. Hence when error is measured by taking norm of the difference between the original and estimated signals, it does not go below a certain value, no matter how close the signal is estimated to the original one. Thus, this method of estimating the performance is not recommended as this method does not show the closeness of the estimated signal to the original one.

Since the norm of absolute error is not a reliable measure of estimation of performance, 0we use relative error of the estimated source matrix obtained using Homotopy based approach with respect to that of the l_1-minimization method used in [1]. As signals recovered from both of these methods involve multiplication of the Hanning window, the relative error measured in this case is more reliable. Thus, to measure the performance of our approach, we use the Signal to Noise Ratio (SNR) for the recovered signal with respect that of l_1-minimization given by

$$SNR = -10\log_{10}\frac{\|\hat{S}_h - \hat{S}_{l_1}\|}{\|\hat{S}_{l_1}\|} \qquad (6)$$

Where, $\hat{S}_h$ denotes the signal recovered via Homotopy based approach and $\hat{S}_{l_1}$ denotes the signal recovered using l_1-minimization method.

4.2. Simulation and Results

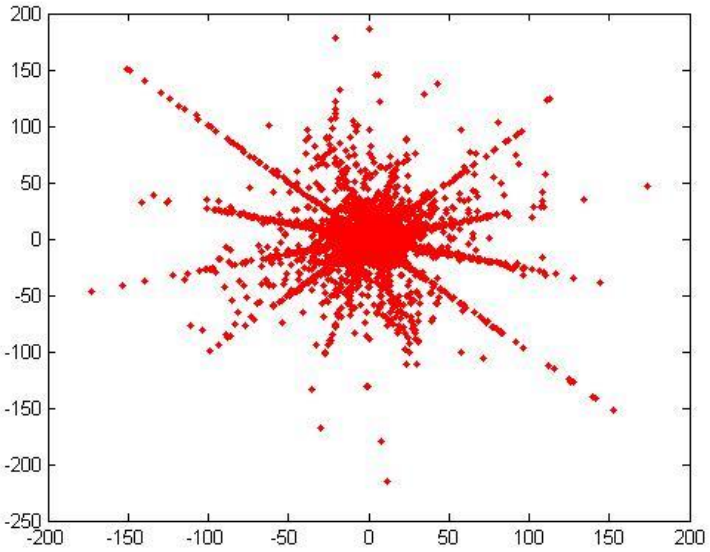
The above approach was tested on some of the sound examples mentioned in [12]. These sound samples can be classified into static and dynamic signals, whose results are discussed separately below. Table 4.1 shows the relative SNR values obtained for each of the recovered signals for various datasets. The parameter used for measuring the potential function is as described in [1].

The approach was first tested on the SixFlutes data, which is a steady source. The data set consists of 6 different musical notes from a flute mixed linearly in two different proportions. Figure 2.2 shows the scatter plot corresponding to this dataset. From this plot we see that the mixture signals in the frequency domain aligned very neatly in the directions of the column vectors and hence the estimated mixing matrix is estimated more accurately. When the algorithm was tested on this data, the estimated sources obtained showed no difference when played and heard, as compared to the original sources. When the plots of the estimated sources were examined, it was found that each of these signals were very close to the original sources, except that the edges have been forced to zero.. This was due to the multiplication of the signals with the Hanning window during the transformations to frequency domain and back. It was also observed that the sources tend to converge towards the solution obtained using the l_1-minimization approach, as the tolerance was reduced. It was also observed that the relative error of both the methods were well under the tolerance value forced on the Homotopy algorithm. This was an expected observation as the LASSO solution is known to converge to the solution of l_1-minimization problem as $\lambda \rightarrow 0$.

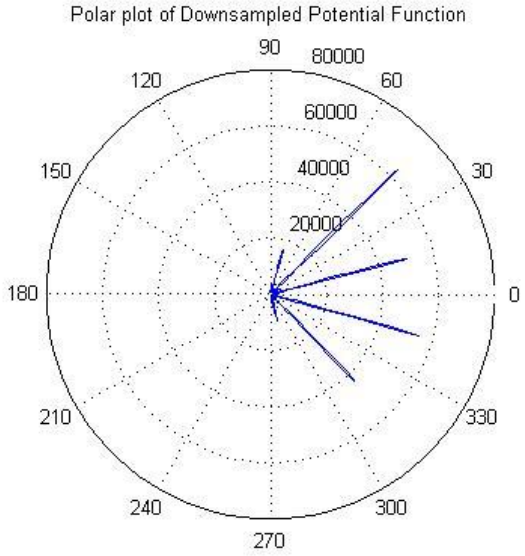
Table 4.1: Relative SNR values obtained for each recovered signal from the respective datasets, when tolerance for Homotopy is set to 1×10^{-5}.

	Source 1	Source 2	Source 3	Source 4	Source 5	Source 6
SixFlutes	163.39	161.93	162.33	162.78	163.49	163.83
SixFluteMelodies	162.48	164.82	165.54	163.23	164.14	162.36
FourVoices	164.29	164.98	163.54	161.71	-	-

Next, the approach was used on another data set SixFluteMelodies, this time a mixture of six different pieces of flute melodies mixed linearly in two different proportions. This dataset was comparatively more complicated as each of the melodies contained various musical notes played in different sequences. From the scatter plot corresponding to this dataset (see Figure 4.1 (a)), we can see that the dataset is of lower sparsity than that of the previous one. Most of the points away from the origin align along four directions, but we see that the most of the points that align along the remaining two directions are not very far away from the origin. This affects the potential function, as the peaks at these angles are considerably smaller than the rest of the four peaks, as seen in figure 4.1 (b). Hence the threshold to be set for finding the directions $\boldsymbol{a}^j$'s should be set considerably lower to separate all the six sources. When each of the recovered signals is played, a little background sound corresponding to other signals is heard. In terms of relative error, this dataset shows the same trait as that of the previous dataset.



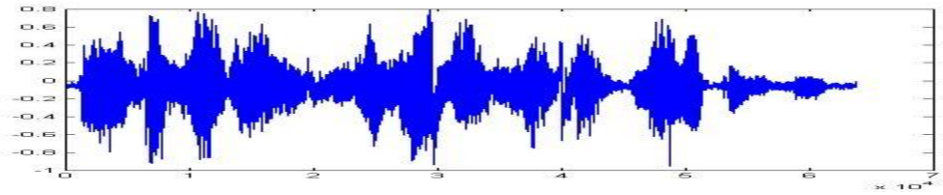
(a)



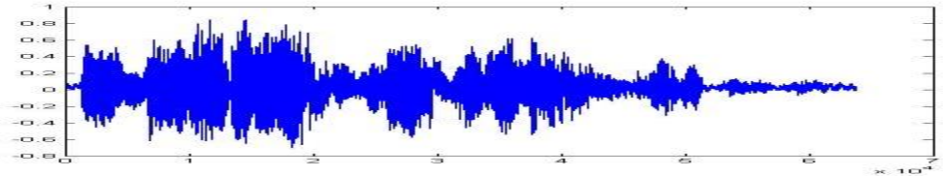
(b)

Figure 4.1: Scatter plot of row vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ for the data in the SixFluteMelodies dataset in frequency domain represented by (a) and polar plot (b) of its corresponding potential function.

After applying the algorithm on the steady sources, we move forward to test it on the dynamic sources. The result of this data is of utmost importance as most of the acoustic signals we encounter are from the dynamic sources and separation of dynamic sources has been a special interest in communication engineering. The approach was used on the data set FourVoices, a mixture consisting of recordings of voices of four different individuals mixed linearly in two different proportions. Figures 4.2 (a) and (b) show the mixed signals corresponding to this data set. Figures 4.3 (a), (b), (c), (d) show the plots of the original sources before mixing. The scatter plot corresponding to this dataset (see Figure 4.4 (a)) shows that the density of points around the direction vectors is in a more distributed fashion as compared to the previous cases. Most of the points are clustered around the origin. As a result, the potential function (see figure 4.4 (b)) does not have sharp peaks as found in the previous cases. Also, since the potential function is not smooth, down sampling and interpolation of the function was necessary in order to avoid finding the insignificant peaks, as mentioned in section 1.2. When each of the recovered signals is played, the voices in the other sources were faintly heard in the background, but only one of the voices was clearly heard in each of the recovered signals, dominating the others. In terms of relative error, this dataset shows the similar traits as that of all the previous datasets. Figures 4.5 (a), (b), (c), (d) show the plots of the recovered signals.

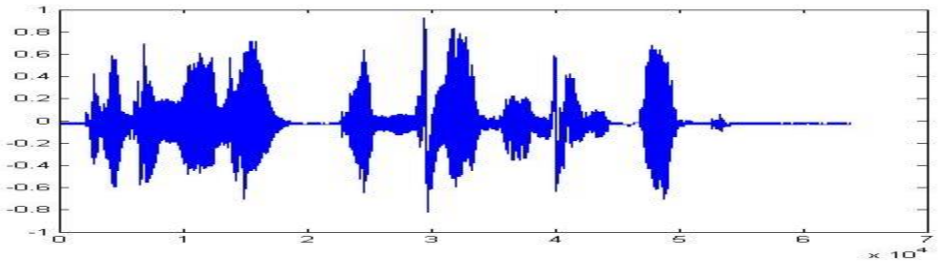


(a)

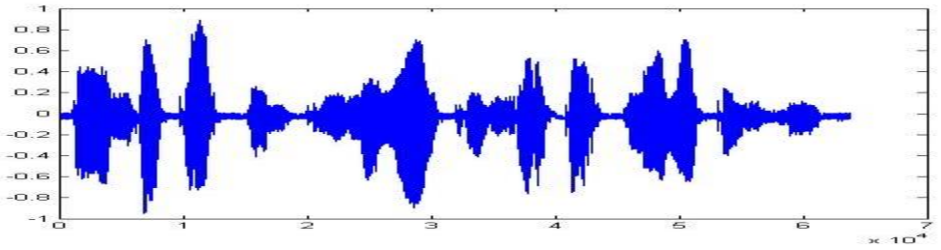


(b)

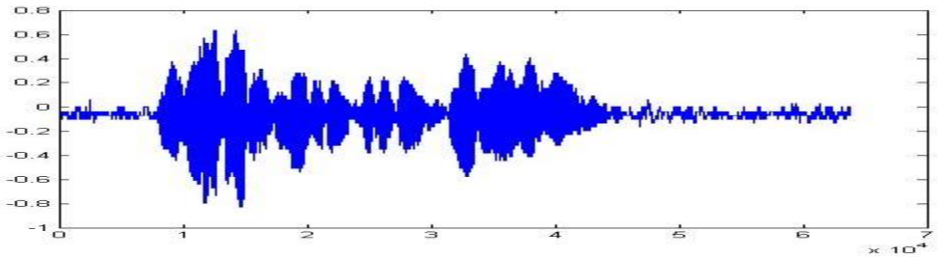
Figure 4.2: The two mixture signals (a) and (b) for the data in the FourVoices dataset



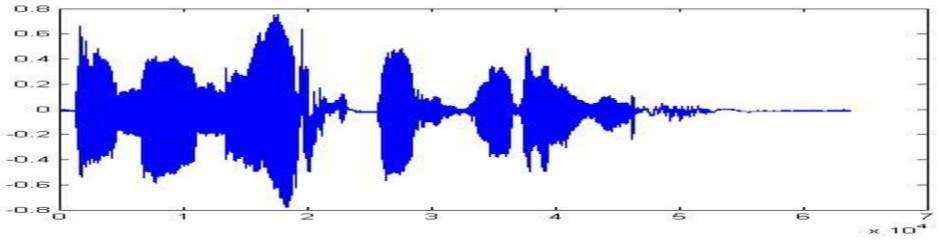
(a)



(b)

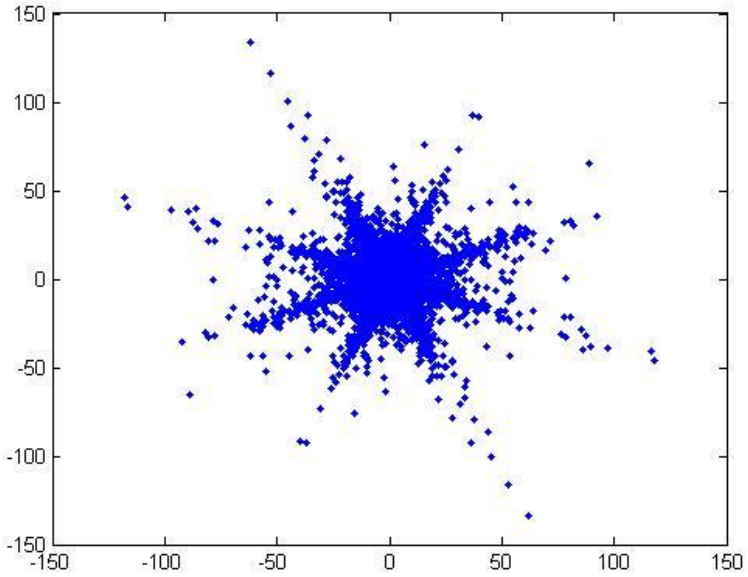


(c)

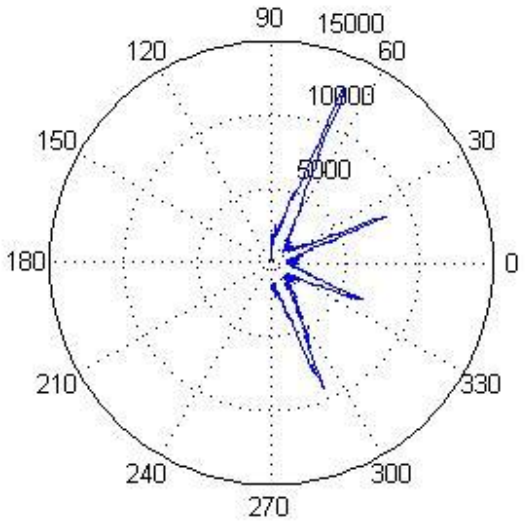


(d)

Figure 4.3: The plots original source signals (a), (b), (c) and (d) for data in the FourVoices dataset

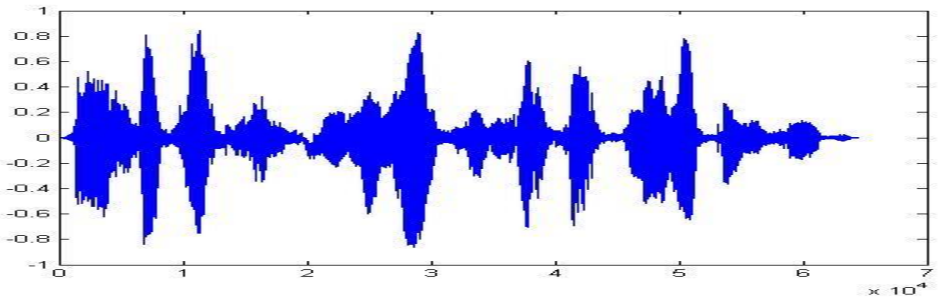


(a)

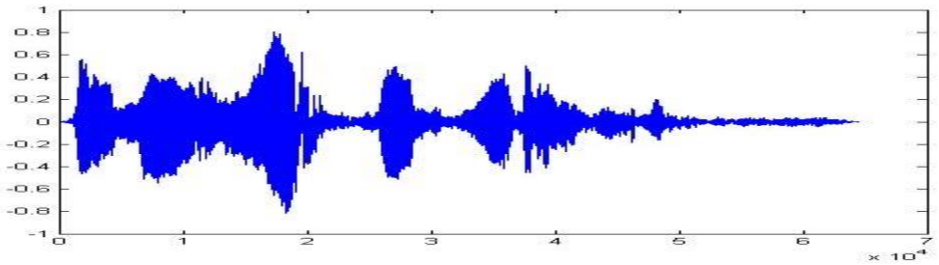


(b)

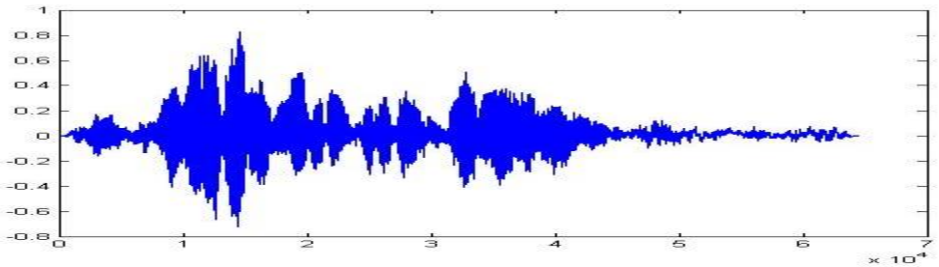
Figure 4.4: Scatter plot of row vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ for the data in the FourVoices dataset in frequency domain represented by (a) and polar plot (b) of its corresponding potential function.



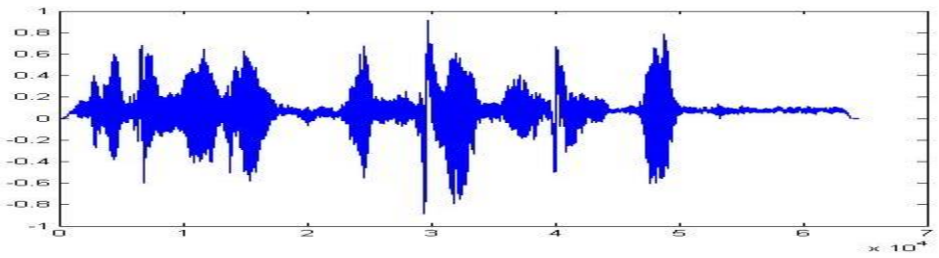
(a)



(b)



(c)



(d)

Figure 4.5: The plots of estimated source signals (a), (b), (c) and (d) for data in the FourVoices dataset

4.3. Conclusion

The introduction of the Homotopy algorithm to the approach described in [1] was observed to be successful for solving the underdetermined BSS problem. The recovered signals using this alteration in the approach also converge to the ones using l_1-minimization approach.

Since the solution to the LASSO also converge to the l_1-minimization problem, other approaches towards solving the former could be tested in this approach, since Homotopy was proved to show good results. One of such algorithms of interest is Least Angle Regression (LARS) algorithm [11], which is another approach towards solving the LASSO problem. LARS works similar to the Homotopy algorithm, but the difference lies in the ways both methods select the covariate vectors. LARS selects the covariate vectors which is most correlated with the response and move towards the direction that is equiangular to all the covariate vectors selected so far.

4.4. References

[1] P. Bofill and M. Zibulevsky, "Underdetermined blind source separation using sparse representations," Signal Processing, vol. 81, no. 11, pp. 2353-2362, Nov. 2001.

[2] T. Xu and W. Wang, "A compressed sensing approach for under determined blind audio source separation with sparse representation", Proc. IEEE Int. Workshop Statist. Signal Process, pp.493 -496 2009.

[3] Chenhao Qi, Xiaodong Wang and Lenan Wu, “Underwater Acoustic Channel Estimation based on Sparse Recovery Algorithms”, IET Signal Processing, Vol.5, No.8, pp.739-747, 2011.

[4] Chenhao Qi, Lenan Wu and Xiaodong Wang, “Underwater Acoustic Channel Estimation via Complex Homotopy”, IEEE International Conference on Communications (ICC 2012), pp. 3821-3825, Ottawa, June 2012.

[5] M.A. Davenport, M.F. Duarte, Y. C. Eldar and G. Kutyniok “Introduction to Compressed Sensing,” Book Chapter in Compressed Sensing: Theory and

Applications, Edited by Y. C. Eldar and G. Kutyniok, Cambridge University Press, 2011.

[6] M. Fornasier and H. Rauhut, "Compressive Sensing", Book Chapter in Handbook of Mathematical Methods in Imaging, Springer, 2011.

[7] C. Jutten, J. Herault, Blind separation of sources, an adaptive algorithm based on neuromimetic architecture, Signal Process. 24 (1) (1991) 1–10.

[8] A. Hyvärinen, E. Oja, "Independent component analysis: algorithms and applications", Neural Networks. Volume 13, Issues 4‑5, June 2000, Pages 411‑430

[9] Pascal Wallisch, Chapter 19 ‑ Principal Components Analysis, MATLAB for Neuroscientists (Second Edition) - An Introduction to Scientific Computing in MATLAB 2014, Pages 305‑315

[10] Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso,' J. Royal. Statist. Soc. B.58, 267–288.

[11] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004) Least angle regression, Annals of Statistics, 32, 407–451.

[12] P. Bofill, M. Zibulevsky, Sound Examples, http://www.ac.upc.es/homes/pau/.

APPENDIX A

CALCULATION OF POTENTIAL FUNCTION

We define a potential function based on the points of the scatter plot with frequency domain representation of mixed signals. The function is given by

$$G(\theta_k,\beta) = \sum_t l_t g(\beta(\theta_k - \theta_t))$$

Where, l_t is the distance of each point $\boldsymbol{x}^t$, θ_t is the angle of each point $\boldsymbol{x}^t$ and β is a scaling parameter. And, *g(α)* is a basis function. The basis function is defined such that the potential function at $\theta_k$considers the contribution from only the points around the direction θ_k. In our work, we define our basis function as

$$g(\alpha) = \begin{cases} 1 - \dfrac{4\alpha}{\pi} & for\ |\alpha| < \pi/4 \\ 0 & elsewhere \end{cases}$$

The number of points θ_k is taken based on the resolution of the potential function required. We define

$$\theta_k = \frac{\pi}{2K} + \frac{k\pi}{K} - \frac{\pi}{2} \text{ for } k=1,2,\ldots,K,$$

Where, the resolution of the potential function is dependent on K. The parameter β has a huge effect on the potential function. The number of maxima of the potential function is hugely dependent on β. Thus the proper choice for the parameter β depends on the dataset itself. In the datasets, we use β =55, which works well with all the datasets used so far, and we set K=1080.