Guided Matching Pursuit and its Application to Sound Source Separation

Dimitrios Zantalis

Doctor of Philosophy

University of York

Electronics

March 2016

Abstract

In the last couple of decades there has been an increasing interest in the application of source separation technologies to musical signal processing. Given a signal that consists of a mixture of musical sources, source separation aims at extracting and/or isolating the signals that correspond to the original sources. A system capable of high quality source separation could be an invaluable tool for the sound engineer as well as the end user. Applications of source separation include, but are not limited to, remixing, up-mixing, spatial re-configuration, individual source modification such as filtering, pitch detection/correction and time stretching, music transcription, voice recognition and source-specific audio coding to name a few.

Of particular interest is the problem of separating sources from a mixture comprising two channels (2.0 format) since this is still the most commonly used format in the music industry and most domestic listening environments. When the number of sources is greater than the number of mixtures (which is usually the case with stereophonic recordings) then the problem of source separation becomes under-determined and traditional source separation techniques, such as "Independent Component Analysis" (ICA) cannot be successfully applied. In such cases a family of techniques known as "Sparse Component Analysis" (SCA) are better suited. In short a mixture signal is decomposed into a new domain were the individual sources are sparsely represented which implies that their corresponding coefficients will have disjoint (or almost) disjoint supports. Taking advantage of this property along with the spatial information within the mixture and other prior information that could be available, it is possible to identify the sources in the new domain and separate them by going back to the time domain. It is a fact that sparse representations lead to higher quality separation. Regardless, the most commonly used front-end for a SCA system is the ubiquitous short-time Fourier transform (STFT) which although is a sparsifying transform it is not the best choice for this job. A better alternative is the matching pursuit (MP) decomposition.

MP is an iterative algorithm that decomposes a signal into a set of elementary waveforms called atoms chosen from an over-complete dictionary in such a way so that they represent the inherent signal structures. A crucial part of MP is the creation of the dictionary which directly affects the results of the decomposition and subsequently the quality of source separation. Selecting an appropriate dictionary could prove a difficult task and an adaptive approach would be appropriate. This work proposes a new MP variant termed guided matching pursuit (GMP) which adds a new pre-processing step into the main sequence of the MP algorithm. The purpose of this step is to perform an analysis of the signal and extract important features, termed guide maps, that are used to create dynamic mini-dictionaries comprising atoms which are expected to correlate well with the underlying signal structures thus leading to focused and more efficient searches around particular supports of the signal. This algorithm is accompanied by a modular and highly flexible MATLAB implementation which is suited to the processing of long duration audio signals. Finally the new algorithm is applied to the source separation of two-channel linear instantaneous mixtures and preliminary testing demonstrates that the performance of GMP is on par with the performance of state of the art systems.

Contents

List of Figures

List of Tables

I dedicate this work to my parents, Alexandra and Xenofontas

Acknowledgements

The completion of my PhD has been a long journey with a lot of thrills and chills, and would not have been possible without the help and support of several people within and without the University of York. First of all, I would like to thank my supervisor, Dr. Jeremy J. Wells, for giving me the opportunity to pursue this PhD and also for introducing me to the fascinating world of sparse signal decompositions and representations. His continuous encouragement, support and guidance and his willingness to meet regularly and have extensive discussions about my research, have been major contributing factors to the completion of this work. I would like to thank my second supervisor, Mr. Anthony Tew, for all the constructive discussions we had through the years and for being an excellent teacher and colleague from whom I learned a lot. I would also like to thank my thesis advisor, Dr. John Szymanski, for his advice on PhD related matters and for all the constructive feedback and challenging questions during our TAP meetings.

A heartfelt thank you to Dr. Julian Miller, for being an excellent teacher, colleague, friend and mentor and for his genuine caring and concern about my dissertation progress and my well-being. I would also like to thank Camilla Danese and Helen Fagan for always helping me out and keeping me on track with my administration duties and for being helpful and patient with all my last minute requests and questions. A special thank you to my dear friends and colleagues, Jingbo Gao and Theofilos Petsas. Thank you for making the bad times good and the good times amazing. Life in York wouldn't have been the same without you guys! We made it and we are still sane (I think)! I would also like to thank my very good friend Lefteris Angelidis for his support, motivation, encouragement, inspiration and all the crazy times throughout the years.

A big thank you to my special friend Karen Hoyland. Thank you for your love, support, encouragement, and all the good memories. Thank you for being there for me throughout this journey, listening to my worries and problems and helping me out in any way you could. Thank you for your patience and for putting up with my quirky behavior and abnormal working hours, especially during the last year; I know it wasn't easy. A lot has changed since we first met but no matter what, I want you to know that I will be forever grateful for having met you and having you in my life for these last four years. I would also like to thank Laura Hoyland and Diogo Martins for all the good times we had during the past few years and for their hospitality. I thank you from the depths of my heart.

A huge thank you to my best friend, mentor and sponsor, my brother, Dr. Nikolaos Zantalis. I made it bro and it wouldn't have been possible without your support. I love you.

Finally, I would like to thank my parents, Xenofontas and Alexandra who are the sole reason for me being here today, writing these words. Thank you for your love, moral support and encouragement throughout my academic years and for always believing in me and letting me chase my dreams no matter the cost. I love you both.

Declaration

I, Dimitrios Zantalis, declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

The work presented in this thesis references the following related publication:

1. Zantalis D. and Wells J.J., "Semi-Blind Audio Source Separation of Linearly Mixed Two-Channel Recordings via Guided Matching Pursuit", *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, 2014

Chapter 1

Intoduction

The problem of blind source separation (henceforth BSS) was first formulated in the early 1980s, in the framework of neural modeling, while researching the problem of motion decoding in vertebrates [1]. Since then, the theory of BSS has evolved significantly with contributions emerging from various scientific fields such as signal processing, statistics, computing and machine learning. Similarly, the application spectrum of BSS is very wide and source separation methods have been applied to several scientific fields including biomedicine, communications and acoustics.

In general, the BSS problem appears in multi-sensor systems such as, for example, electrode arrays in electroencephalography, antenna arrays in communications and microphone arrays in acoustics [1]. The output of such systems is a set of mixture signals comprising a combination of the underlying sources and can be either processed and transmitted in real-time or stored in a medium for post-processing and later reproduction. The main objective of source separation methods is to retrieve the unobserved original sources given the mixture signals.

This dissertation focuses on audio applications of BSS, with emphasis on the separation of stereo music signals. A system capable of separating sources from audio mixtures would be an invaluable tool to audio engineers and end users alike. Although there are currently separation systems that perform well on certain synthetic mixtures, separation of real-world audio mixtures (such as produced music) is very difficult to achieve. This application domain is considered a "hot" topic and is widely researched.

The rest of the chapter discusses in more detail the problem of sound source separation. In particular, section 1.1 introduces several applications of sound source separation with a focus on music signals and section 1.2 provides a mathematical formulation of the problem. In section 1.3 stereophonic sound is introduced and common microphone configurations with their corresponding mixing models are discussed. Section 1.4 presents several available approaches to the solution of BSS. In section 1.5 the scope and assumptions of this dissertation are provided. Finally section 1.6 provides an overview of the remaining chapters.

1.1 Applications of Sound Source Separation

The problem of source separation has been extensively researched in the fields of acoustics and audio signal processing. This can be partly justified by the fact that most acoustic signals of interest are mixtures of several sound sources, so clearly there is a great interest in developing systems that are able to separate such mixtures.

Probably one of the easiest to understand applications of source separation to sound, relates to the well known "cocktail party effect" [2]. This phenomenon refers to the ability of the human auditory system to discriminate and focus on a sound stimulus of interest, in the presence of other competing sound stimuli, as is the case in a cocktail party, where an individual can focus on a single conversation within a noisy environment. A system capable of separating a target sound source from a mixture of several sources can be used in hearing aid devices with the aim to improve the listening experience of the hearing impaired individual.

In a similar context to the "cocktail party problem", sound source separation can be applied to the general field of machine audition (or computer audition) which deals with the automatic analysis and understanding (i.e. extraction of meaningful information) from audio signals. The task of enabling a machine to "listen" is highly complex and its solution is based on a combination of methods, techniques and algorithms, stemming from various research fields such as signal processing, auditory modeling, perception and cognition, pattern recognition, machine learning and psychology [3]. It is clear that source separation forms part of this field and in practice a sound source separation algorithm could be used as the front-end to a machine audition system. The idea is that a complex sound scene should be first separated into its individual components making any subsequent processing easier and semantically relevant.

Another domain, in which sound source separation is widely applied, is music. Musical signals are produced by the combination of several instruments (i.e. sound sources) usually played in unity. Most commercially produced music is recorded and distributed in stereo (i.e. two-channels) format. This process is depicted in figure 1.1 where four sound sources (clarinet, violin, soprano and viola) are recorded by a stereo microphone configuration and mixed down into a two-channel mixture signal via the use of a mixing desk. Given such mixtures it is often desirable to extract or isolate certain instruments. A system capable of separating individual sources from a musical mixture can lead to novel post-processing techniques but also improve the quality of traditional processing tasks. Some possible applications are the following:

- Removal of lead vocals to allow someone to perform karaoke, that is, sing along the rest of the instruments.
- Up-mixing of a stereo recording, making the recording suitable for reproduction over different system formats (e.g. 2.0 to 5.1 up-mixing).
- Spatial re-configuration, allowing arbitrary positioning of instruments within a mix, after the recording has taken place.
- High quality mash-ups where different instruments from several music pieces can be blended together to create new compositions.

- Development of high-quality music transcription systems where each instrument is transcribed individually instead of as a mixture.

- Single source modification, where a particular audio effect can be applied to and affect a single source only, instead of the whole mixture.

- High quality time-stretching and pitch shifting processing, where different algorithms are applied to each source depending on its characteristics (e.g. harmonic and percussive parts are treated differently).

Music Information Retrieval (MIR) is another field that could benefit from the application of source separation. MIR is enganged with the classification, organisation, searching, browsing and retrieval of audio content within very large multimedia databases [4]. Applications of MIR include, but are not limited to, music retrieval [5, 6], music recommendation systems [7, 8] and music playlist generation [9]. A common step in any MIR system is the analysis of audio signals and extraction of features (e.g. timbre, melody, rhythm, harmony, structure, spectral flux, onset detection, beat extraction etc.). In such analysis, polyphonic audio content is typically treated as a whole and the extracted features represent a sum of the information across all instruments/voices [4]. In this case, source separation could be used to segregate a polyphonic mixture into individual music parts which can be then treated separately.

Yet another possible application domain of sound source separation, is that of *Computerised Lung Sound Analysis* (CLSA), which deals with the detection and classification of pulmonary (i.e. relating to the lungs) sounds with the aim of diagnosing different respiratory conditions [10]. Typical CLSA systems use specialised microphone arrays for the recording of pulmonary sounds and software for their analysis and automatic classification. The signals recorded though, usually contain interfering sounds, such as heart sounds and background noise which can hinder the detection and classification of pulmonary sounds. A sound source separation system can be used as the front-end to the software part of a CLSA system where sounds of interest are isolated and processed. Surprisingly and as far as the author knows, source separation algorithms have not been used in this context and this can be easily verified by reading relevant literature on surveys of CLSA systems, as for example in [10, 11].

It should be clear from the discussion so far that sound source separation can be applied to a wide variety of problems and can either lead to novel processing methods or in many cases enhance the quality of results of already established methods. It should be noted, that given the variety of sound signal types and contexts they occur, acoustics is one of the most complex application fields of source separation and no single algorithm (or even a family of algorithms) is appropriate for all cases. The solution to the source separation problem heavily depends on the application at hand, the characteristics and properties of the mixtures and sound sources and finally the mixing process involved. The following section presents a mathematical formulation of the source separation problem.

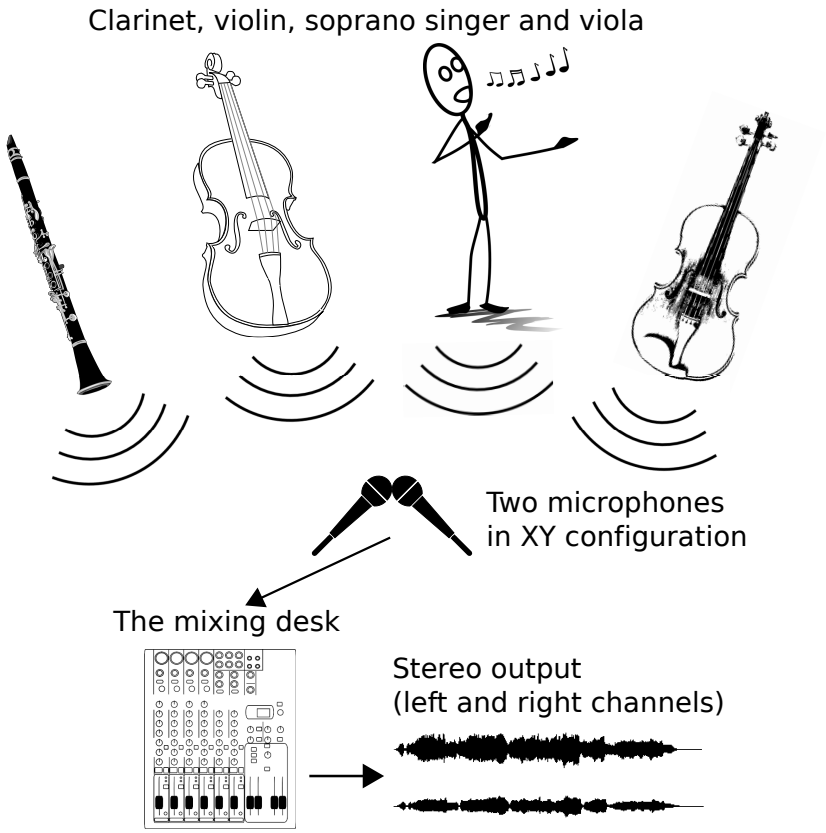


Figure 1.1: Four instruments, a clarinet, violin, soprano and viola (from left to right) are recorded by a pair of microphones and mixed down to a stereo signal via the use of a mixing desk. This recording and mixing process is typical both in live and studio environments.

1.2 Problem Formulation

Source separation deals with the recovery of unobserved source signals from a set of mixture signals which are usually obtained by multi-sensor systems such as electrode arrays, antenna arrays and microphone arrays. When both the sources and the mixing process are unknown, then the problem is known as *blind source separation* (BSS) [1]. In case the signals carry audio information, then the problem is referred to as *blind audio source separation* (BASS) [12]. Finally, when prior information about the mixture or the sources are known (such as number of sources, score information, mixing information etc.), then the problem is considered to be *semi-blind*.

In BSS, the mixing process of the unobserved sources is traditionally described by a noise-free [1] mixing model of the form [1]:

$$\mathbf{x}(t) = \mathcal{A}\{\mathbf{s}(t)\} \tag{1.1}$$

where $\mathbf{x}(t) = [x_1(t), \ldots, x_M(t)]^T$ is an $M \times 1$ column vector of the observed signals (i.e. *mixtures*) at time instance n, $\mathbf{s}(t) = [s_1(t), \ldots, s_P(t)]^T$ is a $P \times 1$ column vector of the unobserved signals (i.e. *sources*) at time instance n, T denotes the transpose operator (details about linear algebra notation are given in section 2.1.8) and $\mathcal{A}\{\cdot\}$ is a linear mixing operator, which corresponds to a chosen mixing model and combines P sources into M mixtures.

[1] In this dissertation, noisy mixtures are not considered. Although noisy observations are often encountered in real-world measurments, in many cases the statistical distribution of noise is unknown and as such it is treated as a nuisance paramater. Generally speaking, noisy models are more complex and difficult to treat than noise-free models. In practice, mixtures are usually pre-processed to reduce the precense of noises or methods found to be invariant to noise can be applied to mixtures containing weak noises.

In particular, most BSS literature, deals with two main linear mixture models. The first one is the linear instantaneous model described by:

$$x_m(t) = \sum_{p=1}^{P} a_{mp} s_p(t), \quad \forall m \in \{1, \ldots, M\} \tag{1.2}$$

where a_{mp} are scalar *mixing coefficients*. This model can also be written in vector-matrix notation:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) \tag{1.3}$$

where $\mathbf{A} \in \mathbb{R}^{M \times P}$ is known as the *mixing matrix* and contains the mixing coefficients a_{mp}.

The model described by equation (1.3) treats the mixtures and sources as random variables and the observed and unobserved values, $\mathbf{x}(t)$ and $\mathbf{s}(t)$ respectively, are realisations of these random variables at time instance n. Assuming that we have N realisations of M variables, then each set $x_m(t)$ for $m = 1, \ldots, M$ and $n = 1, \ldots, N$, is a sample of one random variable [13], thus the linear instantaneous mixing model can be written as:

$$\mathbf{X} = \mathbf{A}\mathbf{S} \tag{1.4}$$

where $\mathbf{X} \in \mathbb{R}^{M \times N}$ is a matrix containing the mixture signals (also known as the *data matrix* or *data set*), $\mathbf{S} \in \mathbb{R}^{P \times N}$ is a matrix containing the source signals and $\mathbf{A} \in \mathbb{R}^{M \times P}$ is the mixing matrix as in equation (1.3). Equations (1.3) and (1.4) express the same process, that is the process of linear instantaneous and noise-free mixing and both are often encountered in literature.

The second mixing model, which is better suited to real-world signals, is the linear convolutive model described by:

$$x_m(t) = \sum_{p=1}^{P} \sum_{k=0}^{K-1} a_{mp}(k) s_p(n-k), \quad \forall m \in \{1, \ldots, M\} \tag{1.5}$$

in which case $a_{mp}(k)$ are finite impulse responses of (causal) mixing filters that usually describe the transfer function between the sound sources and the receiving sensors and K is the length of the mixing filters. In the context of music production the mixing filters can also describe the synthetic mixing effects used in the production (e.g. panning, reverb/echo, equalisation etc.) [12].

The main objective of BSS is to recover the source signals by using only the observed mixtures. In case $\mathcal{A}$ is invertible, then the condition $P \leq M$ is often necessary (which leads to determined ($P = M$) or over-determined ($P < M$) mixtures) and BSS methods try to find a "separating" operator $\mathcal{B}$ such that:

$$\mathbf{y}(t) = \mathcal{B}\{\mathbf{x}(t)\} \tag{1.6}$$

where $\mathbf{y}(t)$ is an estimate of the vector $\mathbf{s}(t)$, $\mathbf{x}(t)$ is the vector of mixtures and $\mathcal{B} = \mathcal{A}^{-1}$ is the inverse operator of $\mathcal{A}$. Note that in these cases, estimation of the operator $\mathcal{A}$ or its inverse $\mathcal{B}$, directly leads to source separation [1].

In case $P > M$ (i.e. more sources than mixtures) then $\mathcal{A}$ is not invertible and the mixing problem is referred to as under-determined. Under this condition, source separation is achieved

by a two-step process: estimation of the mixing operator $\mathcal{A}$ followed by source restoration. A main problem in under-determined mixtures, is that, even if the mixing operator is known, there are infinite solutions to the separation problem and in order to achieve a unique solution further assumptions about the sources are required [1].

In any case, trying to recover the sources given just the mixtures is an ill-posed problem, therefore additional assumptions about the sources are required. In practice, for any source separation method to achieve meaningful results, it is usually necessary that [1]:

- the sources satisfy certain assumptions such as statistical independence, sparsity, positivity, etc.
- the underlying physical mixing model that leads to the separating system is correct.

Different assumptions about the sources and the mixing model lead to different families of solutions, the most common of which are discussed in section 1.4. The next section briefly discusses several microphone configurations that are commonly used in practice and their corresponding mixing models.

1.3 Stereophonic Sound

Stereophonic sound is a method of reproducing sound that conveys spatial information to the listener. Spatial cues that can be reproduced include source directionality, distance, depth and width and environment envelopment and spaciousness. This is in contrast to monophonic reproduction where spatial cues can only be distance and depth provided by reverberation. Stereophony can refer to any sound system that reproduces sound with spatial information. Although multiple channel systems (3.0, 5.1, 7.1 etc.) are widely used today (especially for film and game sound) music is still produced for a 2.0 system which is the most common one, found in most domestic listening environments.

Stereo sound became available to the public in the late 1950s and was a consequence of Alan Blumlein's patent in 1931 [14]. Blumlein showed that amplitude differences between a pair of loudspeakers correspond to phase differences between the ears similar to those produced in natural listening. In 1958 a paper by Clark, Dutton and Vanderlyn [15], based on Blumlein's idea, showed in more detail how amplitude differences between two loudspeakers connect the original position of a source and the phantom image of that source produced on reproduction. Blumlein's patent led to the well known "Blumlein Pair" stereo recording technique which provides excellent imaging accuracy, but over a limited angle and for frequencies below approximately 1 kHz (where the inter aural phase difference mechanism of the human auditory system plays a role in perceiving directional sound). Another approach to stereo sound was the "Decca Tree", developed in the early 1950s by Decca Records. This is a spaced configuration of omnidirectional microphones, producing a sound with poorer imaging accuracy than the Blumlein-pair technique but, according to some engineers, greater sense of spaciousness.

The aim of recording and reproducing stereo sound is a heavily debated subject with no clear answer. As already discussed, various techniques have been proposed. Some people believe that the aim of recording classical music and similar genres for example, is to reproduce the event in such a way so that the listener will perceive the performance as if he was present at the event, that is, create the illusion of "being there" , hence relaying the sound of the natural environment. Others believe that the reproduction of recorded material should sound different from its natural environment. And of course there are cases, such as with most popular music and TV sound, where a natural environment does not exist so the spatial reproduction of the recording is decided by the producer. It is clear that the way to record and reproduce an event depends on the type of the event and in some cases on the personal taste of the sound engineer/producer/listener.

1.3.1 Stereo Recording Techniques

Typically a stereophonic recording uses two microphones (sensors) to capture the audio sources and specifically the spatial information of the sources, so that they can be reproduced via a two loudspeaker system. In general, space in stereo recordings is represented by differences between the two channels (hence between the signals reproduced at each loudspeaker). If there are no differences between the signals contained in each channel then the representation is monophonic (and the listener hears a source in front of him). The differences between the two channels can be:

1. amplitude differences (coincident-pair microphones)

2. time differences (spaced microphones)

3. combinations of amplitude and time differences.

There are two main families of recording techniques: coincident pair, and spaced microphone configurations, each with pros and cons regarding spatial reproduction. Other techniques have been devised, that are a compromise between the coincident-pair and spaced microphone configurations, usually referred to as near-coincident. The subsequent sections briefly discuss the aforementioned recording techniques.

Coincident Pair Microphone Configurations

The coincident pair uses two directional or bidirectional microphones placed as close to each other as possible and angled over a range of settings. In this configuration the spatial position of sound sources is encoded in the level differences between the outputs of the microphones. Since the microphones are placed as close as physically possible, no phase differences (time differences) are produced at the recorded signals. As a consequence, coincident pair recordings have a lesser sense of space (narrow stereo image) and depth when compared to spaced pair techniques [16]. On the other hand coincident pair recordings tend to have very good source imaging accuracy [16]. There are various coincident pair configurations, using microphones with different directional patterns and with different angles between the microphones. Some of the most common configurations include the traditional "Blumlein Pair", the XY configuration and the MS (mid-side) configuration.

Spaced Pair Microphone Configurations

Spaced pair configurations, also known as AB configurations, use two identical microphones spaced a few feet apart both aiming straight at the sound stage [16]. The microphones used are usually omnidirectional but this is not a strict requirement; microphones with other polar patterns can be used, although omnidirectional microphones are favoured because they tend to have a flatter and more extended frequency response [16]. In this configuration spatial information is encoded mainly in the difference of the time of arrival of a source at the two microphones and to a lesser extent on the level differences between the microphones. Spaced microphone arrays are based on the precedence effect, a psychoacoustic effect, which occurs when two sounds separated by a short time delay are heard, in which case the brain localises the sound towards the earlier of the two sources. It has been shown that delays of about 0.5 to 1.5 ms are required to shift a phantom image fully left or fully right [16]. This time delay also dictates the spacing of the microphones in spaced arrays which should not be too wide. Time differences produced by the microphones spacing translate into phase differences between the signals when reproduced at the two loudspeakers making the signals uncorrelated to each other which results in a greater sense of envelopment and spaciousness which is preferred by some listeners. This lack of phase coherence though also leads to less accurate off-centre phantom images which can be considered as a disadvantage, depending on the material that is reproduced. A famous spaced pair configuration is the "Decca Tree".

Near Coincident Microphone Configurations

Near coincident or closely spaced techniques use two directional microphones angled as in an XY configuration and with the capsules spaced a few centimetres apart. These configurations combine the principles of coincident pair and spaced pair configurations. In this case, spatial information is encoded using both time and level differences thus the reproduced sound has good ambience and precise image positioning. Many near-coincident configurations have been suggested, usually named after the organisation that first used them. Some famous configurations include the "ORTF pair" (Office de Radiodiffusion-Television Francaise) and the NOS stereo technique (Nederlande Omroep Stichting).

1.3.2 Mixing Models

Mixing is the process where multiple source signals are combined into one or more channels. In stereo music (2.0 format), the recorded audio signals are *mixed down* (i.e. combined) into two channels. The way signals are mixed down on each channel is referred to as the *mixing model*. Mixing models are directly related to the recording techniques discussed in the previous section. The process of mixing can be distinguished for two cases: two microphone recordings and multi-microphone recordings. When two microphones are used for a recording, the mixture of the source signals is the direct output of the microphones, with each microphone output corresponding to a channel and the mixing model is dictated by the recording technique that was used. In this case mixing of the source signals is instantaneous.

In multi-microphone recordings many mono microphones are used to record each source separately. In this case mixing is performed by the mixing engineer using a mixing desk; each source is fed to a channel in the mixing desk, where the levels, panning and time differences can be controlled individually to produce the final two-channel mix. This is the case in most studio productions where each instrument is recorded individually, possibly at separate times and mixing occurs at a later stage. In this situation the mixing model is decided by the engineer.

The following mixing models are often encountered:

1. two-channel panning
2. delay based mixing
3. panning and delay based mixing
4. convolutive mixing.

A similar distinction was presented in [17].

Two Channel Panning

In this model, a mono signal is fed through a *pan control* and is split into two signals each going into a different channel. The pan control effectively controls the amount of the signal that is fed to each channel. By introducing intensity (i.e. level) differences between the channels, sources can be positioned in space, in a similar fashion as with coincident-pair microphone configurations where intensity differences between the microphones encode the spatial position of the sources. Mixing signals using level differences can be expressed mathematically as (considering a single mono source):

$$x_m[n] = \alpha_m s[n], \quad m = 1, \ldots, M \tag{1.7}$$

where $s[n]$ is the source signal, α_m are the *mixing coefficients* (or mixing weights), $x_m[n]$ is the produced mixture and M denotes the number of mixture channel, which for a stereo mixture is $M = 2$. Note that in this simple case the mixing coefficients are simple gain factors, that is, they only affect the amplitude of the source. The relationship between the mixing coefficients is dictated by a *panning law*. The simplest relationship is the *linear panning law* where the mixing coefficients can be calculated as:

$$\alpha_m = \begin{cases} 1 - \beta', & m = 1 \\ \beta', & m = 2 \end{cases} \tag{1.8a}$$

$$\text{with } \beta' = \frac{\beta + 1}{2}, \quad -1 \leq \beta \leq 1 \tag{1.8b}$$

where β is the panning position. Inspecting equation (1.8b) it can be seen that as β changes from -1 to 1 (fully left to fully right) then β' changes from 0 to 1 so α_1 fades out linearly form 1 to 0 and α_2 fades in linearly from 0 to 1. This is depicted in the left sub-plot of figure 1.2. The linear panning law is not ideal because when $\beta = 0.5$ (i.e. the source is panned to the centre) then $\alpha_1 = \alpha_2 = 0.5$, which will make the signal sound weaker in the middle.

To avoid this problem the *constant power law* can be used instead. In this case the mixing coefficients can be calculated by:

$$\alpha_m = \begin{cases} \cos(\beta'), & m = 1 \\ \sin(\beta'), & m = 2 \end{cases} \tag{1.9a}$$

$$\text{with } \beta' = \frac{\pi(\beta + 1)}{4}, \quad -1 \leq \beta \leq 1 \tag{1.9b}$$

In this case as β changes from -1 to 1, β' changes from 0 to $\frac{\pi}{2}$. Again the mixing coefficients change from 0 to 1 but in this case the transition is not linear. The advantage of using this law is that the sum of the powers of the channels is constant since $\cos(\beta')^2 + \sin(\beta')^2 = 1$ for any value of β'. The constant power law is depicted in the right sub-plot of figure 1.2. The plots in figure 1.2 were produced by directly evaluating equations (1.8a) and (1.9a) for appropriate values of β' (equations (1.8b) and (1.9b) respectively).

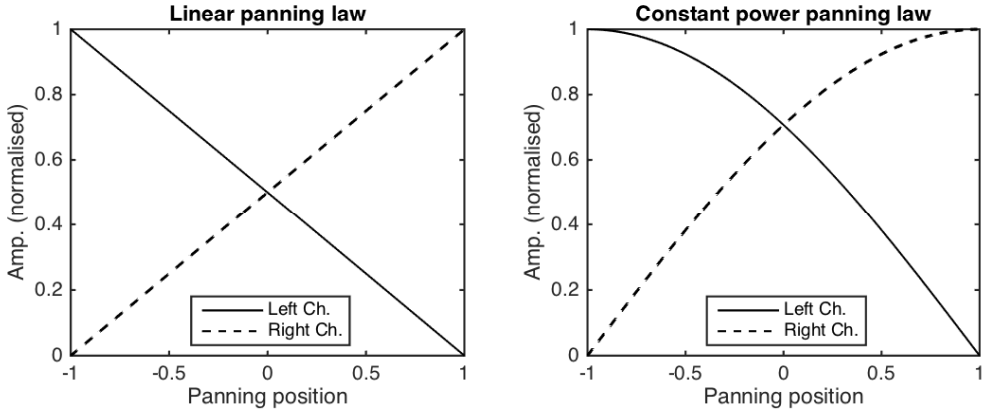


Figure 1.2: Two-channel panning laws.

The above treatment was for one source mixed into two channels. For multiple sources equation (1.7) changes to equation (1.2).

Delay Based Mixing

This mixing model is based on the time differences between the two channels. Positioning of a source in space is achieved by delaying one channel with respect to the other. The principle of this model is the same as in some spaced-pair microphone configurations and is based on the precedence effect. This situation is depicted in figure 1.3. One main assumption in this model is that the source is in the far field which implies that the wave-fronts reaching the microphones are approximately planar. This approximation can be used to express the delay δ between the channels as:

$$\delta = \frac{d \sin(\theta)}{c} \tag{1.10}$$

where c is the speed of sound, d is the distance between the microphones and θ is the angle created by the source and the line bisecting the line between the microphones.

In the case where the delay δ is known or estimated and d and c are also known, then the angle of the produced phantom image can be estimated by:

$$\hat{\theta} = \arcsin\left(\frac{c\hat{\delta}}{d}\right) \tag{1.11}$$

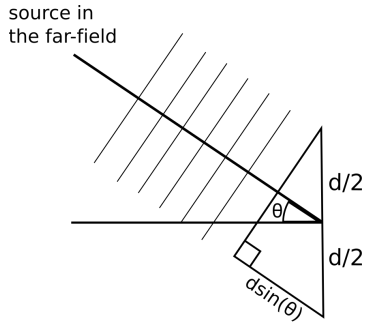


Figure 1.3: Delay based panning.

Panning and Delay Based Mixing

This model is a combination of the previous two. It is directly related to near-coincident microphone setups where both differences in amplitude and in time are captured by the microphones. Panning and delay based mixing are two techniques that are heavily used in electronic music productions.

Convolutive Mixing

Convolutive mixing during recording arises in cases where the microphones capture not only the direct path of the sources but delayed copies of the sources as well, generated by reflections of the sound from different objects. During post-recording, convolutive mixing refers to the addition of artificial reverberation to the direct source signals. This model is expressed mathematically by equation (1.5) and is the most generic, as it can be used to describe any of the aforementioned mixing models. Note that all aforementioned models are noise free but there are cases when a noise term has to be included. For the rest of this dissertation only noiseless mixtures are assumed.

Table 1.1 provides a summary of all mixture types that have been discussed so far. A similar distinction is provided in [12] with the addition of *reverberant* mixtures which exhibit a more realistic reverbaration time (compared to convolutive mixtures which imply relatively short mixing filters) and *time-invariant* or *time-varying* mixing filters which are constant or slowly-varying over time, respectively.

Mixture type	**Meaning**
Over-determined	More mixtures than sources ($M > P$)
Determined	Same number of mixtures and sources ($M == P$)
Under-determined	More sources than mixtures ($P > M$)
Instantaneous	Simple gain mixing filters
Anechoic	Simple gain & delay mixing filters
Convolutive	Non trivial mixing filters

Table 1.1: Various mixture types encountered in practice.

1.4 Previous Work On Source Separation

Source separation of acoustical signals is a very complex task. The literature on sound source separation is vast and many solutions have been proposed from different fields of study, including statistics, signal processing and computing. Broadly speaking, most solutions derive ideas and are based on two main families of techniques: "Blind Source Separation" (BSS) and "Computational Auditory Scene Analysis" (CASA). These are discussed in the following subsections.

1.4.1 Blind Source Separation (BSS)

As already explained in section 1.2, the problem of BSS in its initial form is ill-posed and in order to achieve a meaningful solution, additional assumptions about the sources and the mixing model are required. In particular, let us assume the linear instantaneous model described by equation (1.4), then the following solution classes can be derived:

- The assumption of statistical independence of the sources (i.e the rows of $\mathbf{S}$) leads to the "*Independent Component Analysis*" (ICA) problem which is discussed in section 1.4.2.
- The assumption that the sources are sparse (i.e. having only few significant coefficients) leads to the "*Sparse Component Analysis*" (SCA) problem which is discussed in section 1.4.3.

1.4.2 Independent Component Analysis (ICA)

According to a historical review in [1], ICA was initialy introduced by C. Jutten in 1987 and later formalised for linear mixtures by P. Comon [18]. It was originally developed to deal with problems similar to the "cocktail party problem" mentioned in section 1.1. The basic ICA problem assumes a linear instantaneous and noise free model (equation (1.3)) with an unknown mixing matrix $\mathbf{A}$ and unobserved sources $\mathbf{s}(t)$. The mixing matrix is assumed to be square, which implies an equal number of sources and mixtures (i.e. the determined case) and also both mixtures and sources are assumed to be zero-mean. ICA methods are also based on the following two necessary assumptions:

- The unobserved sources $s_p(t)$ are assumed to be statistically independent and identically distributed. This is known as the i.i.d. assumption.
- The sources must have non-Gaussian distributions (which are usually unknown).

Statistical independence of sources might seem like a very strict assumption, but in reality it holds in many cases. The authors in [19] demonstrated that audio sources have high dependencies over short durations which, in many cases, decrease with increasing signal duration and also demonstrated that speech and music sources exhibit similar dependencies on average. These claims contradict the claims in [20, 21] and the general belief that music signals exhibit larger dependencies than speech signals due to the fact that musicians play in a coherent way (musical sounds are played synchronously and in harmony). Informally, two random variables s_1 and s_2 are said to be independent if information about a value of s_1 does not reveal any information

about s_2 and vice versa [22]. Mathematically, independence is defined using probability densities. In particular two random variables are independent if and only if their *joint probability density function* (*pdf*) is factorisable [22], which can be expressed as:

$$p(s_1, s_2) = p_1(s_1)p_2(s_2) \tag{1.12}$$

where $p(s_1, s_2)$ is the joint *pdf* of s_1 and s_2 and $p_1(s_1)$, $p_2(s_2)$ are the marginal *pdfs* of s_1 and s_2 respectively. The above definition can be extended to any number of random variables.

The non-Gaussianity of the sources is also a very important assumption and without it the ICA estimation is not possible. This assumption is a concequence of the "Central Limit Theorem" which states that, under certain conditions, the distribution of the linear combination of several independent random variables tends to be Gaussian. Following this, the joint pdf of Gaussian random variables will be Gaussian and completely symmetric and as such, it will not contain any information on the mixing directions of $\mathbf{A}$ [13]. In other words, if the underlying sources have Gaussian distributions, then the mixing matrix cannot be estimated and ICA cannot be applied.

By taking advantage of the aforementioned assumptions, ICA methods attempt to estimate the mixing matrix $\mathbf{A}$ by minimising some dependency measure between the sources (also known as contrast functions), essentially treating the problem as a numerical optimisation problem [22]. After estimating the mixing matrix, its inverse $\mathbf{W}$ is computed and finally the unobserved sources are obtained by:

$$\mathbf{y} = \mathbf{W}\mathbf{x} \tag{1.13}$$

where $\mathbf{y}$ are estimates of the unobserved sources, $\mathbf{W}$ is the inverse of $\mathbf{A}$ and $\mathbf{x}$ are the observed mixtures. It should be noted that in this case, the estimated sources are equal to the original sources up to a scale and a permutation [18]. By observing equation (1.2), it can be seen that, since both $\mathbf{s}$ and $\mathbf{A}$ are unknown a scalar multiplication of a source s_p could be canceled by dividing all mixing coefficients $a_{mp}, \forall m \in \{1, \ldots, M\}$ by that same scalar [22]. This leads to the scaling ambiguity. Similarly, since both $\mathbf{s}$ and $\mathbf{A}$ are unknown, then the summation terms in equation (1.2) could have any order with any of the independent components being the first one [22]. This leads to the permutation ambiguity. Formally, ICA leads to an estimated demixing matrix $\mathbf{W}$ that satisfies $\mathbf{WA} = \mathbf{PD}$ where $\mathbf{P}$ is a scaling and permutation matrix and $\mathbf{D}$ is a diagonal matrix [1].

In the past three decades, a lot of contrast functions have been proposed for the solution of the ICA problem. Some solutions try to maximise a quantitative measure of non-Gaussianity of the sources, such as kurtosis [23, 24, 25, 26, 27], differential entropy and its equivalent negentropy [28, 29, 30] or approximations of the latter [31]. Other solutions, inspired by information theory, try to minimise the mutual information between the sources which is equivalent to the Kullback-Leibler divergence; a natural measure for independence [32, 22, 33]. Another popular approach is to treat ICA as a maximum likelihood estimation problem [34, 35, 36], which is closely connected to mutual information and the "infomax principle" [37, 38, 39]. There are several ICA algorithms that are based on the optimisation of the aforementioned criteria, such as *InfoMax* [38], *JADE* [40] and *FastICA* [41, 22].

Basic ICA focuses on determined instantaneous mixtures and achieves good separation performance, given that the mixing matrix is invertible. In real environments, however, the observed mixtures are usually convolutive, as described by equation (1.5) and the ICA problem becomes more difficult since the elements of the mixing matrix are impulse response filters rather than simple scalars, as is the the case with instantaneous mixtures. There are two main approaches to the solution of this problem. The first approach, known as time-domain BSS, applies ICA directly to the convolutive mixtures and tries to estimate time-domain demixing filters [42, 43, 44]. Although this approach achieves good separation results when the algorithm converges, it is computationally expensive, especially when long mixing filters are involved. The second approach, known as frequency-domain BSS, addresses the problem in the frequency-domain [45, 46, 47, 48, 49]. In this case, the short-time Fourier transform (STFT) is applied to the time-domain mixtures and complex-valued instantaneous ICA algorithms are applied separately to each frequency band, that is, the convolutive ICA problem reduces to many instantaneous ICA problems for each frequency. This approach is appealing, since the complexity of the ICA problem is reduced (from convolutive to instantaneous) thus the computational time is reduced significantly, by employing fast ICA algorithms with parallel computation for multiple frequencies. Also any complex-valued instantaneous ICA algorithm can be employed. However, in this case, the permutation ambiguity of the ICA solution becomes a serious problem since the estimated sources can be arbitrarily ordered in every sub-band. This is known as the permutation problem of frequency-domain BSS and several solutions have been proposed [45, 46, 50, 51].

Despite the success of ICA, the majority of the proposed methods are only suitable for the solution of determined (or over-determined) BSS problems. When the mixtures are under-determined, then generally, the mixing matrix is not invertible and there are infinite solutions to the problem. In this case, further assumptions about the sources are required. One assumption that has proved very useful to the solution of under-determined BSS, is that of sparsity of the sources. This assumption leads to a BSS solution class known as *Spare Component Analysis* which is discussed in the next section.

1.4.3 Sparse Component Analysis (SCA)

SCA is largely based on sparse decomposition techniques which underwent considerable development during the 1990's and have been successfully applied to under-determined BSS [52, 53, 54, 55, 56, 57, 58]. Sparse decompositions are very attractive because they provide a relatively simple framework for extracting sources that exceed the number of mixtures and also improve greatly the quality of separation in the determined case [1]. The main assumption is that sources have disjoint temporal supports or have disjoint supports in a suitable domain, which is different from the usual assumption of statistical independence used in most ICA algorithms. The following explanation is based on the simplified model suggested by Van Hulle [59] and expressed mathematically in [1].

Considering the model described by equation (1.3), with M mixtures and P sources, then the main steps of SCA techniques are estimation of the mixing matrix $\mathbf{A}$ followed by extraction of the sources $\mathbf{s}(t)$ given only the mixtures $\mathbf{x}(t)$. Assuming that at every point t only a single source is active (or a single source is much more active than the rest) then the temporal support [2] of

[2] Mathematically, a support of a function is the set of points of that function that have non-zero values. Temporal support in this context refers to the support of the signal (which can be thought as a function) in the time domain.

the p^{th} source can be denoted as $\Lambda_p \in \{1, \ldots, T_{total}\}$ (i.e. the set of points where p^{th} source is more active) and $\forall t \in \Lambda_p$ it can be said that $|s_l(t)| \gg |s_q(t)|$ for $l \neq q$ and therefore:

$$\mathbf{x}(t) \approx s_p(t)\mathbf{A}_p, \quad t \in \Lambda_p \tag{1.14}$$

where $\mathbf{A}_p$ is the p^{th} column of the mixing matrix. The set of points $\mathbf{x}(t) \in s_p(t)$, $t \in \Lambda_p$ will be almost aligned along the straight line passing through the origin and directed by vector $\mathbf{A}_p$. This can be clearly seen on the scatter plot of column A) in figure 1.4 (left column of plots). The first three plots of column A) in figure 1.4 depict the time domain waveforms of three speech signals, $s_1(t)$, $s_2(t)$ and $s_3(t)$, which are disjoint in time. Their disjoint temporal support can be clearly seen in the plots; the signals are never active at the same time which is in accord with the main assumption stated above. The two plots below, depict the waveforms of the two mixtures, $x_1(t)$ and $x_2(t)$, produced by the linear combination of the sources. The source signals were mixed using the constant power law discussed in section 1.3.2, so that they appear at positions $-30°$, $0°$, and $30°$ respectively when reproduced over loudspeakers (these values correspond to the mixing matrix $\mathbf{A} = [0.780.500.21; 0.210.500.78]$). The final plot of column A) depicts a temporal scatter plot produced by the set of points of the pair values $(x_1(t), x_2(t))$ for all t samples. In the scatter plot three distinct lines crossing the origin can be clearly seen. The direction of the lines are essentially the directions of the column vectors of $\mathbf{A}$ and all samples, close to each line, belong to a source signal. By using this interpretation it is possible to use clustering algorithms [60] to separate the scatter plot into P clusters and estimate the direction $\hat{\mathbf{A}}_p$ of each cluster, thus estimating the mixing matrix $\hat{\mathbf{A}}$. If the mixture is determined or over-determined the sources can be estimated using the inverse or pseudo inverse matrix respectively using $\hat{\mathbf{s}}(t) \coloneqq \hat{\mathbf{A}^{-1}}\mathbf{x}(t)$. If the mixture is under-determined and the sources have disjoint supports, they can be estimated using the least squares method by setting $\hat{\mathbf{s}(t)} \coloneqq \frac{<\mathbf{x}(t), \hat{\mathbf{A}}>}{\|\hat{\mathbf{A}}\|_2^2}$, $\forall t \in \Lambda_p$ or 0 otherwise.

This is a straightforward and attractive approach but the assumption of disjoint temporal support is very restrictive, even for speech signals and it can only be met in situations where the speakers talk in turns (and assuming that no reverberation or other background noise exists). In music, most of the time, this assumption does not hold since instruments tend to play in unity (at the same tempo and in harmony). This can be seen in the temporal scatter plot of column B) in figure 1.4 (middle column of plots). The sub-plots of column B) in figure 1.4 depict three musical signals, the two mixtures produced by linearly combining those signals and the temporal scatter plot of the mixtures. In the scatter plot it can be seen that there are no distinguishable directions as was the case for the speech signals of column A). The sub-plots of column C) in figure 1.4 (right column of plots) depict the GMP coefficients of the three music sources, the two mixtures and a scatter plot of the mixture coefficients. GMP stands for *guided matching pursuit* and is a variant of *matching pursuit* which is an sparse decomposition technique. The specifics of sparse representations and matching pursuits will be discussed in detail in chapter 2 and GMP in chapters 3 and 4. Looking at the scatter plot of the GMP coefficients it can be seen that the mixing matrix directions are distinguishable. In this new representation, it is now possible to estimate the mixing matrix directions by clustering the coefficients (or other methods), separate the sources in the new representation and finally restore the separated sources by reversing the GMP decomposition, a process that is referred to as the *synthesis step*.

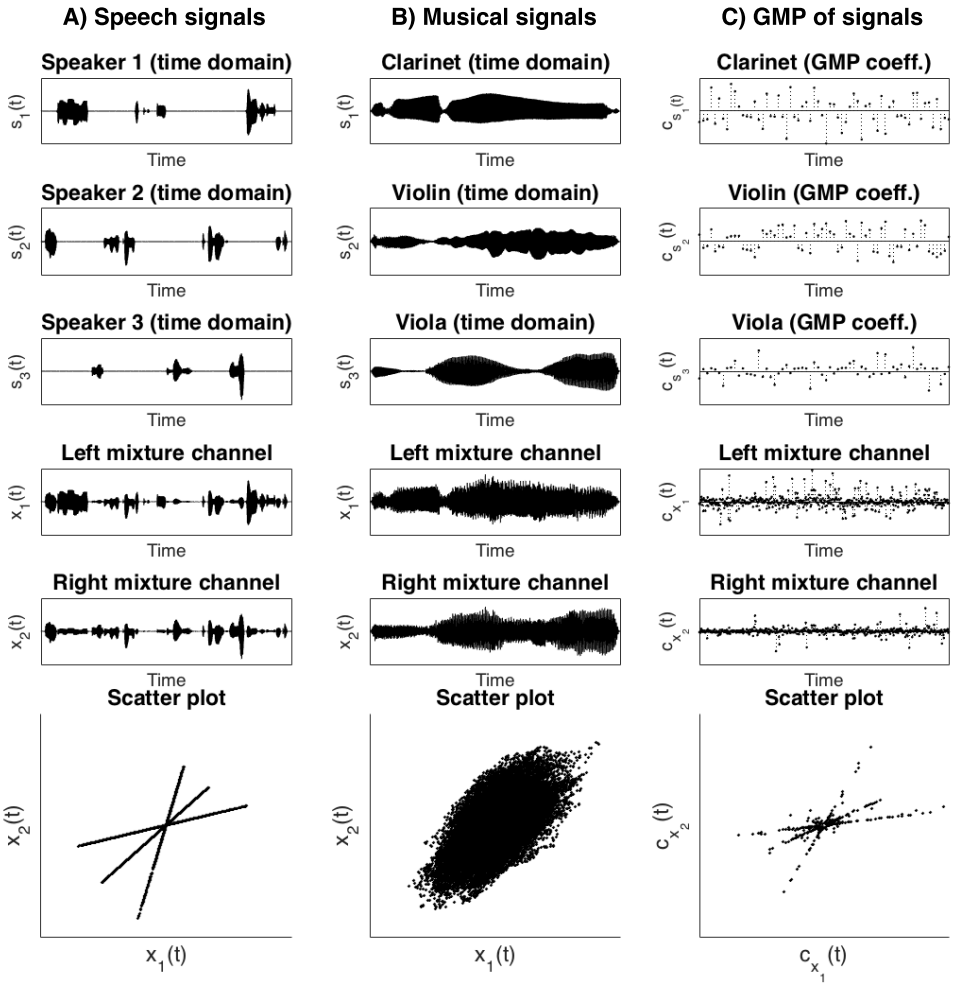


Figure 1.4: Basic principle of source separation using sparse decomposition. The first three sub-plots of each column represent the sources, the following two sub-plots linear mixtures of the sources and the bottom sub-plots depict the temporal scatter plots of the mixtures. A) shows that sources with disjoint temporal supports lead to mixtures that produce scatter plots where the coefficients are aligned along the columns of the mixing matrix. B) shows that mixtures of more complex signals (such as music sources) do not produce legible scatter plots. C) shows that when an appropriate decomposition (GMP in this case) is applied to the music signals (the same signals as in B)) then the column directions of the mixing matrix become visible in the scatter plot of the mixtures.

The results of the GMP coefficient scatter plot can be explained by looking at the GMP coefficients of the mixture signals. It can be observed that the coefficients have almost disjoint supports. This is a very interesting observation and is a consequence of sparse decomposition techniques which decompose a time domain signal (time domain for audio, spatial domain for images etc.) into another domain where the signal is sparse, that is, only a few coefficients are of significant amplitude (or equivalently most of the coefficients are zero or close to zero). This sparsity is the main reason why the coefficients in the new domain have disjoint (or almost disjoint) supports and is a fundamental assumption in SCA-based techniques. Note that similar results can be obtained by using other sparsifying transforms instead of GMP, such as the *short-time Fourier transform* (STFT) or the *modified discrete cosine transform* (MDCT). Sparse decompositions are the basis of SCA-based techniques and also the basis of the source separation algorithm that is proposed in this dissertation.

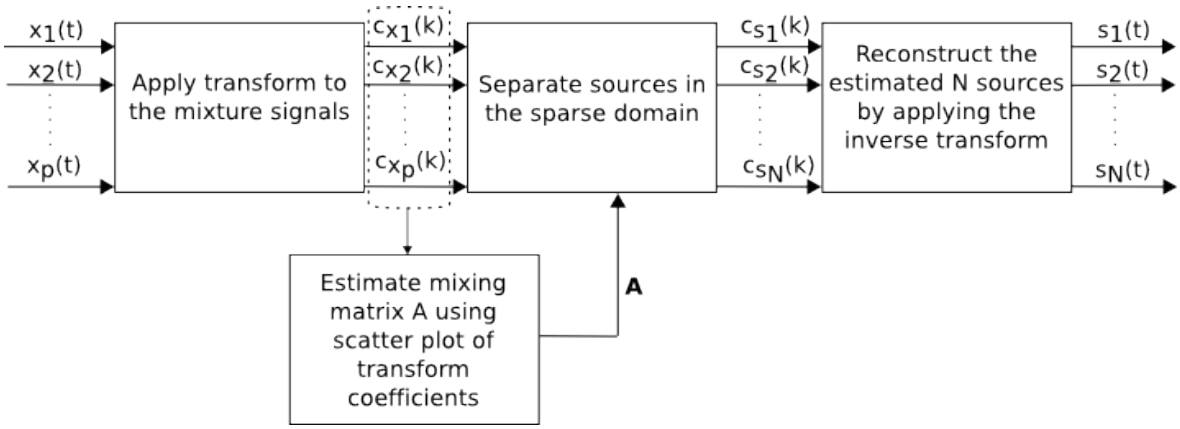


Figure 1.5: Block diagram of SCA-based source separation system. The figure was adapted from [1].

The majority of sparsity-based source separation methods follow, to a large extent, the aforementioned procedure which is depicted in figure 1.5. To summarise, the main steps of a SCA-based system are the following:

1. Transformation or decomposition of the observed mixture signals into a domain where the sources can be sparsely represented and the coefficients are expected to have disjoint (or almost disjoint) supports.

2. Estimation of the column directions of the mixing matrix.

3. Separation of the sources in the new representation domain.

4. Reconstruction of the estimated sources using an inverse transform or the synthesis step of a decomposition.

This first step of a SCA-based system, deals with sparse signal representations which is a huge field of study and extends beyond source separation. In the context of under-determined BSS, sparsity is useful because it enables both the estimation of the mixing matrix and the separation of sources. There are several methods that lead to sparse representations such as linear transforms (STFT [61], MDCT [62], wavelet transforms [63] etc.), methods based on ℓ^τ norm minimisation (Basis Pursuit [64], FOCUSS [65, 66], iterative thresholding algorithms [67] etc.), greedy methods (matching pursuit and variants [68, 69]) and Bayesian methods based on *maximum likelihood (ML)* and *maximum a posteriori (MAP)* estimation [70, 71, 72]. In SCA, the choice of an appropriate sparse representation algorithm depends on several factors, such as computational complexity, selection of an appropriate dictionary and ease of algorithm parameterisation for optimal performance. Sparse representations are discussed in more detail in section 2.2.

The second step of mixing matrix estimation is very crucial in the process. Wrong estimation of the mixing matrix can lead to very poor separation performance, therefore robust algorithms are necessary. In the determined BSS case, the mixing matrix is square and conventional ICA methods can be used, although use of sparsity priors can improve the separation quality even in this case [54]. In the under-determined case there are generally two main classes of methods to this problem. The first one is based on a Bayesian interpretation, which involves the joint optimisation of the mixing matrix and the sources [73, 54]. The second class of methods is based on a geometrical interpretation [74] and involves clustering from the scatter plots produced by the

representation coefficients [20, 75, 76]. A more detailed discussion on mixing matrix estimation is given in section 5.1.

The third step of a typical SCA-based system involves the separation of the sources in the new representation domain. In the case of (over-)determined mixtures, the (pseudo-)inverse mixing matrix can be applied either directly to the time-domain mixtures or to the sparse representation of the mixtures. These two approaches are identical only if the sparse representation is an exact representation of the time-domain mixtures. In case the representation is approximate then the two approaches will produce different results. Applying the inverse matrix in the sparse representation domain is of interest in the case of noisy mixtures where the approximate solution could effectively de-noise the mixture [1]. In case the mixtures are under-determined, then separation of the sources occurs at the new representation domain either by *time-frequency masking* [58] or by ML or MAP estimation by assuming a generalised Gaussian distribution for the source coefficients [53, 54].

The final step of a SCA system is the reconstruction of the sources which is achieved by applying the inverse transform or the synthesis step of the decomposition that was used in the first step.

The sparsity-based methods discussed so far are aimed at separation of instantaneous mixtures but SCA has been applied to anechoic and simple convolutive mixtures as well. In particular the *DUET* algorithm [58] has been successfully applied to anechoic mixtures and an extension of it to echoic mixtures [77]. Another approach for under-determined convolutive BSS is presented in [78] where ICA is applied in the time-frequency domain on each frequency, followed by normalisation and clustering of the resulting basis vectors to solve the permutation problem and time-frequency masking to reduce the residuals produced by the ICA stage. Similar approaches that perform frequency-bin clustering followed by permutation alignment and time-frequency masking are presented in [79, 80].

It should be noted that although these methods have been reported to work on speech mixtures in moderate reverbaration conditions, they are based on strict assumptions about the synchronicity and activity of the sources which do not seem plausible for musical signals. Generally speaking, the problem of under-determined convolutive BSS is very challenging and far from being solved.

1.4.4 Computational Auditory Scene Analysis (CASA)

CASA methods [81] look at the problem of source separation from a psycho-acoustical perspective, that is, they try to mimic the way the human auditory system separates incoming sounds. CASA tries to separate musical sources into individual auditory objects based on psycho-acoustical cues such as harmonicity, note onsets and offsets, amplitude and frequency modulation, temporal and spectral evolution of partials and more [82]. A typical CASA algorithm usually comprises four steps: transformation of the input mixture into a suitable representation such as the correlogram or the STFT, extraction of a set of partials from that representation, re-organisation of these partials into musical sources based on certain grouping rules (such as proximity, similarity, continuity, closure etc.) and extraction of the sources using binary masking. Many approaches have been suggested including data-driven [83] and prediction driven techniques [82] with the latter producing better results [84]. A main disadvantage of CASA techniques is that they require a large amount

of training data or make strong assumptions about the sources. Techniques that use prior-information about the sources such as pitch, note locations etc. have been successful, like for example in [85] but in blind source separation such information is usually difficult to obtain or unavailable. One advantage of CASA systems is their ability to work on monophonic (i.e. single channel) audio signals.

1.5 Scope and Assumptions

It should be clear by now that sound source separation is one of the trickiest types of signal processing and is vast field. It would be very difficult to devise a separation system that is able to handle all use cases and if such a system ever comes to life it would comprise an array of algorithms and techniques. Because of the complexity of the problem we need to state some assumptions, minimise the requirements and design a system that is realisable and scalable. To that end we need to define the scope of this dissertation which could be stated as: *the application of matching pursuits to the problem of sound source separation with emphasis given on the separation of linear instantaneous stereo mixtures of multiple sources.*

The following list enumerates all the assumptions that underpin the work produced in this dissertation.

- We assume that the observed signal is a stereo mixture comprising several unobserved source signals (i.e. $M = 2P > M$). This leads to the under-determined case of source separation and as already discussed in section 1.4.3 sparse component analysis provides a suitable framework for the solution of this problem. Typical approaches involve the estimation of the mixing matrix followed by separation of the sources. A main problem with this case is that even if the mixing matrix is known it can not be used to separate the sources in the time domain since the resulting system has infinite solutions. SCA-techniques solve this problem by transforming the input signal into a domain where the sources are assumed to be sparsely represented and thus easily identifiable and separable.

- We assume that we have a known mixing matrix. Estimation of the mixing matrix is usually treated as a separate problem to recovery of sources and many algorithms have already been proposed, some providing good results for linear instantaneous mixtures. We also assume that the mixture was produced using a 'Blumlein Pair' microphone configuration which results in the linear instantaneous mixing model described by equation (1.2).

- We assume that the unobserved sources can be sparsely represented in a suitable domain. This is an important assumption that drives all SCA-based systems. Informally a signal is considered sparse if it can be approximated using only a few coefficients of significant magnitude, or in other words if most of the approximation coefficients are zero or close to zero.

Based on these assumptions a new semi-blind sound source separation system is proposed which can deal with the separation of linear instantaneous mixtures. In particular a new sparse decomposition algorithm is introduced, tested and evaluated for the solution of the aforementioned problem.

1.6 Dissertation Overview

This section provides a brief overview of the rest of the dissertation. Chapter 2 provides a detailed discussion on background theory and technologies that are relevant to this dissertation. In particular, traditional signal representations are discussed first, followed by sparse representations and finally a discussion on mathcing pursuits and their application to sound source separation. Chapter 3 talks about guided matching pursuit which is a variant of matching pursuit and is one of the main contributions of this dissertation; the guided matching pursuit algorithm is introduced and several implementation aspects are discussed in detail. Chapter 4 discusses about the design of a MATLAB® toolbox that implements guided matching pursuit and includes sections that deal with the testing, verification and performance evaluation of the newly created software. In chapter 5 the applicability of guided matching pursuit in sound source separation problems is discussed and several approaches to the problem are presented. Chapter 6 deals with the testing and evaluation of the algorithms presented in chapter 5 on a set of linear instantaneous stereo mixtures and results of these tests are presented and discussed. Chapter 7 highlights the main contributions of this dissertation.

Chapter 2

Relevant Theory & Technologies

This chapter presents theories and technologies related to the work presented in this thesis. In section 2.1 fundamental concepts of signals & systems theory and linear algebra are introduced and discussed, along with the most common representations of signals in time, frequency, time-frequency and time-scale domains. Section 2.2 describes another family of signal representation techniques called sparse decompositions and briefly discusses the pros and cons of some of the most well known methods. In section 2.3 the matching pursuit algorithm [68] is introduced and various aspects of the algorithm are discussed in detail. Finally section 2.4 presents several matching pursuit variants that deal with the problem of source separation of musical mixtures.

2.1 Signal Representations

2.1.1 Time-domain representation

Signals are quantities that carry information which describe a variety of physical phenomena. Regardless of how signals are represented, the information they carry is contained in patterns of variations of some kind [86]. For example, the patterns of variation over time of the voltage and current in an electric circuit, patterns of variations over time in air pressure (e.g speech) or patterns of variations in brightness over space and/or time (e.g. still image/video), are all different kind of signals. To demonstrate such patterns, let us focus on speech signals. Speech is produced by the human vocal mechanism via fluctuations of air pressure over time. These air pressure variations are then typically captured by a microphone and converted into an electrical signal. Such a signal is depicted in figure 2.1 where it can be seen that different speech utterances produce different patterns of air pressure variation. Figure 2.1 also depicts the most basic representation of the signal in the time domain called the waveform of the signal.

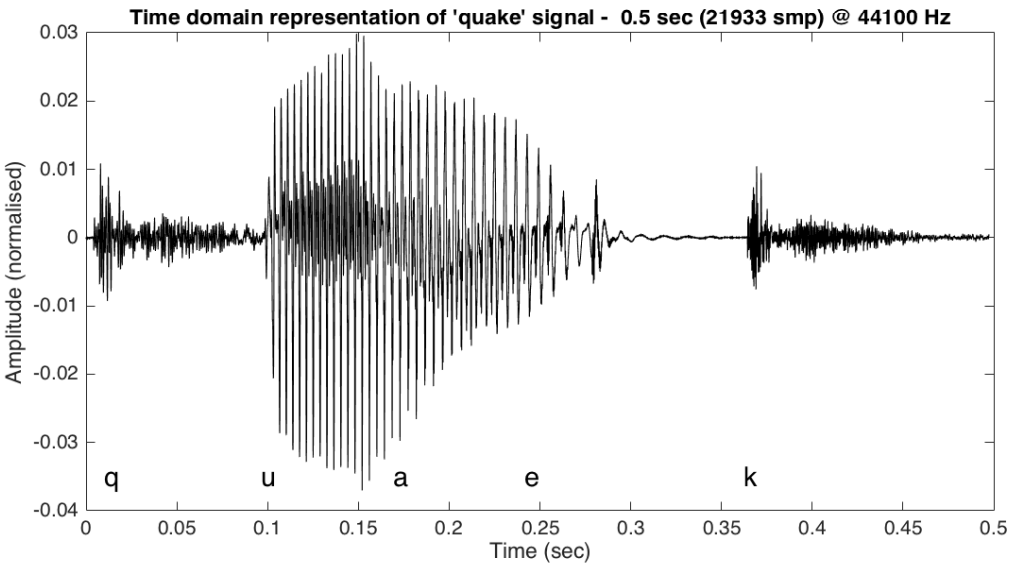


Figure 2.1: Waveform of the word 'quake' (IPA: /kweɪk/) spoken by a female speaker. The different patterns of pressure variations can be clearly seen. The x axis represents time in seconds and the y axis represents normalised amplitude.

Signals can be represented mathematically as functions of one or more independent variables [86]. As already mentioned, sound signals are variations of sound pressure over time, thus in this case the independent variable is time and the signal can be written as $x(t)$. A still image can be represented by a function of two spatial variables, $f(x, y)$ and video as a function of three independent variables (space and time) $v(x, y, t)$. The number of independent variables of the function is one way to categorise signals. Functions of one independent variable represent one-dimensional signals (1-D), with two independent variables, two-dimensional signals (2-D) and so forth.

Signal dimensionality should not be confused with the number of channels that a signal comprises. A 1-D signal can be multi-channel, as is the case with most music signals. The speech signal in figure 2.1 has a single-channel and is also referred to as a mono-signal, in contrast to stereo which implies a two-channel signal. This thesis deals with both mono and stereo, 1-D sound signals and the independent variable will always be time (t). The main techniques described here could be easily applied to 1-D signals with more than two channels as produced by devices such as microphone arrays, electroencephalography (EEG) devices and multi-channel computerised lung sound analysis (CLSA) devices.

Another categorisation, especially relevant to music signals, is based on the polyphony of their content. We have monophonic signals, which literally means single voice (or sound; originating from Greek word 'phone' which means voice), where the information of the signal is produced usually by a single instrument where only one note (mono voice) is played at any given time. Polyphonic signals (many voices) contain the sounds of a single instruments where multiple notes are played simultaneously or multiple instruments played together. The polyphony of a signal is usually a design criterion for algorithms that operate on music signals. This thesis deals mostly with polyphonic music signals.

Signals can be further classified based on the form in which they exist. We distinguish between continuous-time and discrete-time and between continuous-amplitude and discrete-amplitude signals. Thus we have four kinds of signals based on the status of time and amplitude. Signals

that are continuous both in time and amplitude are called analogue signals, whereas signals that are discrete both in time and amplitude are called digital [87]. The microphone in the previous example converts the physical signal into an analogue electric signal which can subsequently be converted into a digital signal by using an Analogue to Digital Converter (ADC). The two processes that occur in the ADC to discretise the time and amplitude of an analogue signal are called sampling and quantisation respectively. Although there exist signals that are inherently discrete-time, such as daily stock market indices, rainfall data, sunspot numbers etc, a big class of discrete-time signals stem from the sampling of continuous-time signals, and these are the signals this thesis focuses on.

Regarding the quantisation step of the ADC, in theory it is often ignored and as such we deal with discrete-time, continuous-amplitude signals. This is often the case because first of all the quantisation step is a very difficult operation for accurate mathematical analysis and secondly the quantisation error can be easily made negligible [88]. As such, and for notational purposes, throughout the thesis, the independent variable of continuous-time signals will be denoted with t and be enclosed in parentheses $(.)$ whereas for discrete-time signals, the independent variable will be denoted with n enclosed in brackets $[.]$. Also both continuous-time and discrete-time notations will be used depending on the context.

Sampling produces a discrete-time signal the values of which are samples of the continuous-time signal, sampled at equally spaced points in time. This can be defined by:

$$x[n] \stackrel{\text{def}}{=} x_a(nT_s) \tag{2.1}$$

where x_a is the continuous-time signal, T_s is a positive number in units of time and is called the sampling period (with its reciprocal, the sampling frequency, $F_s = \frac{1}{T_S}$ measured in Hz) and n is the set of integers $n = 0, \pm 1, \pm 2...$ denoting the n^{th} sample of the discrete-time sequence $x[n]$.

Note that although the abscissa of the signal in figure 2.1 is drawn using a continuous line, the signal itself is discrete-time (and discrete-amplitude also), since it was obtained from a digital copy of pre-recorded speech and as such the signal is only defined for integer values of n. It is incorrect to assume that non-integer values of $x[n]$ are zero; for non-integer values of n the signal is not defined [89]. Figure 2.2 depicts 40 samples taken from the signal in figure 2.1. From now on we will refer to that signal as the 'quake' signal. Also note that using the International Phonetic Alphabet (IPA) its phonetic notation is /kweik/.

The process of sampling at equally spaced time intervals, is a result of the sampling theorem, a property of which, states that under certain conditions, a continuous-time signal can be fully recovered by its samples [86]. In particular in order to be able to reconstruct a continuous-time signal, which is band-limited with limit B, the individual samples must be acquired by a minimum sampling period of $\frac{1}{2B}$ seconds (or $2B$ Hz). The sampling theorem is essentially the mechanism for representing continuous-time by discrete-time signals. The theoretical development, properties and historical reviews of the sampling theorem can be found in many textbooks such as [86], [87], [88] and [89].

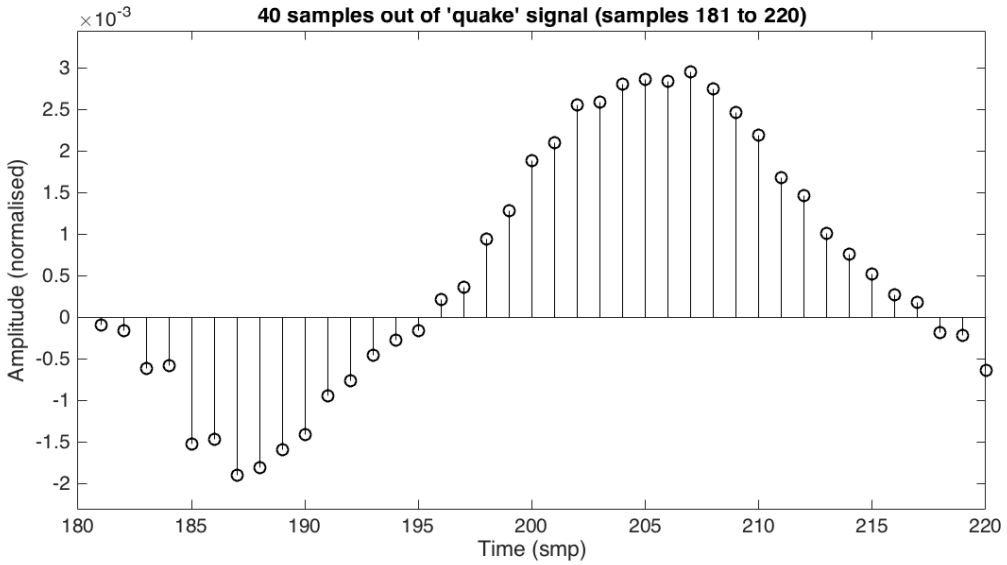


Figure 2.2: 40 samples taken from the 'quake' signal. It can be clearly seen that the signal is defined only for integer values of n (the x-axis values) and it is undefined for all other non-integer values.

A fundamental and useful discrete-time signal is the unit sample sequence, or discrete-time impulse, or simply impulse which has a value of 1 at time zero only.This is defined by [89]:

$$\delta[n] \stackrel{\text{def}}{=} \begin{cases} 0, & n \neq 0. \\ 1, & n = 0. \end{cases} \tag{2.2}$$

The importance of the impulse sequence lies in the fact that any discrete-time sequence can be written as a linear combination (i.e. weighted summation) of delayed impulses. This can be expressed as:

$$\boxed{x[n] = \sum_{k=-\infty}^{+\infty} x[k]\delta[n-k]} \tag{2.3}$$

Equation 2.3 is essentially the time domain representation of discrete-time sequences with the values $x[k]$ being the coefficients of the representation and $\delta[n-k]$ the set of elementary waveforms that are used to represent the signal. Equation (2.3) leads to the formulation of the convolution sum, which will be explained in the next subsection.

The interpretation of a sequence as a linear combination of elementary waveforms is of paramount importance. In fact, this operation is at the heart of linear algebra [90] and many mathematical treatments can be expressed using linear algebra (and functional analysis) notation. Linear algebra deals with finite-dimensional vector spaces and mappings between such spaces in contrast to functional analysis which deals with infinite-dimensional spaces. Generally speaking a vector space is a set of vectors which adhere to certain operations. Vectors can be added together and/or multiplied (scaled) by numbers (which could be real or complex) that, depending on the context, are known as scalars, weights or coefficients. These two operations combined together produce a linear combination of vectors; vectors are scaled by scalars and added (summed) together. The sound signals we deal with, can be treated as vectors in $\mathbf{R^n}$, where $\mathbf{R}$ is the vector space of real numbers and $\mathbf{n}$ the dimensionality of the space. For example the quake signal can be

treated as a vector in $\mathbf{R}^{\mathbf{N}}$, where $\mathbf{N}$ is essentially the length of the signal in samples. Relevant linear algebra notation will be introduced when necessary and appropriate throughout this and subsequent sections.

Signals can be further categorised based on their periodicity. A continuous-time signal is said to be periodic with period T if:

$$x(t) = x(t + T) \tag{2.4}$$

For example the following sinusoidal signal is periodic:

$$x(t) = Asin(\omega_0 t + \phi) \tag{2.5}$$

with period T:

$$T = \frac{2\pi}{\omega_0} \tag{2.6}$$

fundamental angular frequency ω_0:

$$\omega_0 = 2\pi f \tag{2.7}$$

amplitude A and phase ϕ. Sinusoids are another important type of elementary signals. The following figure depicts a periodic sinusoid with a frequency of 10 Hz (i.e. ten cycles per second), and a normalised amplitude of 0.8.

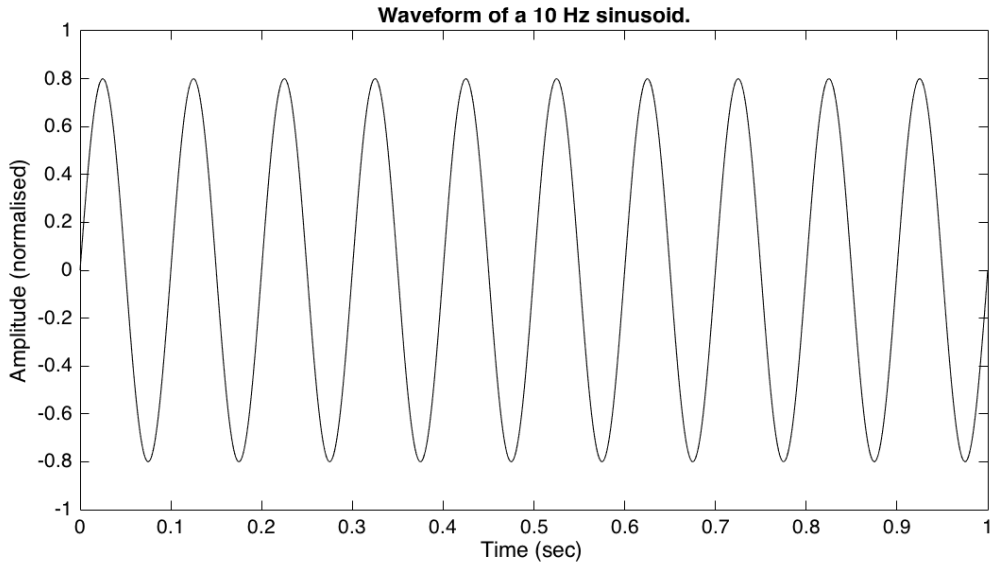


Figure 2.3: Waveform of a periodic sinusoid. The frequency is 10 Hz, that is ten full periods in one second and this can be clearly seen in the figure. The amplitude is also set to 0.8 in normalised units. This signal is periodic with a period of $T = \frac{1}{f} \longrightarrow T = \frac{1}{10} = 0.1$ sec.

We can further categorise signals into real and complex. Signals in nature are real, or real-valued, such as the 'quake' signal. When dealing with signal processing though, in many cases, it is more convenient to use complex signals. Treatments of complex signals are based on complex numbers. A complex number z can be expressed as:

$$z = a + jb, \quad j = \sqrt{-1} \tag{2.8}$$

where j is is called the imaginary unit and a and b are real numbers referred to as the real part and imaginary part of z respectively. The following notation will be used:

$$a = \Re\{z\} \tag{2.9}$$

$$b = \Im\{z\} \tag{2.10}$$

Equation (2.8) is known as the Cartesian form of the complex number z. Another more convenient form is the polar form of z expressed as:

$$z = re^{j\theta} \tag{2.11}$$

where r and θ are the magnitude and phase (or angle) of z and will be denoted by:

$$|z| = r = \sqrt{a^2 + b^2} \tag{2.12}$$

$$\angle z = \theta = \arctan\frac{b}{a}, \quad \text{for } a > 0 \tag{2.13}$$

The two forms are connected via Euler's formula:

$$e^{j\theta} = \cos(\theta) + j\sin(\theta) \tag{2.14}$$

The Cartesian to polar connection can also be interpreted geometrically as shown in figure 2.4

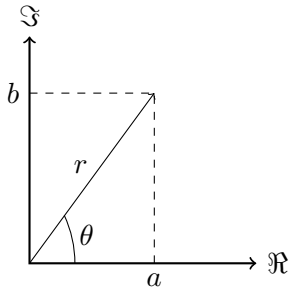


Figure 2.4: A geometrical interpretation of Cartesian to polar coordinates. The vector has magnitude r and angle (or phase) θ. Equations (2.12) and (2.13) can be calculated form this diagram using simple trigonometric identities.

The complex conjugate of z is often encountered and will be denoted by:

$$z^* = a - jb \tag{2.15}$$

Complex signals are signals where the individual samples (or coefficients) are complex. The complex number notation is a powerful mathematical model that simplifies the analysis and synthesis of signals. A very important complex signal is the complex exponential:

$$x(t) = Ae^{j(\omega_0 t + \phi)} \tag{2.16}$$

with amplitude A, fundamental angular frequency $\omega_0 = \frac{2\pi}{T}$ and phase ϕ. Using (2.14) this can be written as:

$$x(t) = Acos(\omega_0 t + \phi) + jAsin(\omega_0 t + \phi) \tag{2.17}$$

Although equations (2.16) and (2.17) are equivalent, the first one is usually preferred in mathematical treatments because it makes analysis easier and it is more compact.

Another useful categorisation depends on the symmetry of the signals. A discrete-time complex signal is conjugate symmetric if:

$$x_e[n] = x_e^*[-n] \tag{2.18}$$

and conjugate anti-symmetric if:

$$x_o[n] = -x_o^*[-n] \tag{2.19}$$

A real conjugate symmetric signal is referred to as even and a real conjugate antisymmetric signal as odd (therefore the e and o subscripts in the notation). Figure 2.5 depicts an even and odd signal.

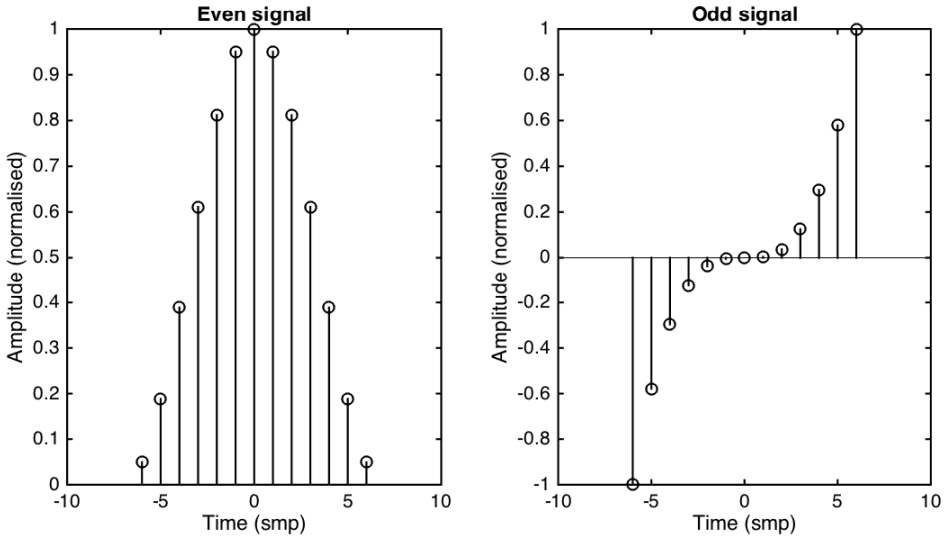


Figure 2.5: Even and odd signals. Even signals are symmetric across the y-axis whereas odd signals are symmetric across the x-axis. Odd signals are always zero at time $n = 0$ since (2.19) requires that x[0]=-x[0]

Any sequence can be written as a sum of conjugate symmetric and conjugate antisymmetric sequences:

$$x[n] = x_e[n] + x_o[n] \tag{2.20}$$

These symmetry properties play an important role when dealing with the Fourier transform which will be discussed in the next subsection.

Finally, signals can be categorised into random or deterministic. Most signals of interest are random in nature. In fact if we consider the definition of a signal which essentially states that information is found in patterns of variation of a physical quantity, and also considering the randomness of the patterns involved (as can be observed in the 'quake' signal, fig. 2.1) then we can say that in order for a signal to carry information it has to be random. The theory and application of random signals, or random processes, is studied using part of probability theory and can simplify the analysis of signals while producing fast algorithmic implementations. As such a lot of modern signal processing and especially communications, deals with random signals. Music signals can be treated as random processes but in this thesis a different approach is examined. Non-random, or deterministic signals, can be predicted at any time so they do not carry any information. Deterministic signals are used in communications as carrier signals [91]. The signal in figure 2.3 is deterministic since we know all the parameters of the model (A, ω_0, and ϕ), thus we can predict its future output at any time. When the parameters are not known and are random then the signal is random and the actual values of the parameters need to be estimated. This process is called parameter estimation and can be achieved using estimation theory [91].

Let us focus on figure 2.1 and consider all the information we have about that signal. The figure depicts a waveform of a spoken word where different patterns of variation can be seen. These patterns are produced by the human vocal mechanism and correspond to different speech utterances such as vowels, consonants, diphthongs etc. which are expressed in particular orders in time, forming larger patterns that convey a higher meaning and eventually leading to intelligible speech. We can also observe that the consonants of the word have generally a more erratic, or random, pattern of variation whereas the triphthong sound (i.e. the vowels /weI/ in 'quake'), contain some visible sinusoidal like waveforms which imply some kind of periodicity. Regarding the technical characteristics of the signal we know it is a 1-D, discrete-time, single-channel signal, sampled at 44.1 kHz with 16 bits per sample and has a duration of $N = 21933$ samples (or smp) or $T = 0.49$ seconds (or sec). We also know that it is a real-valued, random signal and can compute its energy using the following equation:

$$\boxed{E_x \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} |x[n]|^2} \tag{2.21}$$

Finally we know the meaning of the information contained within the signal (quake as in earth-quake). Considering how basic the time-domain representation is, this is an impressive amount of information we have accumulated by just inspecting the waveform of the signal and reading some meta-parameters (i.e. sampling rate, bit resolution and number of channels). But the time-domain representation alone is not enough to extract all the valuable information that exists within a signal. Before moving into other signal representations it would be helpful to discuss some fundamentals of system theory first, in order to introduce necessary notation and provide the foundations for concepts that will follow.

2.1.2 Systems

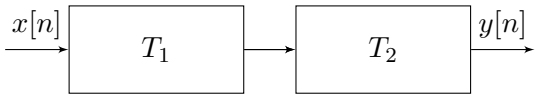
In signal processing, signals are often processed by systems. In this context: "a system can be viewed as a process where input signals are transformed by the system or cause it to respond in some way, resulting in other output signals" [86]. A continuous-time system deals with continuous-time inputs and outputs and a discrete-time system with discrete-time sequences. Mathematically, a discrete-time system can be viewed as a transformation or an operator that maps an input sequence $x[n]$ into an output sequence $y[n]$ [89]:

$$y[n] = T\{x[n]\} \tag{2.22}$$

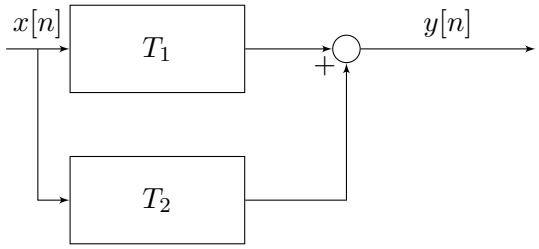
For the purposes of this thesis we are interested in discrete-time systems. In a more general sense, systems can be thought of as the interconnection of components, devices and subsystems [86]. A few basic interconnection configurations that are often encountered are the series or cascade interconnection, parallel interconnection and feedback interconnection. These configurations are shown in figure 2.6 and are called block diagrams. More complex system interconnections can occur by mixing these three basic configurations.

Like signals, systems can also be classified based on their properties. There are systems with or without memory (memoryless). A system is memoryless if the output $y[n]$ for every value of n

Series or cascade interconnection



Parallel interconnection



Feedback interconnection

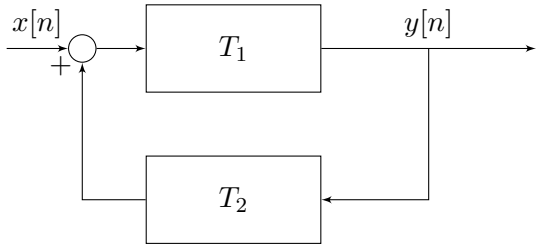


Figure 2.6: Often encountered system interconnections.

depends only on the input $x[n]$ at that same value n. For example:

$$y[n] = |x[n]|^2 \tag{2.23}$$

Another memoryless system is the identity system denoted by:

$$y[n] = x[n] \tag{2.24}$$

The identity system is also a lossless system, in an energy sense. That is, if $y[n]$ is the output of the identity system, with input $x[n]$ and for $0 \leqslant n < N-1$ with N being the length of the input and output sequences, then

$$E_y = E_x \equiv \sum_{n=0}^{N-1} |y[n]|^2 = \sum_{n=0}^{N-1} |x[n]|^2 \tag{2.25}$$

A couple of systems with memory are the accumulator:

$$y[n] = \sum_{k=-\infty}^{n} x[k], \quad \text{or by a change to the upper limit} \tag{2.26a}$$

$$y[n] = \sum_{k=-\infty}^{n-1} x[k] + x[n] \Rightarrow y[n] = x[n] + y[n-1] \tag{2.26b}$$

and the ideal delay:

$$y[n] = x[n - n_d], \quad -\infty < n < \infty \tag{2.27}$$

where n_d is an fixed integer representing the delay of the system. If n_d is zero then the delay system becomes the identity system, and thus memoryless. A positive n_d outputs a delayed input signal (the input is shifted to the right by n_d samples) whereas a negative n_d outputs a time advanced input signal (the input is shifted to the left by n_d samples).

Systems can be causal (or nonanticipative [86]) if the output depends only on current and past values of the input. For a system to be implemented in real time it must be causal. On the other hand, non-causal systems depend on future values of the input as can be realised with previously recorded signals like speech. A system can also be recursive, having a structure similar to the feedback interconnection of figure 2.6 where the output might depend on current and past times of the input but past values of the output as well.

Another important property is stability. A system is said to be stable if a bounded input results in a bounded output [89]. More formally this can be expressed as:

$$\begin{aligned} &\text{If } |x[n]| \leq B_x < \infty \quad \text{for all } n \quad \text{(an input bounded by } B_x\text{)} \\ &\text{and } |y[n]| \leq B_y < \infty \quad \text{for all } n \quad \text{(an output bounded by } B_y\text{)} \\ &\qquad \text{then the system is stable.} \end{aligned}$$

If the above relations hold then we can say that the "system is stable in the BIBO (bounded input bounded output) sense" [89].

A very important class of systems are linear, time-invariant (LTI) systems. A system is said to be linear if it obeys the superposition principle. The superposition principle comprises two properties; the additivity property and the homogeneity or scaling property which can be expressed respectively:

$$T\{x_1[n] + x_2[n]\} = T\{x_1[n]\} + T\{x_2[n]\} = y_1[n] + y_2[n] \tag{2.28a}$$

$$T\{ax[n]\} = aT\{x[n]\} = ay[n] \tag{2.28b}$$

where $x_1[n]$ and $x_2[n]$ are the inputs to system T, $y_1[n]$ and $y_2[n]$ the responses of the system to respective inputs and a is an arbitrary constant. The superposition principle can be expressed in general form as:

$$T\{ax_1[n] + bx_2[n]\} = aT\{x_1[n]\} + bT\{x_2[n]\} \tag{2.29}$$

with a and b being arbitrary constants.

A system is time-invariant (or shift-invariant) if a time shift in the input sequence results in an identical time shift in the output sequence. That is, if $x_1[n] = x[n - n_0]$ is the input to a time-invariant system then the output will be $y_1[n] = y[n - n_0]$ for every n_0. Important classes of systems are either LTI or linear but possibly time-varying.

An important characteristic of LTI systems is the impulse response, which is the response of the system when the input is the unit impulse (eq. (2.2)). Let us define the impulse response of an LTI system as:

$$h[n] \stackrel{\text{def}}{=} T\{\delta[n]\} \tag{2.30}$$

Using the time-invariance property of LTI systems, equation (2.30) can be written as:

$$h[n-k] \stackrel{\text{def}}{=} T\{\delta[n-k]\} \tag{2.31}$$

Let us assume that the input $x[n]$ to an LTI system is equation (2.3) which is a generic representation of a signal via linear combinations (superposition) of delayed impulses. The output will be the following:

$$y[n] = T\{x[n]\} \Rightarrow y[n] = T\{\sum_{k=-\infty}^{+\infty} x[k]\delta[n-k]\} \tag{2.32}$$

and by using the linearity property of the system we can write:

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]T\{\delta[n-k]\} \tag{2.33}$$

by substituting $T\{\delta[n-k]\}$ according to (2.31) we get:

$$\boxed{y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]} \tag{2.34}$$

or in short hand notation:

$$y[n] = x[n] * h[n] \tag{2.35}$$

Equation (2.34) is called the convolution sum for discrete-time LTI systems and the right hand side of the equation is the convolution operator between the sequences $x[k]$ and $h[n-k]$. For continuous-time LTI systems it is called the convolution integral and is expressed in a similar manner to (2.34). This result is very important since it implies that any LTI system is completely characterised by its impulse response, that is the response of the system when the input is the unit impulse (eq. (2.2)). In other words we can completely describe the output of an LTI system for any input as long as its impulse response is known.

It should be noted that the convolution sum as expressed in (2.34) is an explicit algorithm for programming a discrete-time linear system. The convolution sum could be programmed as:

Algorithm 1 Convolution sum

1: Obtain sequence $h[n-k]$ by reversing $h[k]$ at $k=0$ and then delay the reversed sequence by n samples
2: Multiply sample by sample the sequences $x[k]$ and $h[n-k]$ for $-\infty < k < \infty$ to obtain $g[k] = x[k]h[n-k]$
3: Sum all samples in $g[k]$ to form output sample $y[n]$
4: Shift $h[-k]$ to a new position (i.e. increase n) and repeat from step 2

An informative view is to describe the convolution operation in terms of "sliding" the response sequence $h[n-k]$ over $x[k]$ [86]. The convolution operation is directly associated with filters, that is LTI systems that perform a filtering operation on the input sequence. When dealing with filters the impulse response is also called the filter kernel or simply, kernel. The kernel essentially dictates the kind of filtering that the system will perform. Designing such kernels is known as filter design.

A similar operation to convolution is the cross-correlation of two discrete-time sequences, which can be expressed as:

$$\boxed{y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n+k]} \tag{2.36}$$

or in short hand notation:

$$y[n] = x[n] \star h[n] \tag{2.37}$$

Programmatically the difference is that the impulse response is not reversed. Observing equations (2.34) and (2.36) it can be seen that the operations of convolution and cross-correlation are related by:

$$x[n] * h[-n] = x[n] \star h[n] \tag{2.38}$$

It should be noted that if the impulse response is real and even and the input is also real, then according to equation (2.18) $h[n] = h[-n]$ and the cross-correlation operation is equal to convolution. Cross-correlation can be interpreted as a measure of similarity between two sequences. In fact the main operation of equation (2.36) is the inner product of two sequences which is the actual measure of similarity; the cross-correlation performs the inner product operation for all n thus it can be viewed as a sliding inner-product. In the context of linear algebra, the inner-product has a geometric interpretation which will be discussed in a later section. Cross-correlation leads to the idea of matched filtering, where the impulse response is effectively a template sequence which is continuously matched against the input sequence, resulting in an output that spikes at times where the template sequence matches the input signal. Higher correlations, imply high similarity between the template and the input and this leads to higher magnitude spikes in the output. This is a basic pattern matching system. The matching pursuit algorithms that are discussed later on, rely on the calculation of groups of inner products and this can be achieved using cross-correlation. This 'matching' property of cross-correlation can be observed on figure 2.7 that depicts the auto-correlation of the 'quake' signal, that is the cross-correlation of the signal with itself. In the figure the 'quake' signal completely matches itself when the time lag between the two signals is zero.

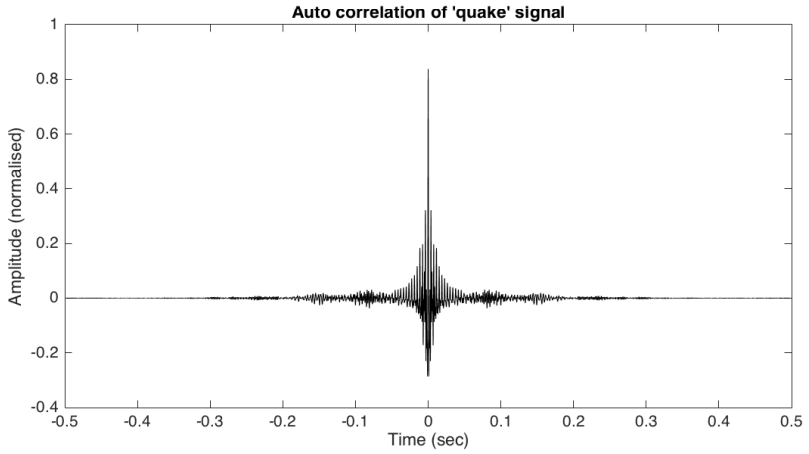


Figure 2.7: Auto-correlation of quake signal. A spike occurs at time zero. The signal is matched with itself completely when the time lag between the signals is zero.

LTI systems theory is beyond the scope of this thesis, therefore only the necessary elements were discussed here. For an in-depth discussion on theory and applications of LTI systems, including filters, the reader is referred to [86] and [89]

2.1.3 Frequency-domain representation

Another signal representation that is complementary to the time domain is the frequency domain representation. The theory stems from Fourier analysis which is a field that evolved from the works of Jean Baptiste Joseph Fourier among others. According to a well referenced historical review in [86], the concept of representing periodic phenomena using sums of harmonically related sinusoidal components dates back at least as far as the Babylonians who used such ideas to predict astronomical events. According to that same review, the modern history of the field started in 1748 with L. Euler studying the normal modes of vibrating strings where he performed the same type of calculation as the Fourier series which is essentially the linear combination of complex exponentials. Regarding the Fourier series, Fourier's motivation was the problem of heat propagation and diffusion where he found that series of harmonically related sinusoids could be used to represent the temperature distribution through a body and he also claimed that any periodic signal could be represented using such series. The following equations define the Fourier series of a continuous-time periodic signal [86]:

$$\boxed{\begin{aligned} x(t) &= \sum_{k=-\infty}^{+\infty} c_k e^{jk\omega_0 t} \qquad (2.39)\\ c_k &= \frac{1}{T}\int_T x(t) e^{-kj\omega_0 t} dt \qquad (2.40) \end{aligned}}$$

where c_k are called the Fourier series coefficients or spectral coefficients and the $e^{jk\omega_0 t}$ components are the harmonically related complex exponentials which can be denoted by $\varphi_k(t)$:

$$\varphi_k(t) = e^{jk\omega_0 t} = e^{jk(2\pi/T)t}, \quad k = 0 \pm 1, \pm 2, ... \qquad (2.41)$$

Equation (2.39) is called the synthesis equation and equation (2.40) the analysis equation. Looking at equation (2.39) we can again observe the process of linear combinations; in this case the weighted (by the spectral coefficients) summation of harmonically related complex exponentials. On the other hand the analysis equation (2.40) is essentially the inner product between the signal $x(t)$ and each complex exponential, and as such the spectral coefficients measure the amount of the signal $x(t)$ that is found at each harmonic of the fundamental component. The inner product between two square-integrable functions (i.e. functions with finite energy) f and g is defined by:

$$\langle f, g \rangle = \int_{-\infty}^{+\infty} f(t) g^*(t) dt \qquad (2.42)$$

where $g^*(t)$ is the complex conjugate of $g(t)$ (in case the functions are complex). The inner product will be discussed further in section 2.3 in the context of matching pursuit algorithms, but generally speaking, the inner product between two functions can be interpreted as a measure of similarity between those two functions. In short the larger the value of the inner product the more similar the two functions are. The mathematical proofs and properties of the Fourier series equations can be found in many texts such as [86] and [88].

A signal property that is directly related to the inner product is orthogonality. In general, a set of functions $\{\varphi_n(t)\}_{n=0}^{\infty}$ on an interval $[-T/2, T/2]$ is said to be orthogonal if:

$$\int_{-T/2}^{T/2} \varphi_n(t)\varphi_m(t)dt = E_n\delta_{nm} = \begin{cases} E_n, & \text{if } n = m \\ 0, & \text{otherwise} \end{cases} \tag{2.43}$$

where δ_{nm} is the Kroneker delta function:

$$\delta_{nm} = \begin{cases} 1, & \text{if } m = n \\ 0, & \text{otherwise} \end{cases} \tag{2.44}$$

and E_n is the energy of $\varphi_n(t)$:

$$E_n = \int_{-T/2}^{T/2} \varphi_n^2(t)dt \tag{2.45}$$

Equation (2.43) states that the inner product between any two functions in the orthogonal set is zero. There are many sets of functions that satisfy (2.43) and in fact the set of harmonically related exponentials used in Fourier series is one of them. Use of orthogonal sets of functions lead to unique signal representations and also simplify mathematical treatments of transforms. Orthogonality will be discussed further in section 2.1.8.

Fourier further developed his ideas to obtain a representation for aperiodic signals as long as they have finite energy (i.e. they are square-integrable), resulting in the Fourier integral or transform (FT). The following equations are the Fourier transform pair of a continuous-time signal $x(t)$ (also shown using the notation of the inner product) [86]:

$$\boxed{X(j\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt = \langle x(t), e^{j\omega t}\rangle} \tag{2.46}$$

$$\boxed{x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(j\omega)e^{j\omega t}d\omega} \tag{2.47}$$

In practical DSP the discrete Fourier transform (DFT) is used instead [87], [89]:

$$\boxed{X(k) = \sum_{n=0}^{N-1} x[n]e^{-j(\frac{2\pi}{N})nk}, \quad k = 0, 1, ..., N-1} \tag{2.48}$$

$$\boxed{x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j(\frac{2\pi}{N})nk}, \quad n = 0, 1, ..., N-1} \tag{2.49}$$

The DFT can be computed using a fast Fourier transform (FFT) algorithm. Such an algorithm was first proposed by Cooley and Tukey in 1965 [92]. The introduction of FFT algorithms heavily influenced the development of DSP in the last five decades and will continue to play an important role in many scientific and engineering fields. Modern numerical software usually provides highly optimised FFT implementations in a form of a function with various input/output configurations. For sound signals in general, the input (i.e. the time-domain representation) is real-valued and the output (i.e. the spectral coefficients) complex-valued. This frequency domain representation

can be converted back to the time-domain using the inverse FTT (IFFT). It should be noted that each pair of equations (2.46), (2.47) and (2.48), (2.49), forms an identity system, in an energy sense, that is, assuming the absence of any modification in the frequency-domain then no energy is lost between the transition from time to frequency and back to time-domain.

Figure 2.8 depicts the frequency representation of the 'quake' signal (fig. 2.1) (zoomed and focused on the first 4000 Hz). We can clearly see that some regions of the spectrum around 250 Hz, 500 Hz and 800 Hz are excited and form visible peaks whereas the frequencies further up the spectrum behave in a more erratic way. The visible peaks of the spectrum will most probably belong to the "stationary" part of the signal, that is the periodic content of the sound, whereas the erratic part of the spectrum corresponds to the "non-stationary" parts of the sound. In fact, when dealing with speech, these excited regions are called formants and are a characteristic of vowel sounds. A quick internet search shows that the excited regions that we see in the figure, approximately correspond to the first formant frequencies of some of the underlying vowels in the word 'quake' (/i/ 240 Hz, /e/ 390 Hz). However the figure is not clear at all in showing any formant transitions, as we intuitively expect from the triphthong in the word 'quake'. The reason is that this representation does not contain any time information. It is a complementary representation to the time-domain representation. Essentially it is possible to look either at the time-domain or frequency-domain of the signal; not both at the same time, or to be precise with any arbitrary resolution.

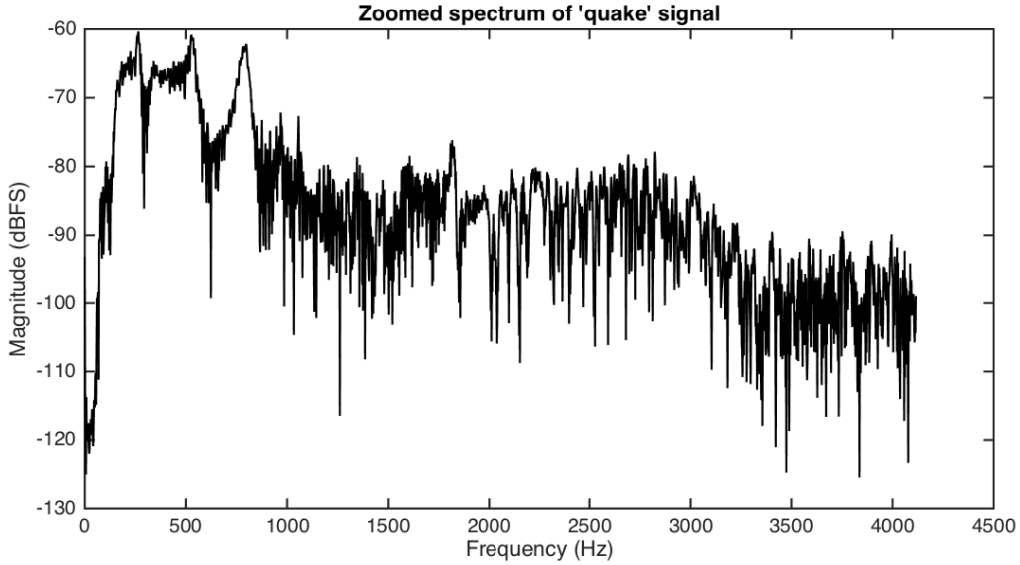


Figure 2.8: Zoomed spectrum of 'quake' signal. This representation is a zoomed version and focuses on the first 4000 frequencies of the signal. We can clearly see that some regions are excited by distinguishable frequency components whereas towards the end the representation becomes more erratic.

The Fourier transform has certain properties and theorems that make it very useful when analysing and implementing systems. First of all the FT of a real sequence is conjugate symmetric with an even magnitude and odd phase spectra. This symmetry is often exploited when storing the results of the DFT (or FFT) where the original signal buffer can be used to store the complex results. Regarding the FT theorems, probably the most important in terms of practicability is the convolution theorem which states that if $X(e^{j\omega}) = FT\{x[n]\}$, $H(e^{j\omega}) = FT\{h[n]\}$ and $y[n] = x[n] * h[n]$ then

$$Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega}) \tag{2.50}$$

where $Y(e^{j\omega})$ is the FT of $y[n]$. The convolution theorem essentially states that convolution of two sequences in the time-domain becomes multiplication of their spectra in the frequency-domain and vice-versa. Another important theorem is Parseval's theorem which is defined as:

$$E = \sum_{k=-\infty}^{+\infty} |x[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\omega \tag{2.51}$$

where the left hand side is the energy of the signal as defined in (2.21) and $|X(e^{j\omega})|^2$ is the energy density spectrum which determines how the energy is distributed in the frequency domain [89]. Parseval's theorem shows that the energy between the two domains is conserved. Finally another theorem that is important for this thesis is the time reversal theorem which says that if $x[n]$ and $X(e^{j\omega})$ are an FT pair and $x[n]$ is reversed then:

$$x[-n] \overset{\text{FT}}{\longleftrightarrow} X(e^{-j\omega}), \quad \text{for complex } x[n] \tag{2.52a}$$

$$x[-n] \overset{\text{FT}}{\longleftrightarrow} X^*(e^{j\omega}), \quad \text{for real } x[n] \tag{2.52b}$$

The convolution theorem combined with the time reversal theorem for real signals and the FFT, allow us to program fast and efficient algorithms that compute groups of inner products which is one of the main operations in the algorithm proposed in chapter 3.

2.1.4 Filters and filter banks

Filters were briefly introduced in subsection 2.1.2 as LTI systems that modify input sequences to produce 'filtered' outputs. In the context of music and audio in general, filtering usually refers to the frequency spectrum of a signal. In particular a filter can attenuate or amplify certain frequency bands (ranges). For example in a hi-fi domestic system the tone controls, bass and treble are filters that modify the low portion and high portion of the frequency spectrum respectively. Another example is the graphic equaliser found in many car audio systems which is a bank of filters that modify different regions of the spectrum (by attenuating or boosting frequencies). A final example of a filter is the human mouth; shaping of the mouth and positioning of the tongue, modify the spectrum of the glottal pulse (generated by the vocal tract) to produce speech.

There are two main families of filters, analogue and digital that operate on analogue and digital signals respectively. Analogue filters are constructed using electronic components such as resistors and capacitors whereas digital filters are programs running either on specialised hardware, like digital signal processors (DSPs) and field-programmable gate arrays (FPGA) or generic hardware (central processing unit (CPU) of personal computers (PC)). It should be noted that digital filters can do exactly what analogue filters can (and sometimes even more) and are usually preferred. Analogue filters tend to be difficult to design and also construct because the filter characteristics are affected by even slight changes in the behaviour of the components. Air temperature, humidity and other factors can change the behaviour of electronic components and as a result analogue filters tend to 'sound' different as time goes by. Digital filters do not have similar problems since they are programs kept in computer memory; they are code instructions that transform an input sequence into a new filtered output sequence. As such a digital filter can be easily upgraded or even replaced by a different type of filter. Compared to analogue, digital filters are much easier to design and construct, they are more flexible and finally cheaper to produce.

A big class of LTI systems can be characterised by an N^{th} order linear constant-coefficient difference equation (or differential for continuous-time) of the following form [89]:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m] \tag{2.53}$$

$$y[n] = \sum_{k=0}^{M} \frac{b_k}{a_0} x[n-k] - \sum_{k=1}^{N} \frac{a_k}{a_0} y[n-k]\} \tag{2.54}$$

Equation (2.54) is a re-arranged version of (2.53). Such equations describe the input-output relationship of a system, that is the implicit specification of the system [89] and must be solved in order to provide an explicit expression. These equations are usually solved in the same way as differential equations and as such require certain auxiliary conditions one of which is the condition of initial rest which states that [89]:

$$\begin{aligned} \text{if } x[n] = 0 \text{ for } n < n_0, \quad \text{then} \\ y[n] = 0 \text{ for } n < n_0 \\ \text{with intial condition: } y[n_0] = 0 \end{aligned}$$

This condition makes the system causal. A causal, LTI system that is characterised by an equation of the form of (2.53) has a unique solution [89].

The accumulator system as described by equation (2.26b) can be exactly derived from (2.53) by setting $N = 1$, $a_0 = 1$, $a_1 = -1$, $M = 0$ and $b_0 = 1$. Equation (2.26b) is thus a difference equation and recursive since its output depends on past outputs (i.e. the system has feedback). Also this form of the equation could be directly implemented on software or hardware. By setting $x[n] = \delta[n]$ in (2.26a) we can get the impulse response of the accumulator system. It is easy to show that the response is infinite. Such systems are called infinite-impulse response (IIR) systems and form one of the main categories of digital filters. Continuing with the accumulator system, if the input to the system is the step function then it can be easily shown that the system becomes unstable; the output of the accumulator will keep increasing without a bound thus violating the stability condition explained in section 2.1.2. This example shows that IIR systems are inherently unstable; this does not mean that stable IIR systems cannot be designed but IIR systems are generally prone to instability due to feedback. More complex IIR systems can be achieved by using higher order linear constant-coefficient difference equations with both $N > 1$ and $M > 1$.

There are many ways to design IIR filters but usually many IIR designs stem from analogue filter designs. Analogue filters have been in existence more than digital IIR and a lot of designs are already available. The treatment usually starts in the continuous domain where the Laplace transform is used to analyse a system and produce the transfer function of the system (i.e. the input/output relationship of the system). This analysis is then converted to the Z-domain which is suited for analysis of digital LTI systems. The conversion from the Laplace to the Z-domain can be achieved using a mathematical mapping known as bilinear transform. The bilinear transform essentially maps the infinite analogue domain into the finite discrete domain. Whilst in the Z-domain, the Z-transform can be used to convert the transfer function of the system into a difference equation (similar to (2.54)) which can be further optimised or implemented.

Another set of systems arises when we set $N = 0$ in equation (2.53). In this case equation (2.53) becomes:

$$y[n] = \sum_{k=0}^{M} \frac{b_k}{a_0} x[n-k] \tag{2.55}$$

It is clear that this equation is non-recursive since it only depends on current and past values of the input alone. Note that equation (2.55) is essentially the convolution sum in equation (2.34) with different summation limits. It is easy to show that the impulse response of (2.55) is:

$$h[n] = \begin{cases} \frac{b_n}{a_0}, & 0 \leq n \leq M \\ 0, & otherwise \end{cases} \tag{2.56}$$

which clearly shows that the response is finite. Such systems form the second main category of filters and are called finite-impulse response (FIR). When dealing with filters the impulse response is also known as the filter kernel, or the filter coefficients, which are responsible for what kind of filtering will occur. It should be noted that all FIR systems are stable. In fact since there is no feedback in the system this makes them inherently stable which is a big advantage over IIR filters.

In the previous example it was shown that filtering can be effectively performed with the convolution operation. The convolution operation is essentially a combination of an accumulator system (eq. 2.26a), multipliers and a delay line (eq. 2.27) and can be implemented either in software or hardware. Many modern DSPs provide built-in hardware multiplier-accumulator units (MAC) and circular buffers which are used to efficiently perform fast convolution operations on incoming signals. When implemented in software there is always more overhead but generally more flexibility.

There are many ways to design FIR filters but the main idea is to somehow create a filter kernel that has certain characteristics. Filtering is then performed by convolving a signal with this kernel. Figure 2.9 depicts the impulse, frequency and phase responses of two FIR filters. The first row of plots corresponds to a low-pass (LP) filter and the second row to a high-pass (HP) filter. These filter kernels (i.e. the impulse responses) were designed using MATLAB®'s filter design application 'fdatool'. In particular the window method was used with a Kaiser window, sampling frequency of 44100 Hz a cut-off frequency of 10800 Hz and order 21. The cut-off frequency is the boundary in the filters frequency response and the order dictates the length of the filter kernel. Inspecting the frequency responses it can be clearly seen that the filters attenuate and reject the frequencies they are supposed to; the LP passes all low frequencies up to the cut-off frequency and attenuates the rest whereas the HP attenuates all low frequencies up to the cut-off frequency and passes all the rest. The phase responses indicate that the filters are linear-phase (excluding the phase wraps) which is a desirable property and easily attainable with FIR filters. In short linear-phase means that all frequency components of an input signal will be delayed equally by the filter thus avoiding phase distortion and subsequent 'colouration' of the sound. Linear-phase filters are usually preferred when mastering audio for music production, because of this property. A sufficient, but not necessary, condition for a filter to be linear-phase is to have a symmetric kernel as is the case of the LP filter impulse response in fig 2.9. The impulse response of the HP filter is produced by flipping vertically the first half of the LP impulse response thus creating an odd-symmetry (i.e. $h[n] = -h[-n]$).

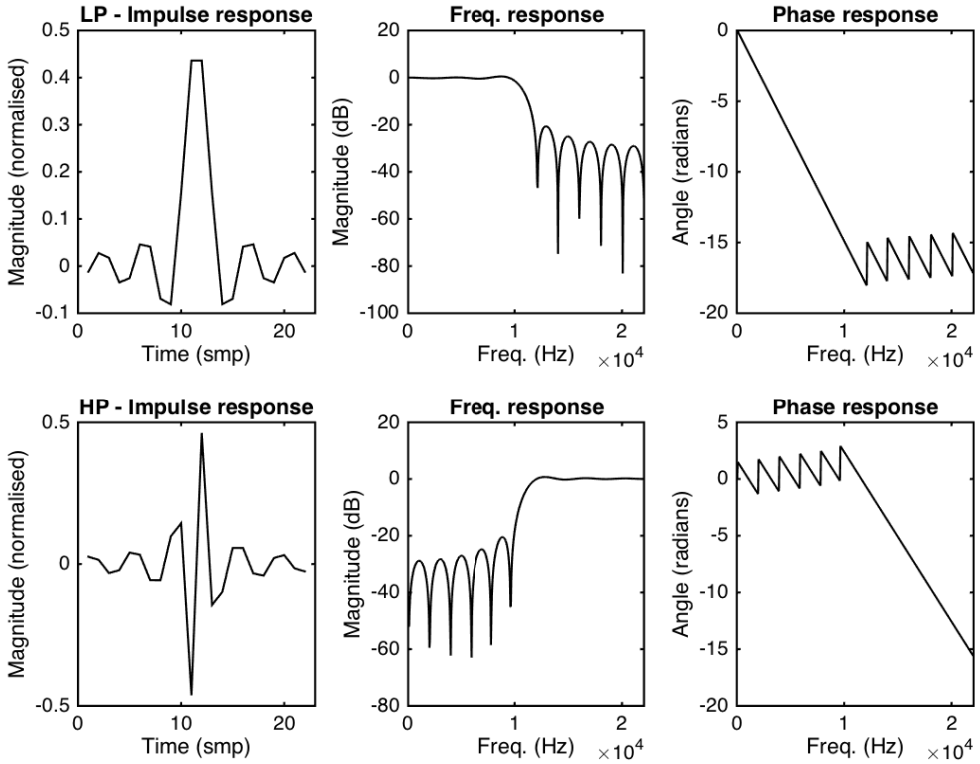


Figure 2.9: Typical low-pass (LP) and high-pass (HP) filters. Each row of plots show the impulse response, frequency response and phase response of an LP and HP filters respectively. Both filters were designed in MATLAB®'s fdatool. The filters were designed using the window method based on a Kaiser window with order 21. The sampling rate was set to 44100 Hz and the cut-off frequency to 10800 Hz. The HP impulse response is the LP impulse response with its first half flipped vertically (across the x-axis), creating an odd-symmetry.

Some other common filter types that are encountered are the band-pass (BP) filter that allows a specific frequency range to pass (e.g. 1000 Hz to 1100 Hz) and rejects everything else and the notch-pass (NP) filter that rejects (attenuates) a specific frequency range and allows everything else. No matter the type of filter, all filters have certain common characteristics that describe them. These characteristics essentially become filter design parameters that are used to compute and generate the filter coefficients. Figure 2.10 depicts the most common characteristics found on a filter. The passband is the frequency region of the signal that is allowed to pass through the filter and the stopband the frequency region of the signal that is rejected by the filter. The transition band is the frequency region around the cut-off frequency where the transition between the boundary of the filter occurs. In this particular filter the transition band seems to be a few hundred Hz. The passband ripple is the maximum amount of magnitude fluctuation that occurs at the passband frequencies and the actual ripples can be better seen in the second plot of the figure. The stopband attenuation is the magnitude range between the nominal passband attenuation (which ideally is 0 dB) and the stopband attenuation level. An ideal filter would have perfectly flat (no ripples) passband attenuation at 0 dB, a very sharp transition band, and very low stopband attenuation level (thus large stopband attenuation range). The ideal filter is not realisable but optimal filters can be designed depending on the application. Generally the better characteristics a filter has the longer the impulse response will be, leading to more memory being utilised, longer computation times and larger overall system delay.

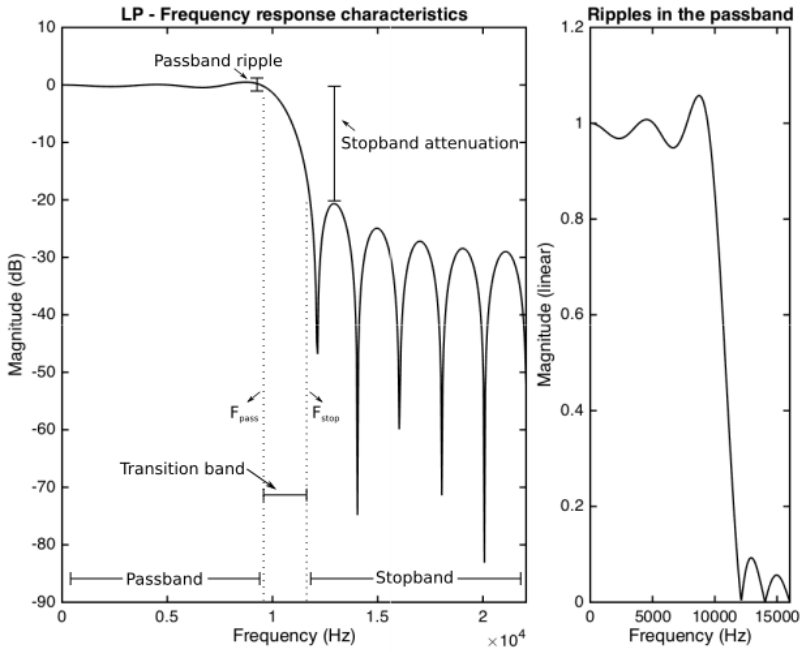


Figure 2.10: Typical LP filter characteristics. The passband is the range of frequencies that pass through the filter, whereas the stopband is the frequency range that does not pass through the filter, the passband ripple is the allowed magnitude fluctuation the passband frequencies can have, the stopband attenuation is the magnitude range between passband and stopband magnitudes and the transition band ($F_{pass} - F_{stop}$ range) is the range of frequencies where the filter transition between passband and stopband occurs.

By taking advantage of the properties of LTI systems, filters can be interconnected, as explained in section 2.1.2, to produce interesting configurations. For example two 2^{nd} order LP IIR filters can be cascaded (connected in series) to produce a 4^{th} order filter. Another interesting configuration is filters connected in parallel which leads to the creation of filter banks. The graphic equaliser mentioned at the beginning of this subsection is an example of a filter bank. Also many transform methods, that will be discussed on later sections, can be interpreted in terms of filter banks.

Probably the simplest possible filter bank comprises two filters; an LP and a HP both having the same cut-off frequency set at the middle of the frequency bandwidth of the signal. These filters can be designed to be complementary so that the output of their pass-bands cover up the whole frequency spectrum of the signal. This filter bank is depicted in figure 2.11 where an input signal is split into two new signals; one containing the lower half of the spectrum of the original and the other containing the upper half of the spectrum, that is two new signals are created occupying different sub-bands. Filter H_0 is the low-pass filter and filter H_1 the high-pass. These filters are referred to as the analysis filters.

Simple filter bank

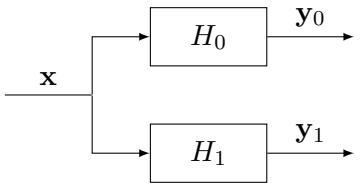


Figure 2.11: A simple filter bank comprising two filters; a low-pass (H_0) and high-pass (H_1). The incoming input signal is split into two new signals y_0 and y_1 containing a low-passed and a high-passed version of the original respectively.

In many applications it is often desirable to split the incoming signal in more than two sub-bands. This can be achieved by repetitively using the structure in figure 2.11 to create a tree-like filter bank. That is, each of the outputs of the first filter bank is passed through a new set of filter banks leading to four new signals with each passing through another set of filter banks and so on so forth until the desired depth level has been reached. Each level has a set of filter banks, with level 0 having the first filter bank. In order to achieve this decomposition using the same set of basic filters (H_0 and H_1), the signals at each level have to be decimated (down-sampled) before the filtering of the next level. This constraint stems from the Nyquist-Shannon sampling theorem explained in section 2.1.1. To recall, the sampling theorem states that under certain conditions a continuous-time signal can be fully recovered by its samples. In particular a band-limited signal with a bandwidth B can be fully recovered by its samples, if these were sampled at a constant sampling period of $\frac{1}{2B}$. Applying this to our situation, after the first filtering the output signals y_0 and y_1 will contain half of the bandwidth of the original signal (since the filters split the bandwidth in two equal sub-bands) but with the same number of samples as the original. Down-sampling by a factor of 2 will reduce the number of samples by half but since the output signals already have half the bandwidth of the original, they will still represent the same time interval. This process is illustrated in figure 2.12.

A more advanced filter bank.

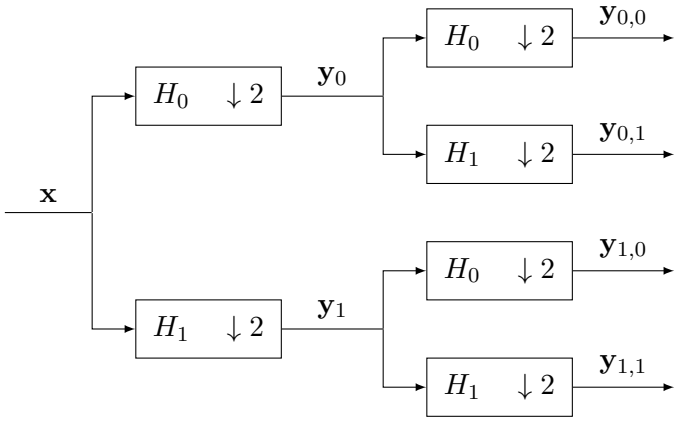


Figure 2.12: The initial signal x is split by the 0^{th} level filter bank into two sub-band signals, y_0 and y_1 with each being decimated (down-sampled) by a factor of 2; the filtering and decimation operations are denoted in the same box for each filter (decimation is the $\downarrow 2$ symbol). The filtered and decimated outputs are each then filtered and decimated by the 1^{st} level filter banks producing four new signals $y_{0,0}$, $y_{0,1}$, $y_{1,0}$ and $y_{1,1}$. This process can be continued up to the desired level of the decomposition.

In many situations (e.g. encoding/decoding schemes) it is desirable to re-synthesise the signal back to its original form and this can be achieved by following the reverse process, that is, starting at the end of the filter bank tree, each signal is up-sampled by a factor of 2 (by inserting a zero value every other sample) and then filtered with a new set of filters, called reconstruction filters, that effectively reverse the effect of the analysis filters. In order for this process to be possible (i.e the filter bank to be reversible) the analysis and reconstruction filters must meet certain conditions known as 'Perfect Reconstruction' (PR). In practice the reconstruction filters are derived from the analysis filters [93]. A detailed discussion on PR conditions can be found in [94].

2.1.5 Time-Frequency representation

A fundamental law in communication theory states that:
"The transmission of a certain amount of information per unit time requires a certain minimum waveband width."

(quoting Denis Gabor [95])

This principle is a consequence of the uncertainty principle that states (in the context of quantum mechanics) that the velocity and position of an object cannot both be measured exactly at the same time. The uncertainty principle is also known as Heisenberg uncertainty principle (or Heisenberg inequality) published by Werner Heisenberg in 1927. The following mathematical description of the principle is taken from [96]. Assume that a continuous-time signal has unit energy, that is $\int_{-\infty}^{+\infty} |x(t)|^2 dt = 1$, The duration, or time spread, of $x(t)$ is defined as:

$$\Delta_t = \int_{-\infty}^{+\infty} t^2 |x(t)|^2 dt \tag{2.57}$$

The bandwidth, or frequency spread of $X(\omega)$ is defined as:

$$\Delta_\omega = \int_{-\infty}^{+\infty} \omega^2 |X(\omega)|^2 d\omega \tag{2.58}$$

where $X(\omega)$ is the FT of $x(t)$. The uncertainty principle states that:

$$\Delta_t \Delta_\omega \geq \sqrt{\frac{\pi}{2}} \tag{2.59}$$

and the equality holds only for Gaussian signals:

$$g(t) = \sqrt{\frac{\alpha}{\pi}} e^{-\alpha t^2} \tag{2.60}$$

That is, a lower bound is imposed to the product of time and frequency spreads or in other words we cannot have arbitrary time and frequency resolution but there is always a trade-off between one another. The full mathematical proof can be found in [96].

Although, this principle was known to the engineers of the early 20th century, it was not until the publication of D. Gabor's "Theory of Communication" in 1946 [95], that certain practicalities were addressed and a more rigorous formulation was provided. In this paper , Gabor proposed a system where any signal can be represented in the time-frequency domain. He called such two-dimensional representations "information diagrams" since areas within them are proportional to the amount of data they can convey. This representation comprises "elementary signals" that occupy the smallest area in the information diagram and thus convey exactly one datum or one "quantum of information". He also suggests the use of harmonically related complex exponentials modulated by a Gaussian pulse, as "elementary signals", since Gaussian signals provide the best joint time-frequency localisation (as mentioned earlier, equation (2.59) is only equal for Gaussian signals of the form shown in (2.60)). Furthermore he suggests representing each elementary signal as a rectangle on the information diagram, the width and height of which is the time and frequency

spreads of the elementary signal respectively (equations (2.57) and (2.58)) and centred around the mid points of the spreads (t_0, f_0). He proposes the name "logon" to describe such an area with its associated datum. Thus a logon is one elementary quantum of information. This system if known today as the Gabor transform and is a special case and the precursor of the short-time Fourier transform (STFT). Figure 2.13 depicts a conceptual Gabor representation. Each rectangle is a logon, representing a complex exponential signal with certain magnitude and phase.

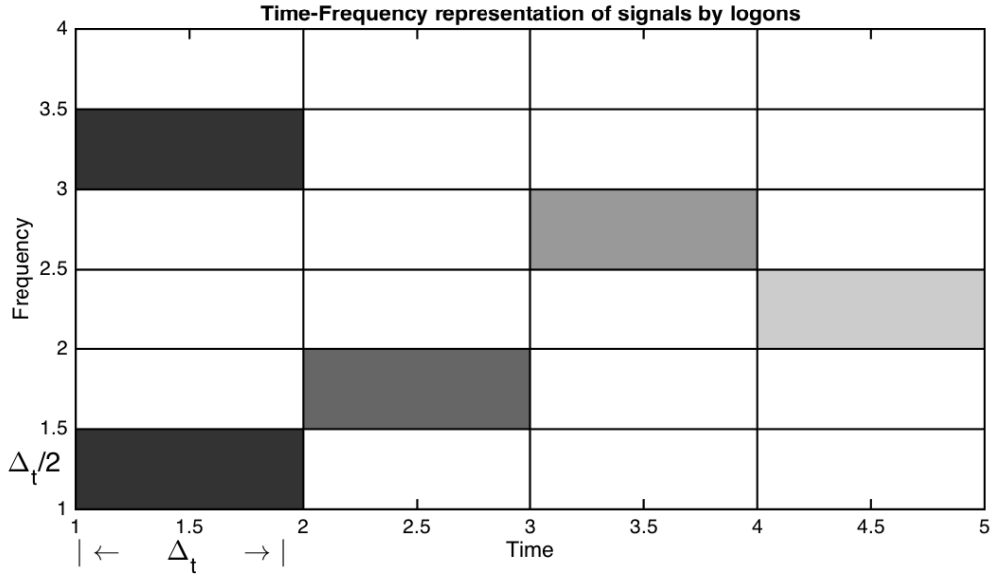


Figure 2.13: This is a Gabor representation of a hypothetical signal. Each rectangle represents a logon which is one elementary quantum of information. The colour of each logon represents magnitude with darker colours corresponding to higher energy logons.

After 1946 the theory of short-time Fourier analysis and synthesis evolved through a series of important developments in the field of communications such as [95], [97], [98], [99], [100], [101] and [102]. At the time the main motivation was the development of robust vocoders, that is speech analysis/synthesis systems for efficient and high quality speech transmission as well as speech modification. Vocoders that are based on short time Fourier analysis/synthesis are usually called "Phase Vocoders", as named by Flanagan and Golden who first described such a system [97]. There are essentially two main mathematical models to describe STFT analysis/synthesis systems; the filter-bank summation model suggested by Flanagan and the block-by-block analysis/synthesis model suggested by Gabor. In the filter-bank summation model, a signal is passed through a parallel bank of contiguous band-pass filters, the outputs of which are used to define the amplitude and phase spectra of the signal for a sliding time aperture, thus creating a short-time Fourier spectrum of the signal at particular time frames. The original signal can be reconstructed by the summation of the outputs of the band-pass filters. In the block-by-block analysis/synthesis model, an input signal is segmented into successive overlapping time segments. Each segment is then Fourier transformed to produce a short time Fourier spectrum. The original signal can be obtained by taking the inverse Fourier transform of each spectrum to recover the time domain segments and by appropriately summing those segments to produce the original signal. The synthesis method is known as the overlap-add synthesis method. It should be noted that, assuming appropriately defined analysis and synthesis windows, both models are mathematically identical and interchangeable [102].

Crochiere provides a generic framework for STFT analysis/synthesis in [102]. In particular the STFT analysis of a signal $x(m)$ can be defined by :

$$\boxed{\begin{aligned} STFT\{x\} \stackrel{\text{def}}{=} X(sR_a, k) &= \sum_{m=-\infty}^{+\infty} x(m)h(sR_a - m)W_N^{mk} \\ k &= 0, 1, 2, ..., N-1 \\ W_N &= e^{-j2\pi/N} \end{aligned}} \tag{2.61}$$

where N is the FFT size (and thus the number of frequency samples), k is the discrete-frequency index, also known as the frequency bin number, $h(m)$ is the analysis window, s denotes the frame time-index and R_a the analysis hop size measured in samples. W_N^k is the set of complex exponential functions. In practical FFT implementations, W_N^k are called twiddle factors and they are treated as pre-computed constants, thus speeding up the computation of the algorithm.

The STFT synthesis equation (or inverse STFT (ISTFT)) can be defined by:

$$\boxed{\begin{aligned} ISTFT\{Y\} \stackrel{\text{def}}{=} y(n) &= \sum_{s=-\infty}^{+\infty} g(n - sR_s)y_s(n - sR_s) \\ y_s(sR_s) &= \frac{1}{N}\sum_{k=0}^{N-1} Y(sR_s, k)W_N^{-nk} \end{aligned}} \tag{2.62}$$

where $y(n)$ is the approximation signal, $g(n)$ the synthesis window, $Y(sR_s, k)$ the potentially modified discrete short-time transform at time-index s, with R_s denoting the synthesis hop size measured in samples. This synthesis equation expresses the overlap and add method.

The output of the STFT is complex valued:

$$X(u, k) = X_R(u, k) + jX_I(u, k) = |X(u, k)|e^{j\phi(u,k)}, \quad u = sR_a \tag{2.63}$$

with the magnitude and phase spectra of each frame given respectively by:

$$|X(u, k)| \stackrel{\text{def}}{=} \sqrt{X_R^2(u, k) + X_I^2(u, k)} \tag{2.64}$$

$$\angle X(u, k) \stackrel{\text{def}}{=} \arctan\left(\frac{X_I(u, k)}{X_R(u, k)}\right) \tag{2.65}$$

The complex output values are essentially the coefficients of the representation. It is typical in analysis/synthesis systems to modify the coefficients in particular ways with the aim of data-reduction or sound modification. As such $Y(sR_s, k)$ in the synthesis equation denotes a possibly modified spectrum. In case of absence of any modifications (i.e. $X(sR_a, k) = Y(sR_s, k)$), and assuming that $R_a = R_s$ and that the analysis and synthesis windows are chosen appropriately, then the above equations constitute an identity system. In practice the original and approximation signals have minor discrepancies due to numerical precision errors, produced usually by the FFT and similar operations.

Figure 2.14 depicts a spectrogram of the 'quake' signal. In this representation the x-axis denotes time, the y-axis frequency and the color of each time-frequency point, magnitude, with darker

colors representing higher magnitude (i.e black corresponds to maximum magnitude). This is a more complete representation compared to the time and frequency-domain representations alone. In order to make the figure clearer, the spectrogram is zoomed in to show a frequency range of up to 7000 Hz, since there was no significant energy above this frequency. The figure clearly shows the behaviour of the underlying utterances of the word 'quake'. In particular the time region 0.0 to time 0.09 seconds approximately, represents the first 'q' (/k/)consonant. It can be seen that this region does not contain any visible periodicities and that the energy is spread out across all frequencies. This is typical for transient, non-stationary sounds. Although looking a bit more carefully, the start of some formants can be barely distinguished. The time region 0.09 up to 0.29 seconds represents the triphong /wei/. In this time region, quasi-stationary (i.e. slowly time-varying) components are clearly visible. From this figure it is difficult to distinguish which components belong to which vowels but the transitions are visible and this is what was expected. The time region 0.32 to 0.35 has almost no energy and this is the instance in the word just before the final consonant, where there is a little pause while the tongue moves to the right position. Finally the region 0.35 to 0.49 represents the final consonant sound and again it can been seen that the energy is spread throughout with no visible quasi-stationary components.

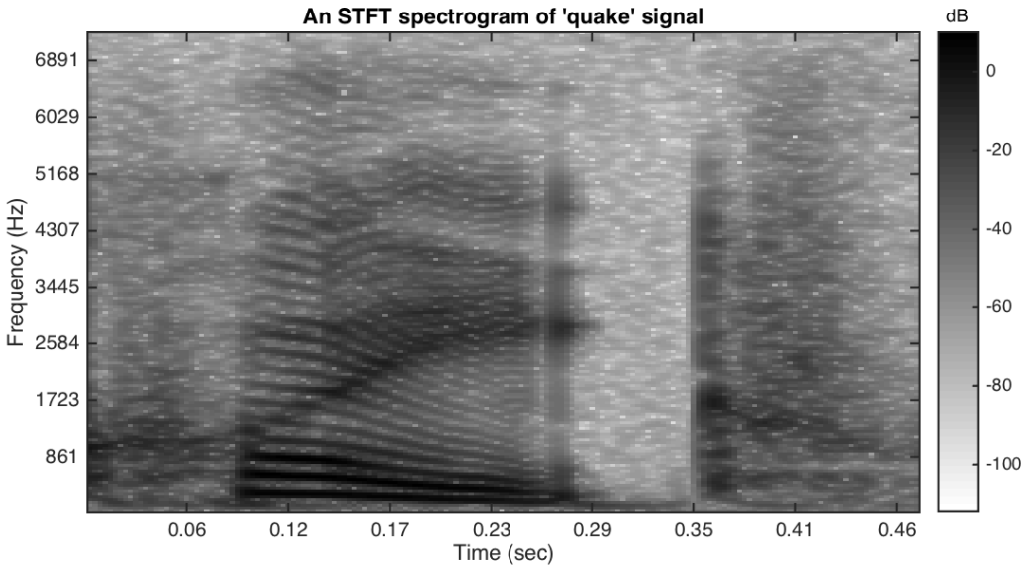


Figure 2.14: An STFT spectrogram of the 'quake' signal. This is a more complete signal reprsentation compared to the time-domain and frequency-domain representations alone.

The STFT analysis/synthesis theory provides a powerful and flexible framework for implementing a variety of signal modifications such as time-frequency filtering, time stretching and compressing (or scaling in general), pitch-shifting, sound morphing and mutation, de-noising and source separation to name a few. An excellent resource with phase vocoder theory, practical implementations and applications can be found in [103]. Albeit, algorithms based on STFT representations have some important drawbacks.

First of all the modifications of the time-frequency representation usually lead to audible artifacts such as musical noise and 'phasiness'. Many ways have been developed to reduce these artifacts, but even so, outputs from such algorithms always feel that they have been processed by a phase vocoder system, especially in extreme cases of signal modification. Another important drawback of STFT analysis/synthesis is the fact that the time and frequency resolution of the representation are fixed, or in connection to the uncertainty principle, the time-frequency tiles (i.e Gabor's logons)

have a fixed time-frequency product. In particular the frequency and temporal resolution of the STFT is determined by the window size. This relationship can be expressed by:

$$F_r = \frac{F_s}{N} \tag{2.66}$$

where F_r is the frequency resolution, F_s the sampling frequency of the signal and N is the length of the window and usually the length of the FFT to be performed as well (although the size of the FFT can be larger which leads to spectral interpolation and not an increase in frequency resolution). Generally, non-stationary sounds like transients are better analysed using short time windows (good temporal resolution, bad frequency resolution), whereas stationary and quasi-stationary sounds like vowels are better analysed using longer windows (good frequency resolution, bad temporal resolution). There is always this trade-off between time and frequency resolution which to a certain extent is dictated by the type of signal under analysis. Thus we need to know the properties of the signal and then select appropriate parameters for the algorithm; parameters like the size of the FFT, type and size of analytsis and synthesis windows, size of overlap etc. Again there are solutions to overcome these problems, within the STFT framework, such as using multi-rate STFTs but usually these solutions tend to be convoluted resulting in an explosion of control parameters that are not easy or intuitive to manage. Finally, probably the biggest limitation is that all STFT algorithms represent a signal using a fixed basis, namely the set of harmonically related complex exponentials. A basis in a vector space V is a set of linearly independent vectors (i.e. no basis vector can be represented in terms of any other basis vectors) that can describe any other vector in that space via linear combinations (more will be said about basis vectors in a later section). But not all signals can be well represented using the complex exponential basis. For example think about the consonant 'q' in figure 2.1; why represent it using a sinusoidal waveform when its pattern is clearly more rough and noisy? Of course it can be represented using the complex exponential series, as it can be seen in figure 2.14 the energy of the signal at this time instance, spreads out to all frequency components; but clearly this is not the best representation.

Despite the limitations of STFT analysis/synthesis systems, they do possess one very attractive quality and this is fast computation with results of acceptable quality. In fact many STFT based analysis/synthesis systems can be programmed to operate in real-time. Also current, improved phase vocoder technologies work really well with certain types of sound signals and for certain applications. Having said that, today's state of the art time scaling and pitch shifting technologies, such as Zplane's élastique time-stretching and pitch-shifting algorithms [104][1] and Zynaptiq's 'ZTX precision time stretching and pitch shifting' algorithms [105][2] (which seems to be connected with Prosoniq's MPEX [106][3]), work on a combination of different technologies such as adaptive transforms and neural networks (as they claim on their websites).

2.1.6 Spectral processing

The theory of STFT analysis/synthesis spurred the development of numerous algorithms, an important family of which is spectral processing. In the mid eighties Smith and Serra [107]

[1] https://products.zplane.de/elastique-pitch-2
[2] http://www.zynaptiq.com/ztx/
[3] http://mpex.prosoniq.com/

(PARSHL system), [108] and independently McAulay and Quatieri [109] (STS system) developed the ideas of spectral processing (also known as spectral modelling). The main idea behind spectral processing is to analyse a sound and produce alternative frequency-domain representations which can be modified and transformed back to produce new sounds [103]. Such systems typically start with an STFT analysis/synthesis method to get the frequency-domain representation. Then the spectrum is used to extract and compute certain relevant features which can be used to form a higher-level representation of the sound. These features can then be modified to produce an array of effects and finally are synthesised to produce the processed output signal. There are several models that have been suggested through the years. Initially the sinusoidal model treats sounds as sums of time-varying sinusoids. This can be expressed as:

$$x(t) = \sum_{r=1}^{R} A_r(t) cos(\theta_r(t)) \tag{2.67}$$

where R is the number of sinusoids, $A_r(t)$ and $\theta_r(t)$ are the instantaneous amplitude and phase of the r^{th} sinusoid. The basic algorithm starts by converting the sound to the frequency-domain using the STFT. The next step detects the spectral peaks from the spectrum of each frame extracting the magnitude, frequency and phase for each peak and finally a peak tracking and continuation algorithm is used to organise those features into time-varying sinusoidal tracks (in the form of (2.67)). The original sound is reconstructed by using additive synthesis. Figures 2.15 and 2.16 depict the two main steps in the sinsusoidal model algorithm, peak picking and peak tracking respectively. Figure 2.15 shows the spectrum of one frame as obtained from the STFT analysis step. The peak picking algorithm has selected a number of peaks, marked with 'x'. Notice that not all visible peaks have been selected and this is due to constraints set to influence what constitutes a peak or not. Also note that not all these peaks will be selected; some of them might be rejected in the peak tracking step. Figure 2.16 depicts the peak tracking and continuation step in full, that is after the algorithm has finished. The sinusoidal tracks are clearly visible. Based on our previous investigation of the signal it is relatively easy to distinguish between the different regions of the word 'quake'. Frames 1 until 17 approximately represent the first consonant 'q', frames 18 until 53 the triphthong /weɪ/, frames 54 until 55 are almost silent as already observed and finally frames 56 until the last one represent the final consonant. We can also observe that the sinusoidal tracks that belong to the consonant regions have a more erratic behaviour and are more tightly packed together whereas the sinusoidal tracks that occur at the triphthong region are slow-varying and seem harmonically related. The sudden dip in frame 29 cannot be observed in the other representations (figures 2.1 and 2.14) and is most probably a mistake maybe due to bad analysis parameters of some other peak tracking computation. Nonetheless the original and reconstructed signals in figure 2.17 look very similar and sound almost identical; in fact an untrained ear might not even notice the differences. It is worth noting though that the sinusoidal tracks in figure 2.16 behave in a different manner than the visible partials in the STFT representation of figure 2.14. Actually the whole uprising transition of partials that is seen in the STFT representation (as seen at the triphthong region) is missing from the sinusoidal model representation.

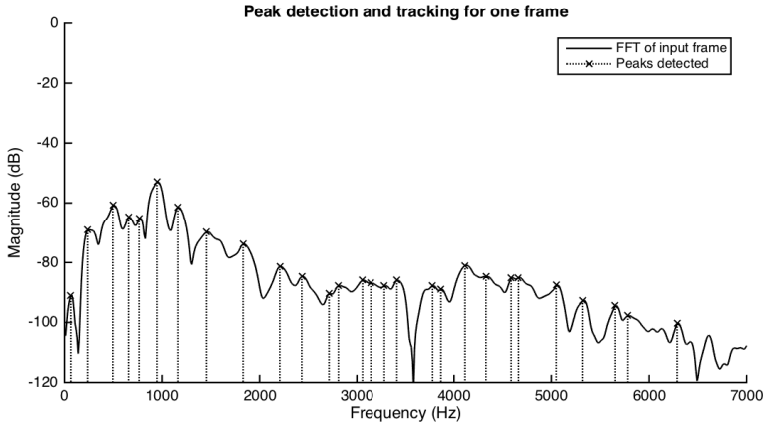


Figure 2.15: Peak picking stage of the sinusoidal model. The spectrum is obtained via STFT analsysis. The peak picking algorithm selects spectral peaks based on certain constraints. Some of these peaks will be kept and tracked while others will be rejected by the peak tracking and continuation algorithm.

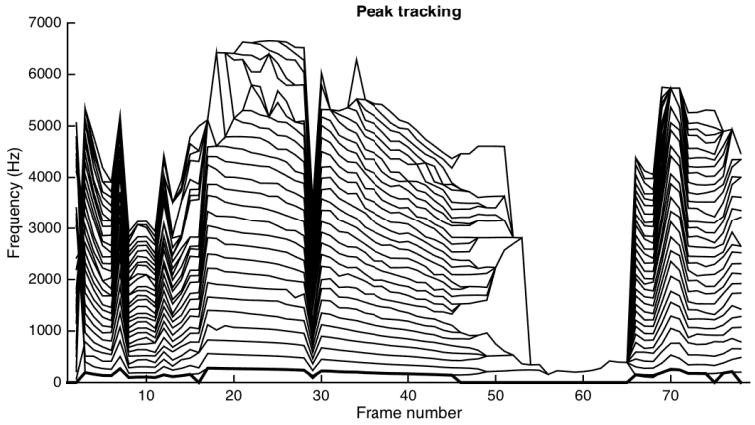


Figure 2.16: Peak tracking and continuation stage in full. The sinusoidal tracks can be clearly seen along with the regions of individual speech utterances.

Another spectral model is the sinusoidal plus residual model [108][110][111]. In this model the sinusoidal part models only the quasi-stationary partials of the sound and the residual models the signal that is left after extracting the sinusoidal part from the original signal. The residual should ideally be a stochastic component. This model can be expressed as:

$$x(t) = \sum_{r=1}^{R} A_r(t) cos(\theta_r(t)) + e(t) \tag{2.68}$$

where R, r, $A_r(t)$ and $\theta_r(t)$ have the same meaning as in the sinusoidal model and $e(t)$ is the residual. The residual can also be modelled as filtered white noise [103]:

$$e(t) = \int_0^t h(t,\tau) u(\tau) d\tau \tag{2.69}$$

where $u(t)$ is white noise and $h(t,\tau)$ is the impulse response of a time-varying filter. The original signal can be reconstructed using additive synthesis for the sinusoidal component and subtractive synthesis for the residual, in case it was modelled as filtered white noise. Another further improvement is the sinusoidal plus transient plus noise model described by Verma in his PhD thesis [112]. The three parts of this model operate in series. The first step models the transients which are removed from the original signal to produce a residual signal. The second step models the sinusoidal part of the signal which is removed from the residual of the previous step to produce

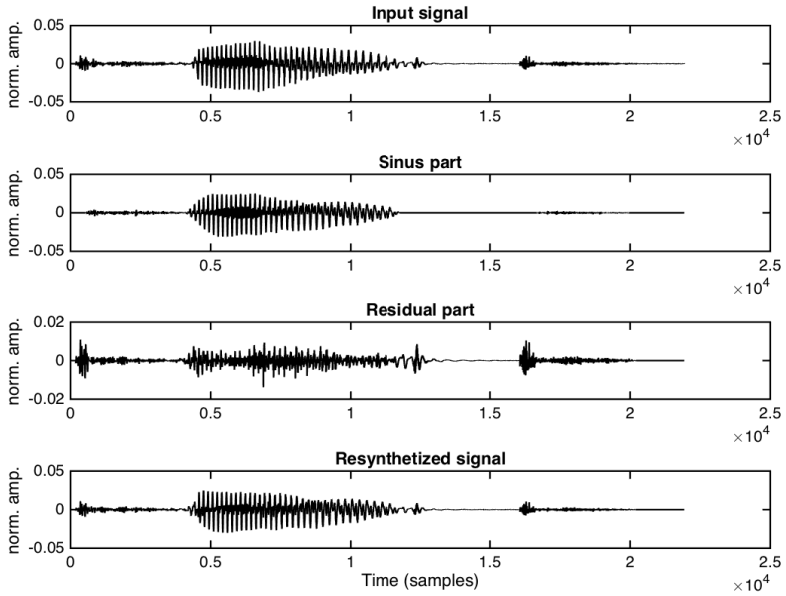


Figure 2.17: Sinusoidal model results for 'quake' signal. These are the waveforms of the resulted components, from top to bottom: the 'quake' signal, the sinusoidal component, the residual component and the reconstructed signal. Although the original and reconstructed signals (first and last signals) do not look exactly the same they do sound very similar to the extend that an untrained ear might not even hear any differences.

a final residual signal. This final residual signal is a noise-like signal which can either be left untreated or be modelled as in (2.69) or using more advanced schemes some of which are described in [112].

The field of spectral processing has evolved significantly in the past couple decades with research focusing on problems such as parameter estimation (e.g. magnitude, frequency and phase), high level attribute computation (e.g. fundamental frequency, spectral shape of sinusoidal and residual components, degree of harmonicity, spectral tilt and spectral centroid), peak detection, pitch detection, peak tracking and continuation and noise modelling. A big list of relevant research can be found in [113]. Spectral processing methods are powerful signal representation techniques that allow a signal to be expressed in a domain that is musically intuitive and from which sound modifications and/or transformations can occur in novel ways. It should be noted though that such methods have certain drawbacks. First of all many of the techniques described in the aforementioned references work well for monophonic signals only and rely on the 'pseudo-harmonicity' of the signals, a constraint which sometimes is prohibitive and leads to bad analysis. Secondly the choice of the model depends on the type of signals to be analysed and there is no straight forward way to develop and implement a generic model that deals with all type of signals equally well. Spectral modelling methods involve a lot of complex steps that all affect the final representation in some way and any choice of parameters should be carefully considered based on the application in mind. Finally the increased complexity of the algorithms lead to increased computation times which renders many of this techniques unsuitable for real-time processing. Generally there is a trade of between flexibility and computation time, with more advanced and complex models being more flexible but more time-consuming and vice versa. Slow computation times is not necessarily a major drawback since some applications do not require real-time processing.

Despite the drawbacks and limitations of spectral modelling, these techniques offer an alternative view of sound representation and modification and many good technologies have stemmed from spectral processing research. For the purposes of this thesis, the biggest result of spectral modelling techniques is the idea of an alternative, higher-level representation of sound by generalising the standard STFT analysis/synthesis framework. Essentially the STFT is used as a "pre-processing" step to extract and compute higher-level parameters that are used to describe a signal. The idea of this intermediate, pre-processing step is pivotal to the development and implementation of the algorithm that is proposed in chapter 3.

2.1.7 Wavelets and time-scale representation

Wavelet theory, as known today, stems from the cumulative results of independent research and technologies that occurred during the 20^{th} century, in different scientific fields such as mathematics, quantum physics, electrical engineering and seismic geology [114] and has been applied to a diverse range of signal processing problems including computer vision, image and speech compression, earthquake detection, de-noising and sparse representations to name a few. Since the early 20^{th} century researchers were aware of the limitations of the FT and were investigating signal expansions using basis signals other than the Fourier basis (i.e. the set of harmonically related complex exponentials). Wavelet theory evolved almost in parallel to STFT theory. The first wavelet and only example for a long time was created by Alfred Haar in 1909 and it appeared in the appendix of his thesis which dealt with the theory of orthogonal function systems (also the name of his thesis; translated in English). During the 1930's several physicists, like Paul Levy, John Littlewood, Richard Paley and Elias M. Stein, independently contributed ideas to the evolving field. Between 1960 and 1980 mathematicians Guido Weiss and Roland R Coifman further contributed by their study of elementary functions called "atoms". In the early eighties, major contributions to the field were made by Jean Morlet and Alex Grossmann who started the theoretical treatment on wavelets and in 1985 Stephane Mallat made further contributions by discovering relationships between quadrature mirror filter banks and wavelet bases leading to multi-resolution techniques. Based on these results, Yves Meyer constructed the first non-trivial wavelet and in 1988, Ingrid Daubechies constructed orthogonal wavelets which are probably the most well known wavelets to date. It was during the late eighties and early nineties that wavelet theory started to unify under a common mathematical framework and begun to be applied to a large range of problems. More detailed introductions to wavelets with historical perspectives can be found in [114] and [94]. An excellent review and tutorial on wavelets can be found in [115]. Finally an extensive mathematical treatment of wavelets and multi-resolution techniques in general, can be found in [63].

Probably one of the main motivations behind wavelets was the desire to overcome the drawbacks of the FT and subsequently the STFT. As it was shown in figure 2.8 of section 2.1.3 the FT is unable to represent non-stationary signals; the spectral coefficients represent the average energy of each frequency component for the whole duration of the signal i.e a global frequency, thus any discontinuity, transient or abrupt change in the time domain will be spread throughout the frequency spectrum. The STFT tries to resolve this issue by segmenting the time-domain signal into overlapping frames that are Fourier transformed to obtain localised in time spectra, where

the information within can be considered stationary, thus leading to the two-dimensional, time-frequency representation shown in figures 2.13 and 2.14. But as already explained in section 2.1.5 the main drawback of the STFT is its fixed time-frequency resolution which is dictated by the choice of the analysis window and conceptually can be explained as a segmentation of the time-frequency plane similar to figure 2.13 where all time-frequency regions, or tiles, have a fixed time-frequency product. Wavelets essentially try to overcome this problem by allowing the time and frequency resolutions (Δ_t and Δ_ω, equations (2.57) and (2.58) respectively) to vary thus leading to different segmentations of the time-frequency plane. This is the main idea behind multi-resolution analysis.

It is informative to view wavelet analysis as a bank of band-bass filters which have constant relative bandwidth [115]:

$$\frac{\Delta_\omega}{\omega_c} = c \tag{2.70}$$

where ω_c is the center frequency of the band-pass filter and c is a constant. This is also known as "Constant-Q" analysis [115]. Equation (2.70) effectively produces frequency spreads that are logarithmically related. It should be noted that both Δ_t and Δ_ω change while ω_c changes so the uncertainty principle is not violated; the time-frequency tiles have a different arrangement which satisfies equation (2.59). In particular long windows are used for good frequency resolution at low frequencies and short windows are used for good time resolution at high frequencies. This time-frequency segmentation is depicted in figure 2.18.

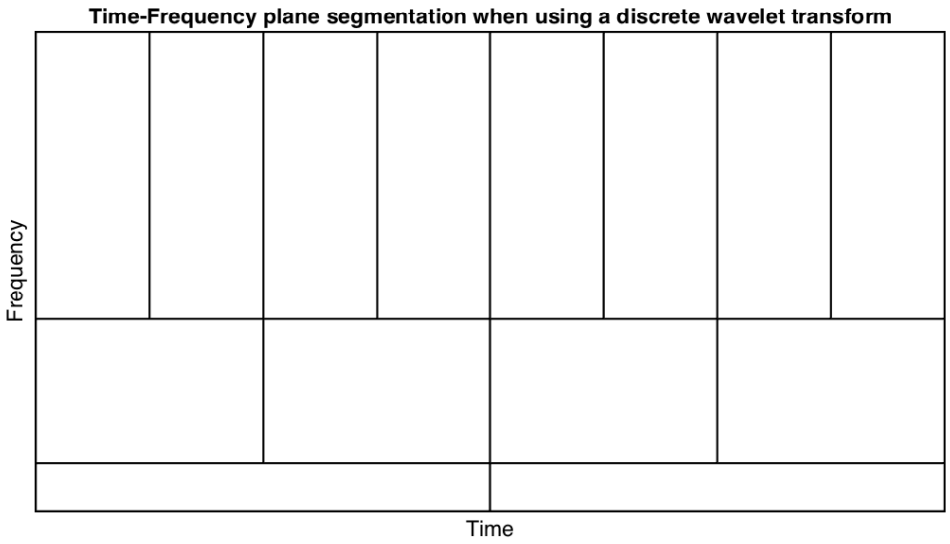


Figure 2.18: Coverage of time-frequency plane for WT.

Another way to see wavelets is as basis functions similar to the basis functions of FT, with the main difference being that these functions are scaled (i.e stretched or compressed in time) and time shifted (translated in time) versions of a prototype function, known as the "analysing" wavelet (or "mother" wavelet). Wavelets introduce the notion of scale which is an alternative to frequency and can be roughly thought of as being the inverse of frequency; a precise relation does not exist but the following generalisation holds [93]:

- Low scale $a \Rightarrow$ Compressed wavelet $\Rightarrow$ Rapidly changing $\Rightarrow$ High frequency ω
- High scale $a \Rightarrow$ Stretched wavelet $\Rightarrow$ Slowly changing $\Rightarrow$ Low frequency ω

Wavelets are essentially mathematical functions which satisfy certain criteria and are used to analyse a signal of interest at different scales (resolutions) which leads to time-scale representations. Some common wavelets are shown in figure 2.19.

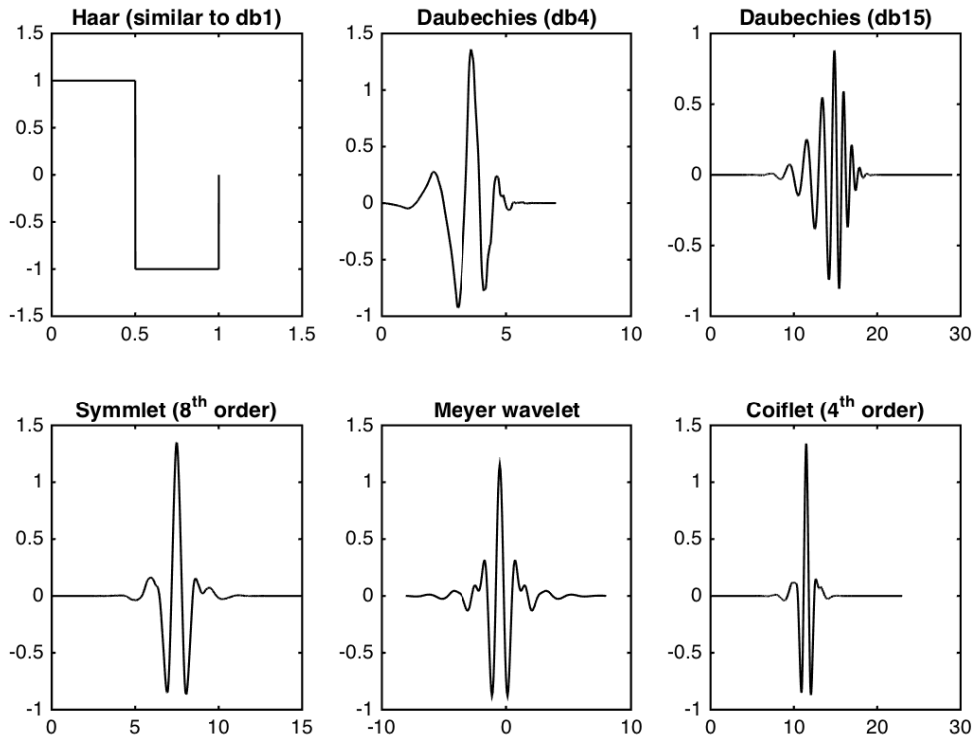


Figure 2.19: Common wavelets.

In a similar fashion to FT the continuous wavelet transform (CWT) of a signal $x(t)$ can be defined by:

$$\boxed{CWT\{x\} \stackrel{\text{def}}{=} T(a,b) = \int_{-\infty}^{\infty} x(t)\psi^*_{a,b}(t)dt = \langle x(t), \psi_{a,b}(t)\rangle} \tag{2.71}$$

where $T(a,b)$ are the wavelet coefficients, * denotes complex conjugation and $\psi_{a,b}(t)$ are the wavelets (i.e. the basis functions) defined by:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} h\left(\frac{t-b}{a}\right) \tag{2.72}$$

where, a is the scale factor, $\frac{1}{\sqrt{a}}$ is an normalisation constant used for energy conservation and b is the current position in time. Equation (2.71) is also written using the inner product notation which emphasizes the idea of the transform as a similarity measure between the signal and the basis functions. The wavelets coefficients $T(a,b)$ are redundant (since a and b are continuously varying) and depending on the input signal and the wavelet that is used, they can be either real or complex. In particular the CWT of a real input signal will produce a real output, but if the wavelet is complex then the output of the CWT will be a complex-valued function of scale and position [93].

From the CWT coefficients it is possible to get the original signal back using the inverse CWT (ICWT):

$$ICWT\{T\} \stackrel{\text{def}}{=} x(t) = \frac{1}{C_h} \int_{-\infty}^{\infty} \int_{a>0}^{\infty} T(a,b)\psi_{a,b}(t) \frac{dadb}{a^2} \tag{2.73}$$

$$C_h = \int_{-\infty}^{\infty} \frac{|\widehat{H(\omega)}|^2}{|\omega|} < \infty \tag{2.74}$$

where $\widehat{H(\omega)}$ is the FT of $\psi_{a,b}(t)$. Equation (2.74) is known as the admissibility condition and C_h is called the admissibility constant and its value depends on the chosen wavelet. The inverse CWT exists only if C_h exists and this happens when the wavelet $\psi_{a,b}$ has zero mean [116]. A possible CWT algorithm could be the following [93]:

Algorithm 2 CWT

1: Choose a wavelet and place it on the starting position of a signal ($b = 0$).
2: Calculate the inner product between the wavelet and the signal at this position. The bigger the inner product the more similar the wavelet and the signal are. This is one wavelet coefficient.
3: Delay the wavelet (i.e. shift to the right, $b > 0$) and repeat steps 2 to 3 until the whole signal is covered.
4: Stretch the wavelet (i.e. increase the scale $a > 1$) and repeat steps 1 to 3.
5: Repeat steps 1 to 4 for all scales.

It should be noted that the wavelet coefficients depend on the chosen wavelet and will be different for different wavelets. Similarly to the STFT spectrogram, the squared modulus of the coefficients can be calculated in order to produce a CWT spectrogram or 'scalogram' as it is commonly called. Figure 2.20 depicts a scalogram of the 'quake' signal. The CWT for that particular figure was computed using a symlet of 14^{th} order and 318 scales.

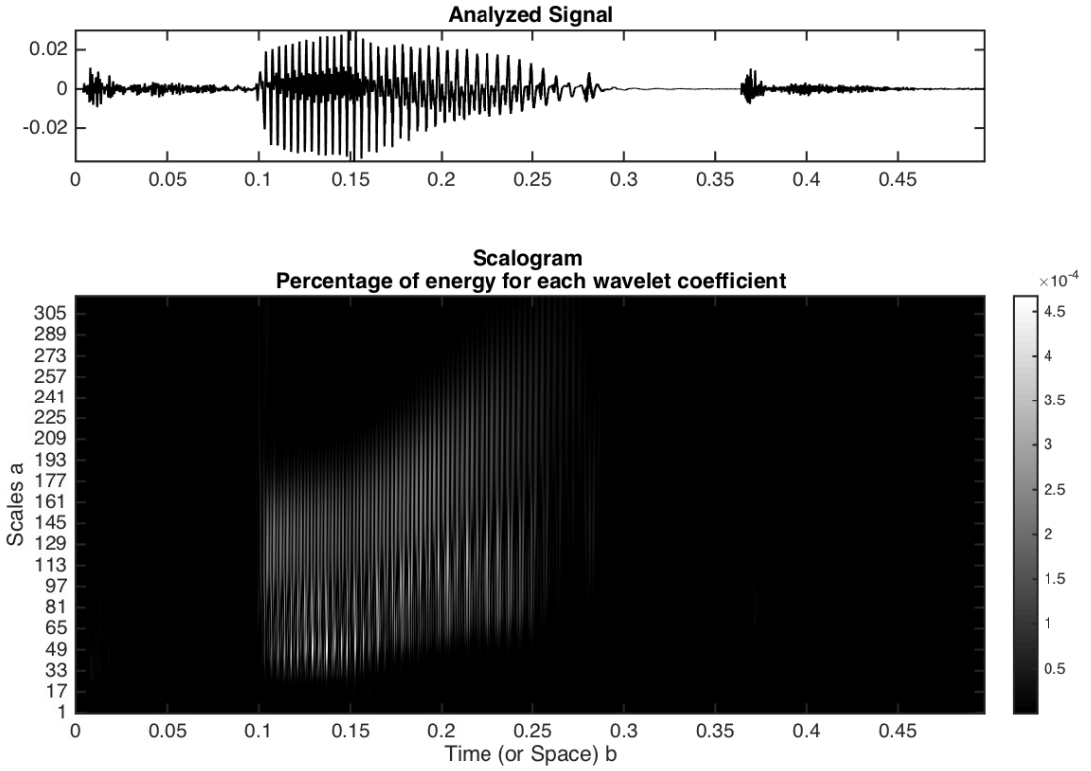


Figure 2.20: Scalogram of 'quake' signal.

The CWT is defined for continuous varying scales and translations but in digital systems the discrete wavelet transform (DWT) is used. The DWT can be calculated in a similar fashion to the CWT with the difference that the time and scale parameters are discretised. Note that this is analogous to the Fourier series. One way is to use a logarithmic discretisation of scale parameter a and link this to the translation parameter b leading to the following wavelets [116] :

$$\psi_{j,k}(t) = \frac{1}{\sqrt{a_0^j}} h\left(\frac{t - kb_0a_0^j}{a_0^j}\right) \tag{2.75}$$

where integers j and k control the scale and translation of the wavelet respectively, $a_0 > 1$ is the fixed dilation step and $b_0 > 0$ is the time-location parameter. If we use $a_0 = 2$ and $b_0 = 1$ then we get the dyadic DWT (scaled in powers of two). Substituting these values into (2.75) we get:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}} h\left(\frac{t - k2^j}{2^j}\right) \tag{2.76}$$

or more compactly:

$$\psi_{j,k}(t) = 2^{-j/2} h(2^{-j}t - k) \tag{2.77}$$

and the DWT is calculated as:

$$DWT\{x\} \stackrel{\text{def}}{=} T(j,k) = \int_{-\infty}^{\infty} x(t)\psi_{j,k}^*(t) \tag{2.78}$$

The wavelets in the dyadic DWT are usually constructed to be orthonormal. Using an orthonormal basis the original signal can be reconstructed using the inverse DWT:

$$IDWT\{T\} \stackrel{\text{def}}{=} x(t) = \sum_j \sum_k T_{j,k}\psi_{j,k}(t) \tag{2.79}$$

Although the DWT can be calculated with the algorithm described earlier, in practice a much faster algorithm is preferred. In 1988, Stephane Mallat produced a fast algorithm for wavelet decomposition and reconstruction which connects wavelets with the theory of filter banks and in particular conjugate quadrature filters (or quadrature mirror filters) that were described briefly in section 2.1.4. In short the decomposition stage of the algorithm involves successive high-pass and low-pass filtering operations on the input signal using a set of filters called the decomposition filters, followed by decimation which yields approximations (low-passed versions of the signal) and details (high-passed versions of the signals) of the original signal. The reconstruction algorithm is the reversed procedure starting with the details and approximations of the decomposition stage and applying up-sampling and successive low-pass and high-pass operations using another set of filters called the reconstruction filters. Because of the decimation step this transform is sometimes referred to as the decimated DWT.

Orthonormal dyadic discrete wavelets, like the ones in (2.77), are associated with a scaling function $\phi(t)$ and their dilation equation. The scaling function is calculated using the dilation equation:

$$\phi(t) = 2\sum_{k=0}^{N} h_0(k)\phi(2t - k) \tag{2.80}$$

where $\phi(t)$ is the scaling function, $h_0(k)$ are the low-pass filter coefficients and N is the total number of filter coefficients. Equation (2.80) is recursive and shows that we can calculate a scaling function at a certain scale from a number of scaling equations at previous scales. Solution of the dilation equation produces the scaling function. The filter is called the scaling filter and is a low-pass FIR filter of length $2N$ with unit norm. Having obtained the scaling function, the wavelet function $\phi(t)$ can be calculated with the wavelet equation:

$$\psi(t) = 2\sum_{k=0}^{N} h_1(k)\phi(2t-k) \tag{2.81}$$

where $\psi(t)$ is the wavelet function, and $h_1(k)$ are the high-pass filter coefficients. It can be seen that once the scaling function is known the mother wavelet can be calculated without recursion.

Wavelet filter banks are designed to obey the PR conditions so that reconstruction of the original signal is possible. Wavelet filter banks produce a 'wavelet tree' similar to that shown in figure 2.12. The frequency content of an input signal is split into lower and upper half; this is the first level of the decomposition. In the second level the newly created signals are first down-sampled and then filtered again and the process is repeated up to the desired decomposition level. In Mallat's algorithm after the first level and for every subsequent level only the low-passed signals are processed which produces the wavelet tree depicted in figure 2.21.

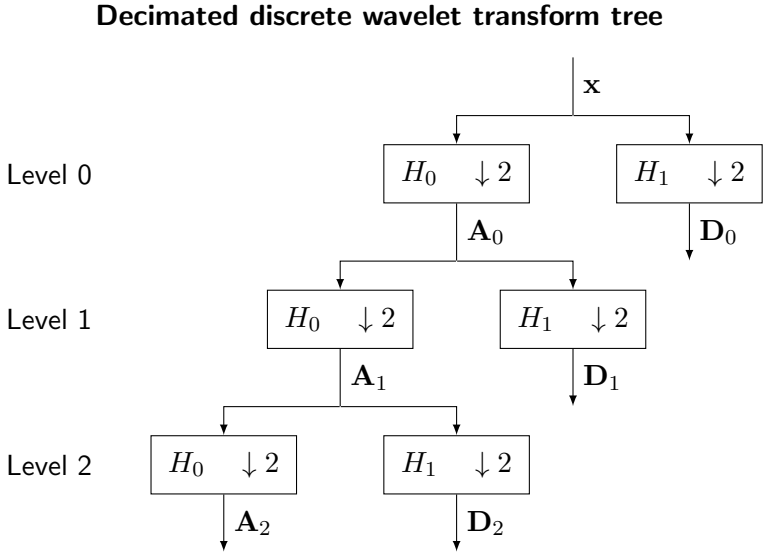


Figure 2.21: Decimated DWT tree. This is essentially the wavelet decomposition algorithm proposed by Mallat. The approximation ($\mathbf{A}$) coefficients and the detail ($\mathbf{D}$) coefficients are the same as the ones produced by the dyadic DWT. It should be noted that at each level only the the approximations get filtered.

In the wavelet tree the outputs of each filter are termed details 'D' (outputs of high-pass filters) and approximations 'A' (outputs of low-pass filters) and these are essentially the wavelet coefficients. This scheme has exactly the same output as the DWT. Comparing the wavelet tree with the DWT, we see that the details coefficients are the T(j ,k) coefficients of the DWT and the level of the tree corresponds to index j. Wavelet re-synthesis follows the same logic as explained with the filter banks.

A generalisation of the decimated DWT is the wavelet packet decomposition (WPD). The difference is that in WPD both the details and approximations are split which produces a wavelet packet tree as shown in figure 2.22. This decomposition allows for a richer signal analysis than the dyadic decimated DWT. In particular for an n level decomposition there are $2^{2^{n-1}}$ different ways to represent a signal. The interesting aspect of this method is that we can choose an optimal decomposition of the original signal based on the minimisation of a convenient criterion. Because of the binary tree structure classical entropy-based criteria can be used to find an optimal decomposition. One of the wavelet packet decompositions corresponds to the dyadic DWT; although this might not be the optimal one. These ideas lead to signal adaptive decompositions which will be discussed in a later section.

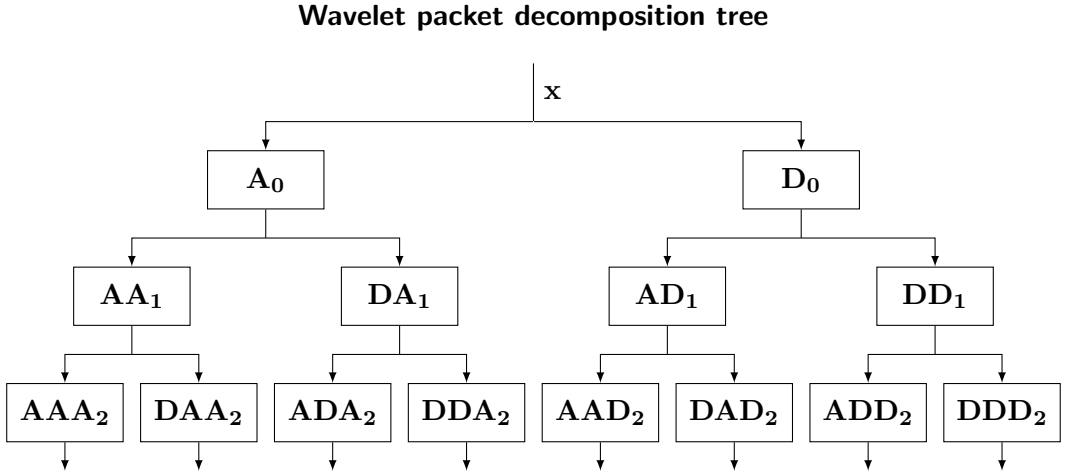


Figure 2.22: Wavelet packet decomposition tree. The main difference to the decimated DWT is that both details and approximations are calculated, leading to more coefficients. Clever schemes allow the best (in some sense) selection of wavelet coefficients to represent the signal.

From the discussion so far, it should be clear that wavelet transforms play an important role in signal processing. In particular the WT introduces the intuitive notion of scale which leads to novel signal representations and is better suited for the analysis of transient phenomena and signal discontinuities. It can also be classified as a constant Q technique where low frequencies use narrow analysis bands and high frequencies use wider analysis bands, or in other words long windows are used for low frequencies (good frequency resolution) and narrow windows for high frequencies (good time resolution). This is in contrast to the STFT described in section 2.1.5 where the size of the window is fixed (i.e. single scale) therefore both time and frequency resolutions are fixed.

A constant Q analysis is similar to the analysis performed by the human auditory system and in certain applications this is a desirable property. The interpretation of the WT as a bank of constant relative bandwidth band-pass filters leads to fast computational algorithms which allow the DWT to be used in real-time systems. Also there exists a plethora of ready-made wavelets to choose from, each with different properties and suited for different applications. Which wavelet to use, depends to a great extent on the type of the signal in question. For example a picture usually has sharp edges which can better be approximated with a short and rough wavelet, whereas a quasi-stationary musical signal would be better approximated by a longer, smooth wavelet.

One important property determining accuracy is the number of vanishing moments of the wavelet. In general a wavelet with p vanishing moments can approximate a polynomial of order $p-1$

[94]. Some other properties that relate to wavelet functions are orthogonality, which states if the filter bank is orthogonal or bi-orthogonal, symmetry which leads to symmetric linear-phase filters and compact support which leads to FIR filters. The selection of an appropriate wavelet should depend mainly on these properties. Finally it is important to note that the better the approximation, due to a carefully selected wavelet, the sparser the representation of the signal would be. An intuitive explanation is that when the applied wavelet does not match the underlying signal then many non-zero wavelet coefficients are needed for approximating it. So a wavelet that matches the 'characteristics' of a signal would produce few significant coefficients, thus a sparser representation.

Despite the interesting benefits (compared to FT and STFT) that wavelets provide to signal analysis, they have some drawbacks some of which are similar in nature to the STFT. First of all, although both the CWT and DWT are linear transforms, which is a desirable property, they are shift-variant which means that a small shift of the input signal will produce wavelet coefficients that vary substantially. Also the non-ideal transition bands of the filters used in the decimated DWT would result in aliasing when the signal is down-sampled, which is cancelled by the inverse DWT provided that the coefficients have not changed. Any change in the wavelet coefficients would result into artifacts in the reconstructed signal. Kingsbury has proposed the dual-tree complex wavelet transform (DT-CWT) and its wavelet packet generalisation the dual-tree complex wavelet packet decomposition (DT-CWPD) which alleviate the problems of shift-variance and aliasing to an extend and also produce complex-valued coefficients which can provide the magnitude and phase information of a signal similar to the FT. Generally speaking though it is very difficult to construct wavelets that comprise all desired properties for a particular task. Finally, although the WT performs a constant Q analysis which leads to a more natural segmentation of the time-frequency plane, it is not adaptive to the signal, in the sense that the configuration of the time-frequency tiles remains the same for the duration of the signal, similarly to the STFT. Signal adaptivity can be achieved to a certain extend using wavelet packets but even so these do not change over time. Adaptive signal decompositions will be discussed in a later section.

2.1.8 Linear algebra essentials

In section 2.1.1 signals were described both as functions and vectors and the importance of linear algebra and functional analysis in digital signal processing was outlined. In particular a discrete-time sequence was represented as a linear combination of delayed impulses, the convolution and cross-correlation operators were introduced and discussed and signals were described as vectors in an n-dimensional space. In section 2.1.3 the inner-product between two functions was defined. This section aims to better show the connection between those concepts using linear algebra notation. Linear algebra can be thought of as the mathematical language to express discrete-time signals and realisable digital systems. Only necessary concepts will be introduced. The following material can be found for example in [90] and [87].

Vectors

A vector is a mathematical object that comprises components (or elements) which can be numbers, symbols or expressions. A pair of real numbers v_1 and v_2 produce a two-dimensional vector:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = (v_1, v_2) \tag{2.82}$$

When all components of a vector are zero then $\mathbf{v}$ is the zero vector denoted by $\mathbf{0}$. Vectors will be denoted with boldface lower letters. Equation (2.82) represents a two-dimensional column vector. Note that the notation using parentheses still denotes a column vector. A row vector can be written using brackets and without commas in-between the elements.

Vectors can be added together and multiplied by scalars producing linear combinations (or weighted summations). If $\mathbf{v}$ and $\mathbf{w}$ are two-dimensional vectors and c and d are real scalars then the following is a linear combination of $\mathbf{v}$ and $\mathbf{w}$:

$$c\mathbf{v} + d\mathbf{w} \tag{2.83}$$

A special operation between vectors is the dot product defined by:

$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{n} v_i w_i = \mathbf{v}^T \mathbf{w} = \langle \mathbf{v}, \mathbf{w} \rangle \tag{2.84}$$

where v_i and w_i are the components of vectors $\mathbf{v}$ and $\mathbf{w}$ respectively, n is the number of components in each vector and the symbol $'\cdot'$ denotes the dot-product operation. The dot product is similar to the inner product in equation (2.42). The inner product is a generalisation of the dot-product and is mostly used in functional analysis but in the space covered by real-valued vectors comprising n components (i.e. a Euclidean vector space) the inner product becomes the dot-product. For the purposes of this thesis the angle bracket notation on the right of equation (2.84) will be used and not the $'\cdot'$ symbol. The dot-product has a geometrical interpretation. In particular the dot product is the magnitude of the projection of one vector into another. This can be expressed by the cosine formula [90]:

$$cos(\theta) = \frac{\langle \mathbf{v}, \mathbf{w} \rangle}{\|\mathbf{v}\| \|\mathbf{w}\|} \tag{2.85}$$

where θ is the angle between the two vectors. Given two vectors and their norms (defined below), the angle can be found from equation (2.85) by getting the inverse cosine of the output. A zero dot product indicates that the vectors are perpendicular to each other (i.e. $cos(\theta) = 0 \Rightarrow \theta = \arccos(0) \Rightarrow \theta = 90°$).

The length or norm of a vector $\mathbf{v}$ can be defined by:

$$\text{lentgh}(\mathbf{v}) = \|\mathbf{v}\| \stackrel{\text{def}}{=} \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{v_1^2 + \cdots + v_n^2} \tag{2.86}$$

Equation (2.86) is also known as the Euclidean length , ℓ^2 distance or ℓ^2 norm. The term 'norm' will be used for denoting the ℓ^2 norm. A vector of unit length (or unit norm) is called a unit

vector and any vector can be normalised by:

$$\mathbf{u} = \frac{\mathbf{v}}{\|\mathbf{v}\|} \tag{2.87}$$

where $\mathbf{v}$ is the vector to be normalised and $\mathbf{u}$ is a unit vector at the same direction as $\mathbf{v}$. Notice that for unit vectors, the dot product becomes the cosine of the angle between the vectors (the denominator in (2.85) becomes one for unit vectors)

Matrices

A matrix can be thought of as a rectangular array (or table) of elements arranged in rows and columns. The elements can be numbers (real or complex) or other expressions. Matrices will be denoted with boldface capital letters. For example the following is a 2×4 (read as 2 rows by 4 columns or 2 by 4 for short) matrix:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \tag{2.88}$$

where $a_{i,j} = \mathbf{A}(i,j)$ are the matrix elements. Generally we have $N \times M$ matrices with N rows and M columns, with individual elements $a_{i,j}$ for $i = 1, \ldots, N$ and $j = 1, \ldots, M$. If $N = 1$ the matrix becomes a row vector and if $M = 1$ a column vector. Two matrices $\mathbf{A}$ and $\mathbf{B}$ are equal if and only if:

$$a_{i,j} = b_{i,j} \tag{2.89}$$

where $a_{i,j}$ and $b_{i,j}$ are the elements of matrices $\mathbf{A}$ and $\mathbf{B}$ respectively. Two equal size matrices can be added or subtracted, element by element, to produce a new matrix:

$$\mathbf{C} = \mathbf{A} \pm \mathbf{B} \equiv c_{i,j} = a_{i,j} \pm b_{i,j} \tag{2.90}$$

Two matrices can also be multiplied to produce a third matrix:

$$\mathbf{C} = \mathbf{AB} \equiv c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j} \tag{2.91}$$

that is in order to obtain element $c_{i,j}$, the i^{th} row of $\mathbf{A}$ is multiplied, element by element with the j^{th} column of $\mathbf{B}$ and the products are summed together. This operation is the dot product defined in (2.84) and as such matrix multiplication can be seen as the dot products between the rows of $\mathbf{A}$ and columns of $\mathbf{B}$. In order for this matrix multiplication to be possible the columns of $\mathbf{A}$ must be the same size as the rows of $\mathbf{B}$. An $N \times M$ matrix multiplied by an $M \times P$ matrix will produce an $N \times P$ matrix. In general $\mathbf{AB} \neq \mathbf{BA}$ (commutative law does not hold) and $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$ (associative law holds) . Another interpretation occurs by considering multiplying a matrix with a vector. An $N \times M$ matrix $\mathbf{A}$ multiplied by an $M \times 1$ column vector $\mathbf{x}$ will produce an $N \times 1$ column vector $\mathbf{b}$:

$$\mathbf{Ax} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + x_3\mathbf{a}_3 = \mathbf{b} \tag{2.92}$$

where $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ are the first, second and third columns of $\mathbf{A}$. In this case, the output vector $\mathbf{b}$ can be interpreted as a linear combination of the columns of $\mathbf{A}$. For three matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and a scalar c, the following multiplication properties hold:

$$c(\mathbf{A}+\mathbf{B}) = c\mathbf{A} + c\mathbf{B} \quad \text{(distributivity of scalar multiplication)} \tag{2.93}$$

$$\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB} + \mathbf{AC} \quad \text{(distributivity of matrix multipliation)} \tag{2.94}$$

The operation of matrix multiplication has more properties but only the ones used in this thesis are presented here.

Of interest is also the Hadamard product which is the element by element (or point wise) multiplication of two matrices (or vectors) of the same size and produces a third matrix of the same size. The Hadamard product for two equal sized matrices $\mathbf{A}$ and $\mathbf{B}$ will be expressed as:

$$\mathbf{C} = \mathbf{A} \odot \mathbf{B} \equiv c_{i,j} = a_{i,j} \cdot b_{i,j} \tag{2.95}$$

where the symbol '$\cdot$' int his case denotes multiplication. In a similar fashion the element by element division of matrices can be expressed as:

$$\mathbf{C} = \mathbf{A} \oslash \mathbf{B} \equiv c_{i,j} = a_{i,j}/b_{i,j} \tag{2.96}$$

A matrix $\mathbf{A}$ can be transposed by interchanging its rows with its columns. A transposed matrix is denoted by $\mathbf{A}^T$. For two matrices $\mathbf{A}$ and $\mathbf{B}$ and a scalar c, the following transpose properties hold:

$$(\mathbf{A}^T)^T = \mathbf{A} \quad \text{(property 1)} \tag{2.97}$$

$$(\mathbf{A}+\mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T \quad \text{(property 2)} \tag{2.98}$$

$$(c\mathbf{A})^T = c\mathbf{A}^T \quad \text{(property 3)} \tag{2.99}$$

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T \quad \text{(property 4)} \tag{2.100}$$

In case the matrix has complex elements, its Hermitian transpose is denoted by $\mathbf{A}^H$ and results from the complex conjugate elements of $\mathbf{A}^T$. The following holds:

$$(\mathbf{AB})^H = \mathbf{B}^H\mathbf{A}^H \tag{2.101}$$

and if $\mathbf{A}$ is real then $\mathbf{A}^H = \mathbf{A}^T$.

Some special matrices are the symmetric matrix, where $\mathbf{A} = \mathbf{A}^T$ (i.e $a_{j,i} = a_{i,j}$), the square matrix where $N = M$ (equal number of rows and columns), the zero matrix where all its elements are equal to zero, the diagonal matrix, where only the elements $a_{i,i}$ are non-zero, the lower-triangular matrix $\mathbf{L}$, where all elements above the diagonal are zero and its diagonal elements are all one and the upper-triangular matrix $\mathbf{U}$, where all elements below the diagonal are zero. A special diagonal matrix is the identity matrix, denoted by $\mathbf{I}$ where its diagonal elements are equal to one. A square matrix can have an inverse denoted by $\mathbf{A}^{-1}$ and the following holds:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \tag{2.102}$$

Systems of linear equations

The main application area of linear algebra is the solution of systems of linear equations of the form:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m &= y_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m &= y_2 \\ &\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m &= y_n \end{aligned} \tag{2.103}$$

where $a_{i,j}$ are the coefficients of the system with $i = 1 \ldots n$ and $j = 1 \ldots m$, the y values are the known outputs and the x values are the unknowns to be found. The system denoted by (2.103) can be re-written using matrix notation:

$$\mathbf{A}\mathbf{x} = \mathbf{y} \tag{2.104}$$

where $\mathbf{A}$ comprises all $a_{i,j}$ system coefficients, $\mathbf{x} = (x_1, x_2 \ldots, x_m)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. Linear systems might have a unique solution, no solution or infinitely many solutions. If $n = m$, then $\mathbf{A}$ is a square matrix and a solution can be found by using the inverse of $\mathbf{A}$ (if it exists):

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} \tag{2.105}$$

In practice finding the inverse of a matrix, especially for large matrices, is a computationally expensive operation and is usually avoided and other numerical methods are used to get a solution. In case $n > m$, then there are more equations than unknowns and the system is overdetermined which might have a solution or not. When $n < m$, there are more unknowns than equations and the system is under-determined in which case it has no solution or infinite number of solutions.

Vector spaces

All aforementioned vector and matrix properties and operations occur within a vector space. A vector space is a set of vectors with rules for addition and multiplication by scalars and some structure provided by the norm and the inner product. The best known vector spaces are the one-dimensional Euclidean space $\mathbf{R}^1$ which is represented by the line of real numbers, the two-dimensional Euclidean space $\mathbf{R}^2$ which is represented by the xy plane and the three-dimensional $\mathbf{R}^3$ which is represented by the xyz space. All linear combinations of one-dimensional vectors fill a line, two-dimensional vectors fill a plane and three-dimensional vectors fill a space. These spaces can be generalised to the n-dimensional Euclidean vector space $\mathbf{R}^n$. The symbol $\mathbf{R}$ denotes that the components of the vectors are real numbers. We can also have vector spaces in $\mathbf{C}^n$ where the vector components are complex numbers. Generally all operations occurring in a vector space must produce results that stay within that space. $\mathbf{R}^n$ can be defined as the space that comprises all column vectors $\mathbf{v}$ with n components [90].

Vector spaces found within other vector spaces are called subspaces. For example if we start in $\mathbf{R}^3$ and choose a plane that passes through the origin $(0,0,0)$ then that plane is a subspace of $\mathbf{R}^3$. Similarly lines passing through the origin $(0,0,0)$ are subspaces of $\mathbf{R}^3$. Every subspace must pass through the origin or in other words every subspace must contain the zero vector of the initial space [90]. More formally a subspace of a vector space is a set of vectors, including the zero vector, which is 'closed' under addition and multiplication [90], meaning that the results of any linear combinations of vectors in the subspace remain in that subspace. Regarding a matrix $\mathbf{A}$ an important subspace is the column space of $\mathbf{A}$, denoted by $\mathbf{C}(\mathbf{A})$, which consists of all the linear combinations of the columns of $\mathbf{A}$ (i.e. $\mathbf{Ax}$). This space is crucial because the linear system in equation (2.104) has a unique solution only if the right hand side vector $\mathbf{y}$ is in the column space of $\mathbf{A}$. Another important subspace is the null-space of $\mathbf{A}$, which comprises all solutions to $\mathbf{Ax} = \mathbf{0}$ and is denoted by $\mathbf{N}(\mathbf{A})$. A trivial solution to $\mathbf{Ax} = \mathbf{0}$ is $\mathbf{x} = \mathbf{0}$ and will always be included in the null-space.

Basis vectors and independence

Every vector space has at least one basis. A basis is a set of vectors that are linearly independent and span the vector space. Formally, given a matrix $\mathbf{A}$, we say that the columns of $\mathbf{A}$ are linearly independent when the only solution to $\mathbf{Ax} = \mathbf{0}$ is $\mathbf{x} = \mathbf{0}$ [90]. In other words no other combination of $\mathbf{Ax}$ gives the zero vector. A basis needs to span the space which means that any vector in that space can be written as a linear combination of the basis vectors. An important consequence of this is that any given vector has a unique representation in terms of the basis of that space. Also the dimension of the space is given by the number of independent columns of $\mathbf{A}$. To demonstrate with an example, imagine the $\mathbf{R}^2$ vector space which represents the xy plane. The following vectors form a basis in $\mathbf{R}^2$:

$$\mathbf{v}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = (1,0), \quad \mathbf{v}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = (0,1) \tag{2.106}$$

Vectors $\mathbf{v}_0$ and $\mathbf{v}_1$ are linearly independent since $\mathbf{Ax} = \mathbf{0}$ can only be solved when $\mathbf{x} = \mathbf{0}$. Also any vector in $\mathbf{R}^2$ can be written as a linear combination of $\mathbf{v}_0$ and $\mathbf{v}_1$, thus vectors $\mathbf{v}_0$ and $\mathbf{v}_1$ are a basis in $\mathbf{R}^2$. Consequently the dimensionality of $\mathbf{R}^2$ is 2. A vector space can have multiple bases, but all bases must have the same number of vectors. The $\mathbf{R}^n$ space always has a basis with n vectors, the standard basis, thus the dimensionality of $\mathbf{R}^n$ spaces will always be n. It should be noted that the aforementioned basis vectors are also orthogonal to each other, that is their dot product is zero, which is a special case of linear independence, although a basis in general does not have to be orthogonal. More specifically the standard basis in $\mathbf{R}^n$ is orthonormal which means that the vectors are orthogonal and of unit norm.

So far many bases have been introduced. For example in section 2.1.1 a discrete-time sequence was represented by a linear combination of delayed impulses. Equation (2.3) can also be written using the matrix notation that was introduced earlier:

$$y[n] = \sum_{k=0}^{N-1} x[k]\delta[n-k] \Rightarrow \mathbf{y} = \mathbf{Ax} \tag{2.107}$$

where $\mathbf{x}$ are the scalars (weights or coefficients $x[k]$) and $\mathbf{A}$ comprises all vectors that represent delayed impulses $\delta[n-k]$. The delayed impulses are the basis vectors, that is the standard orthogonal basis in $\mathbf{R}^N$. In a similar fashion, the Fourier series and the FT represent signals as linear combinations of harmonically related trigonometric functions and complex exponentials, which sets are also orthogonal and form bases over their spaces. As already mentioned signals are uniquely represented by a basis.

Orthogonal vectors and projections

In section 2.1.3, equation (2.43) describes an orthogonal set of functions, where the inner-product between any function and all other functions in the set is zero. In linear algebra we talk about orthogonal vectors. In particular a couple of vectors $\mathbf{x}$ and $\mathbf{y}$ are orthogonal if their dot-product is zero:

$$\mathbf{x}^T\mathbf{y} = 0 \tag{2.108}$$

where the notation of (2.108) stems from the rules of matrix multiplication. Orthogonality, of two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbf{R}^2$ is depicted in figure 2.23 below.

Orthogonal vectors

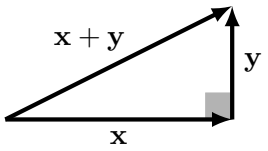


Figure 2.23: Orthogonal vectors in $\mathbf{R}^2$. The angle between vectors $\mathbf{x}$ and $\mathbf{y}$ is 90° so their dot product, calculated using (2.84) is zero.

Equation (2.84) can be easily calculated from Pythagoras theorem, which for any right-angled triangle can be expressed as:

$$\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 = \|\mathbf{x}+\mathbf{y}\|^2 \tag{2.109}$$

where $\|\mathbf{x}\|^2$,$\|\mathbf{y}\|^2$ and $\|\mathbf{x}+\mathbf{y}\|^2$ are the lengths of vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{x}+\mathbf{y}$ respectively. It should be noted that although the picture in figure 2.23 is in $\mathbf{R}^2$, equation (2.109) can be applied to vectors in any dimension. And this true in linear algebra in general; operations in two-dimensional vectors easily translate into vectors of higher dimensions.

Another important operation directly related to orthogonality is vector projections. We are generally interested in projecting a vector $\mathbf{y}$ into a vector $\mathbf{x}$, the geometry of which, in two dimensions, is depicted in figure 2.24.

In figure 2.24, $\mathbf{p}$ is the projection of $\mathbf{y}$ onto $\mathbf{x}$ and $\mathbf{e} = \mathbf{y} - \mathbf{p}$ is the error. In order for the error to be minimum it has to be orthogonal to $\mathbf{x}$, that is $\mathbf{x}^T\mathbf{e} = 0$, and if that is the case then we can say that the projection $\mathbf{p}$ is the optimal representation of $\mathbf{y}$ in terms of $\mathbf{x}$. From figure 2.24 and the orthogonality principle expressed by equation (2.108) it is easy to show that the projection $\mathbf{p}$ can be given by:

$$\mathbf{proj} \quad \mathbf{p} = \mathbf{P}\mathbf{y} \tag{2.110}$$

$$\mathbf{P} = \frac{\mathbf{x}\mathbf{x}^T}{\mathbf{x}^T\mathbf{x}} \tag{2.111}$$

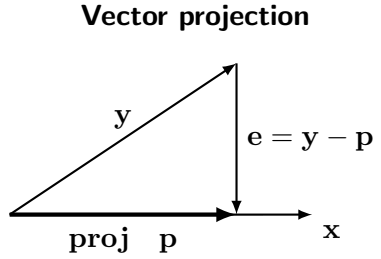


Figure 2.24: Vector projection. Vector $\mathbf{y}$ is projected down to vector $\mathbf{x}$

where $\mathbf{P}$ is called the projection matrix, with its numerator being a matrix and the denominator a number.

The main purpose of projections is the solution of equation (2.104) when there are more equations than unknowns (non square matrix $\mathbf{A}$). In this case equation (2.104) might have no solutions so the goal is to solve the closest problem that can be solved, which is:

$$\mathbf{A}\hat{\mathbf{x}} = \mathbf{p} \tag{2.112}$$

where $\mathbf{p}$ is the projection of vector $\mathbf{y}$ (of equation (2.104)) onto the column space of $\mathbf{A}$ and $\hat{\mathbf{x}}$ is the approximate solution. This fundamental idea of projections is used in many problems and is one of the main steps that occurs in matching pursuit algorithms that will be discussed in a later section.

In this section the connection between functions and vectors was made by introducing fundamental concepts in linear algebra such as the dot product, vector spaces, basis vectors, vector space dimensionality, orthogonal vectors and projections. Of course there is a lot more to say on the theory of vectors and vector spaces but this section has covered all concepts that are essential for the work presented in this thesis.

2.2 Sparse Representations

So far, several signal representations have been discussed. Earlier sections explained how a discrete-time signal can be represented in the time domain via sampling of a continuous-time signal, in the frequency domain via the use of the Fourier series and the Fourier transform, in the time-frequency domain using the STFT and spectral modelling and in the time-scale domain using the wavelet transform and its variations. No matter how a signal is represented there is a main concept behind each representation and that is the idea of expressing a signal in terms of a linear combination of elementary signals. These elementary signals are the delayed impulses in the time-domain, the Fourier basis in the frequency domain, the 'logons' of Gabor and quasi-stationary sinusoids in spectral modelling in the time-frequency domain and wavelets in the time-scale domain. But many of the techniques described have a main limitation in representing signals and that is, they are not adaptive to the signal. This 'non-adaptivity' is a consequence of the type of elementary functions (i.e. bases) that are used for the representation which dictate the way the time-frequency plane is segmented. Figures 2.13 and 2.18 depict the time-frequency segmentation that the STFT and WT produce respectively and it can be seen that this segmentation is fixed

as soon as the main parameters of each algorithm are selected. Representing a signal over a single basis is not flexible enough. For example a Fourier basis is not good in representing signals localised in time (i.e. transient phenomena) whereas wavelet bases are not good in representing signals with narrow high-frequency support. This limitation is the main motivation behind signal decompositions over large and redundant sets of elementary waveforms, that are capable of capturing a wide variety of patterns found in signals. Such decompositions are known as *sparse decompositions* and lead to *sparse signal representations* or *sparse approximations*.

Sparse decomposition techniques underwent considerable development during the 1990's with the advent of wavelet processing [63]. In audio processing, sparse signal representations have been applied to a variety of tasks, such as audio coding [117, 118, 119, 120, 121, 122, 123, 112], audio de-noising [124, 125], audio/musical analysis and transformation [126, 127, 128, 129, 130, 131], audio detection/discrimination and classification [132, 133, 134] and audio source separation [52, 53, 54, 55, 56, 57, 58].

As already explained in section 1.4.3, obtaining a sparse representation (or approximation) is the first step in a SCA-based source separation system. The rest of this section presents the basic principles of sparse representations and briefly discusses several methods that are available for the solution of this problem.

2.2.1 Sparse signal representations

Sparse decomposition techniques aim to expand a signal into a linear combination of elementary waveforms, called *atoms*, that are chosen from a large and usually redundant set of such atoms, called *dictionary*. Mathematically this can be expressed as:

$$x(t) = \sum_{k=1}^{K} c_k \varphi_k(t) + \epsilon(t) \tag{2.113}$$

where $x(t)$ is the signal of interest, c_k are the expansion coefficients, $\varphi_k(t)$ are the atoms, and $\epsilon(t)$ is an error term to account for noise and model inaccuracies. The representation of $x(t)$ is an *exact signal representation* when $\epsilon = 0$ or an *approximate signal representation* when $\epsilon > 0$. Using linear algebra notation, equation (2.113) can be written as:

$$\mathbf{x} = \mathbf{\Phi c} + \boldsymbol{\epsilon} \tag{2.114}$$

where $\mathbf{\Phi}$ is the dictionary of atoms, that could be either real or complex, and $\mathbf{c}$ is the vector of expansion coefficients. In practice, the dictionary $\mathbf{\Phi}$ is an $N \times K$ matrix where each column represents an atom (thus there are K atoms in the dictionary) and N represents the length of the atoms and also the vector $\mathbf{x}$. When $N == K$, then $\mathbf{\Phi}$ is complete and becomes a square matrix and if the atoms form a basis and the noise term is set to zero, then (2.114) has the trivial solution of $\mathbf{c} = \mathbf{\Phi}^{-1}\mathbf{x}$ which is unique. If the atoms also form an orthonormal basis then the solution is simplified even further since $\mathbf{\Phi}^{-1} = \mathbf{\Phi}^{T}$, thus avoiding matrix inversion. In sparse decompositions though, the dictionary $\mathbf{\Phi}$ is over-complete (therefore redundant), that is $K >> N$, which implies that its columns (i.e. the atoms) are linearly dependant and as such the solution in (2.114) is non-unique. This 'non-uniqueness' is what leads to *adaptive representations*;

essentially there are many possible atom combinations that can represent the original signal. The problem then becomes that of finding the 'best' possible combination of atoms that leads to a desired representation. The 'best' solution depends on the application at hand and is dictated by the constraints of the problem but generally speaking sparse decompositions try to achieve highly sparse adaptive representations. Some possible approaches are presented in section 2.2.4.

2.2.2 Sparsity

A set of coefficients $\mathbf{c} = \{c_k\}_{k=1}^{K}$ is considered sparse if only few of the coefficients are significant and most of the them are zero or negligible. In statistical terms, the coefficients $\mathbf{c}$ are sparse if their histogram (or probability density function) follows a super-Gaussian distribution. Sparsity is usually modeled by the family of generalised Gaussian distributions:

$$p_c(c) \propto exp(-\alpha|c|^{\tau}) \tag{2.115}$$

for $0 < \tau \leq 1$, with $\tau = 1$ corresponding to the Laplacian distribution. These distributions have a strong peak at the origin and 'heavy' tails, therefore they are well suited to model sparsity. Figure 2.25 depicts the histogram of the GMP coefficients of the 'quake' signal. GMP is a sparse decomposition technique and is described in full detail in chapters 3 and 4. For this figure the quake signal was decomposed into 512 atoms using a multi-scale Gabor dictionary. As it can be seen in the figure the distribution of the coefficients follows that of a super-Gaussian distribution with a strong peak around zero and heavy tails which implies that this particular decomposition is sparse. For the generalised Gaussian model, the set of coefficients $\mathbf{c}$ has the following distribution [1]:

$$p_c(\mathbf{c}) \propto exp(-\alpha\|\mathbf{c}\|_{\tau}^{\tau}) \tag{2.116}$$

$$\|\mathbf{c}\|_{\tau} := \left(\sum_{k=1}^{K} |c_k|^{\tau}\right)^{\frac{1}{\tau}} \tag{2.117}$$

where $\|\mathbf{c}\|_{\tau}$ denotes the ℓ^{τ} norm. In general, the norm $\|\mathbf{c}\|_{\tau}^{\tau}$ can be used to quantify the sparsity of $\mathbf{c}$. It should be noted that this is a norm (in a correct mathematical sense) only when $1 \leq \tau \leq \infty$ but conventionally the *"ℓ^0 pseudo norm"* counts the number of non-zero coefficients in $\mathbf{c}$ with $c^0 = 1$ if $c > 0$ and $0^0 = 0$ and can be thought of as a direct measure of sparsity.

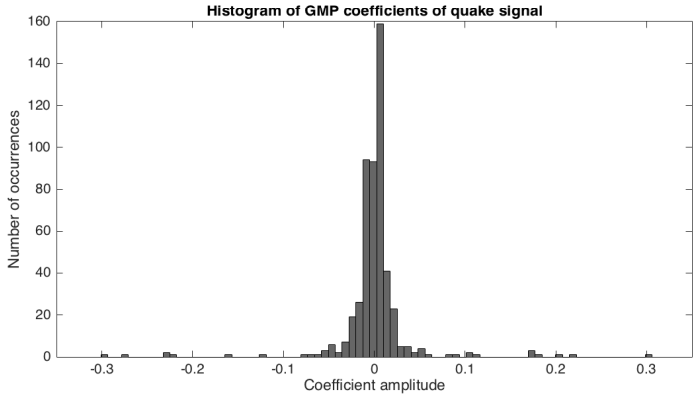


Figure 2.25: Histogram of GMP coefficients of quake signal. Most of the coefficients are 'gathered' around zero and only few have significant amplitude.

2.2.3 Dictionaries

Sparse decompositions are based on over-complete dictionaries to produce sparse adaptive representations (or approximations) that are more flexible than the traditional signal representations presented in earlier sections. It should be clear that the selection of the dictionary in such decompositions is of paramount importance. The dictionary will essentially dictate the way a signal is decomposed and also the amount of sparsity achieved. The design of over-complete dictionaries is a research topic on its own and is usually treated as a separate problem. An excellent review on dictionaries for sparse adaptive representations can be found in [135].

Generally speaking there are two main families of over-complete dictionaries: analytic and trained dictionaries. Analytic dictionaries comprise atoms that can be mathematically defined by a set of parameters, that is they have an analytic expression, and are designed to represent basic signal structures. In many occasions analytic dictionaries are designed in such a way so that they lead to fast implementations of the underlying decomposition algorithms as is the case for example with the Fourier dictionary, the wavelet dictionary and wavelet packets and cosine packets dictionaries. The main drawback is that the underlying model of an analytic dictionary is often too simplistic or generic for capturing the complex patterns found in natural signals.

Trained dictionaries comprise atoms that are 'learned' from a set of training signals. The main idea behind this approach is that the complex patterns of natural signals can be extracted directly from the data rather than using a simplified mathematical model [135]. Such dictionaries allow for a finer adaptation of signal patterns which can lead to sparser representations. But sparsity is not always the most desirable property in a given task. For example in a source separation task using sparse component analysis, a sparse representation of the mixture is desirable as long as the individual sources can be sparsely represented in the new domain, in which case the individual sources can be distinguished and separated in that domain. An overly sparse representation though might have a negative effect in distinguishing the sources since components of multiple sources might be represented by a single atom. On the other hand an overly sparse representation would be beneficial to signal compression tasks. Another potential drawback of trained dictionaries has to do with physical storage of the atoms. Since the trained atoms lack an analytic expression, a decomposition algorithm would have to store the actual data of the atoms leading to increased storage space, in contrast to analytic dictionaries where only the values of the parameter set of the atoms have to be stored. To demonstrate, imagine that a 1024 samples discrete-time signal is decomposed twice using an analytic and a trained dictionary and each decomposition produces 100 atoms. If we assume that the analytic dictionary comprises simple Gabor atoms with four parameters each and that each parameter requires 16 bits of storage space then the storage needed for the analytic atoms is $(100 \times 4 \times 2)/1024 = 0.78$ Kbytes. In contrast the atoms from the trained dictionary have to be stored in whole. Assuming that each audio sample requires 16 bits of storage the trained atoms will occupy $(100 \times 1024 \times 2)/1024 = 200$ Kbytes, that is the analytic atoms occupy only 0.39% of the size of the trained atoms. A final problem posed by trained dictionaries, especially for the purposes of the algorithm presented in this thesis, is that trained atoms are not amendable to modifications. For example a Gabor atom of specific amplitude, frequency, phase and scale can be easily modified and transformed into another atom by simply changing the values of these parameters whereas a trained atom might comprise components of

multiple amplitudes, frequencies, phases and scales and modifying these individual parameters would require further analysis of that atom.

The algorithm presented in chapter 3 heavily depends on analytic dictionaries and some of them will be described in more detail in section 3.6.

2.2.4 Approaches to the sparse approximation problem

The problem of sparse signal approximation deals with finding a solution to equation (2.114) using a sparsity constraint. That is, the 'best' solution is the one that has the fewest non-zero coefficients. Using the ℓ^0 pseudo-norm as the measure of sparsity, the problem can be expressed as an optimisation problem:

$$\arg\min_{\mathbf{c}} \|\mathbf{c}\|_0 \quad \text{subject to} \quad \|\mathbf{x} - \mathbf{\Phi c}\|_2^2 \leq E \tag{2.118}$$

where E is a quantity proportional to the noise variance or zero in the case of a noiseless model [136]. Solving this problem would require to enumerate all possible combinations of the columns of $\mathbf{\Phi}$ which has been shown to be an NP-hard problem [137], therefore sub-optimal strategies are employed. Generally speaking there are two main strategies that are used in practice. The first one is based on minimisation of an ℓ^τ criterion for $0 \leq \tau \leq \infty$, which can lead to both convex and non-convex optimisation problems. The second strategy involves *greedy algorithms* where a solution is built iteratively by selecting one coefficient in $\mathbf{c}$ per iteration, in such a way so that the approximation error is reduced to the largest extent.

In the ℓ^τ minimisation approach it is common to replace the ℓ^0 pseudo norm by another sparsity measuring function which is easier to minimise. Because of this "relaxation" of the problem such approaches are also referred to as *relaxed*. Let us assume a sparsity measuring function $f(\mathbf{c})$, then equation (2.118) can be written as:

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} f(\mathbf{c}) \quad \text{subject to} \quad \|\mathbf{x} - \mathbf{\Phi c}\|_2^2 \leq E \tag{2.119}$$

or in the equivalent unconstrained form [136]:

$$\hat{\mathbf{c}}^\lambda = \arg\min_{\mathbf{c}} \frac{1}{2}\|\mathbf{x} - \mathbf{\Phi c}\|_2^2 + \lambda f(\mathbf{c}) \tag{2.120}$$

The first term in (2.120) is a data fidelity term that measures the quality of the approximation. The second term, $f(\mathbf{c})$, is referred to as the *regulirisation* term and measures the sparsity of the representation $\hat{\mathbf{c}}$ and λ is a scalar parameter controlling the amount of regularisation, effectively determining the trade-off between a sparse solution and a solution with a low approximation error (i.e. high quality). By setting $f(\mathbf{c})$ to the norm $\|\mathbf{c}\|_\tau^\tau$, then (2.120) becomes:

$$\hat{\mathbf{c}}^{\tau,\lambda} = \arg\min_{\mathbf{c}} \frac{1}{2}\|\mathbf{x} - \mathbf{\Phi c}\|_2^2 + \lambda \sum_{k=1}^{K} |c_k|^\tau \tag{2.121}$$

It should be noted that the criterion in (2.121) applies to a single-channel vector $\mathbf{x}$ but as explained in section 1.4.3, the majority of SCA-based systems assume multi-channel mixtures (at least two),

so the regularisation term in (2.121) has to change in order to accommodate the multi-channel case. One possible choice is $f(\mathbf{c}) = \sum_{k=1}^{K}\sum_{m-1}^{M}|c_{km}|^{\tau}$ but in this case the criterion decouples optimisation over the different channels. Instead, a *joint sparse approximation* could be achieved with the optimisation of the following criterion [1]:

$$\hat{\mathbf{C}}^{\tau,\lambda} = \arg\min_{\mathbf{C}} \left\{ \|\mathbf{X} - \mathbf{\Phi}\mathbf{C}\|_F^2 + \lambda \sum_{k=1}^{K} \left(\sum_{m=1}^{M} |c_{km}|^2 \right)^{\frac{\tau}{2}} \right\} \tag{2.122}$$

where now $\mathbf{X} \in \mathbb{R}^{N\times M}$ and $\mathbf{C} \in \mathbb{R}^{K\times M}$ with M being the number of channels and $\|\mathbf{Y}\|_F^2 = \sum_{m=1}^{M}\|\mathbf{y}_m\|^2$ denotes the Frobenius norm of matrix $\mathbf{Y}$ (i.e. the sum of the energies of its columns). It should be noted that although all aforementioned criteria use the ℓ^{τ} norms as the regularisation term, other sparsity measures could be used (e.g. Shannon entropy [138]).

The selection of the τ parameter plays an important role in the optimisation problems discussed. In particular, different values of τ lead to different optimisation problems and different algorithms for their solution. When $\tau < 1$ the resulting optimisation problem is non-convex and thus there is the risk of obtaining a local minimum instead of a global solution. For $1 < \tau \leq 2$ we get convex optimisation problems which are easier to solve and also their convergence to a unique global optimum is guaranteed. Nevertheless both convex and non-convex problems have been extensively studied in the literature. Another important factor is the selection of the regularisation parameter λ. Especially in the context of blind source separation, choosing an appropriate λ can be complex. A very high value for λ will produce a highly sparse representation but will compromise the quality of the reconstruction whereas a very low value could falsely lead to the assumption that the model is accurate and noiseless thus leading to artifacts [1].

Of particular interest is the minimisation of the ℓ^1 criterion because it often leads to sparse solutions. Optimising (2.119) using the ℓ^1 norm as the regularisation term (i.e. the norm for $\tau = 1$) is known as *basis pursuit* (BP) when $E = 0$ and *basis pursuit denoising* when $E \neq 0$ [64]. ℓ^1 optimisation is attractive because it leads to convex optimisation, which is easier to solve and also the ℓ^1 norm promotes sparsity. In fact the authors in [139] have shown that ℓ^1 minimisation leads to a solution that is often equal or similar to that obtained by ℓ^{τ} minimisation for $0 < \tau \leq 1$. BP is not an algorithm per se but rather an optimisation principle [64], and as such, any convex optimisation algorithm could be used for the solution (e.g. linear programming, quadratic programming, conic programming etc.). However, such numerical optimisation methods are computationally expensive and although there are methods to accelerate the calculations (e.g. [140]) it is often difficult to adapt these to the multi-channel case.

Another algorithm that can be used for the solution of the sparse approximation problem for $0 < \tau \leq 2$ (i.e. both convex and non-convex reguralisation terms), is the *iteratively re-weighted least squares* (IRLS) [141, 142]. In this approach the criterion in (2.120) is expressed as:

$$\hat{\mathbf{c}} = \frac{1}{2}\|\mathbf{x} - \mathbf{\Phi}\mathbf{c}\|_2^2 + \lambda \mathbf{c}^H W_{\tau}\mathbf{c} \tag{2.123}$$

where $W_{\tau} = diag(|c_k|^{\tau-2})$ is a diagonal weight matrix, $\|\mathbf{c}\|_{\tau}^{\tau} = \mathbf{c}^H W_{\tau}\mathbf{c}$ is the ℓ^{τ} norm expressed in quadratic form and H denotes the Hermitian transpose.

Equation (2.123) has the following least squares solution:

$$\hat{\mathbf{c}} = (\mathbf{\Phi}^H \mathbf{\Phi} + 2\lambda W_\tau)^{-1} \mathbf{\Phi}^H \mathbf{x} \tag{2.124}$$

Since W_τ depends on $\mathbf{c}$, equation (2.124) has to be solved iteratively. Popular variants of this algorithm are the FOCUSS algorithm [66] and its multi-channel extension M-FOCUSS [143]. IRLS algorithms are known to converge quickly, in few iterations, but in the case where $\tau < 1$, the resulting optimisation problem is non-convex and therefore, it is difficult to know if the solution found is a global optimum. The solution in this case is also affected by the initialisation of the algorithm. Most importantly, the algorithm is based on matrix inversion which is a computationally expensive operation, especially when high dimensional problems are considered (i.e large $\mathbf{x}$ vector), which is usually the case for audio signals.

In the case where the ℓ^τ norm reguralisation term takes values $1 \leq \tau \leq 2$, then another family of algorithms known as *iterative shrinkage* (IS) (or *iterative thresholding*) can be used [144, 67]. In this case the resulting optimisation problem is convex and a global optimum solution is guaranteed. In this approach the criterion in (2.119) is re-written as:

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} \frac{1}{2} \|\mathbf{\Phi}(\mathbf{\Phi}^{-1}\mathbf{x} - \mathbf{c})\|_2^2 + \lambda f(\mathbf{c}) \tag{2.125}$$

When the dictionary forms an orthonormal basis (i.e. $\mathbf{\Phi}^{-1} = \mathbf{\Phi}^H$), (2.125) can be simplified to:

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} \frac{1}{2} \|\mathbf{\Phi}^H\mathbf{x} - \mathbf{c}\|_2^2 + \lambda f(\mathbf{c}) \tag{2.126}$$

$$= \arg\min_{\mathbf{c}} \sum_i \frac{1}{2} (\boldsymbol{\phi}_i^H \mathbf{x} - c_i)^2 + \lambda f(c_i) \tag{2.127}$$

where equation (2.127) essentially decouples the joint optimisation problem into a sum of I scalar optimisation problems that have to be solved separately [136]. The solution to the i^{th} optimisation problem of (2.127) is given by:

$$c_i + \lambda f'(c_i) = \boldsymbol{\phi}_i^H \mathbf{x}_{(\mathbf{i})} \tag{2.128}$$

Solving for c_i we get:

$$c_i = \psi(\boldsymbol{\phi}_i^H \mathbf{x}) \tag{2.129}$$

where ψ is known as the shrinkage function. For different values of τ (i.e. different regularisation terms) we get different shrinkage functions. When the ℓ^1 norm is used then ψ becomes:

$$\psi(u) = sgn(u) max(0, |u| - \lambda) \tag{2.130}$$

Iterative shrinkage algorithms, start with an arbitrary initialisation of the coefficient vector and then at each iteration the shrinkage/thresholding function is applied to the coefficients, effectively filtering the coefficients. Iterative shrinkage converges slower than IRLS but is computationally simpler and fast implementations have been proposed [145, 146]. Such algorithms can also be used in the multi-channel case [147].

The second strategy used for the solution of the sparse approximation problem involves greedy algorithms which instead of optimising a sparsity inducing criterion are based on an approach

where the required coefficients are extracted form the signal at each iteration in such a way so that the approximation error is reduced to the largest extend. This dissertation deals with the implementation of such an algorithm, therefore this greedy approach is discussed in detail in the following sections.

2.3 Matching Pursuit

Matching pursuit [68] is a greedy, iterative algorithm that produces sub-optimal signal expansions in the form of equation (2.113). At each iteration the algorithm chooses an atom from an over-complete dictionary that best correlates (i.e. matches) with parts of the underlying signal. The standard MP algorithm starts with an initial approximation that is empty and a residual that equals the original signal and at each iteration it subtracts from the residual the atom that has the highest correlation with it and adds that atom to the approximation. Continuing in this fashion, MP builds-up the signal model one atom at a time. MP is greedy in a sense that at each iteration it selects the atom that will absorb most of the energy from the residual signal [148]. Matching pursuit is similar to projection pursuit strategies that are used for statistical parameter estimation [149]. A similar algorithm to MP, that was developed independently can be found in [150].

2.3.1 MP algorithm

MP can be better explained by considering a single iteration of the algorithm and expressing it in terms of linear algebra. In particular, let us treat the residual $\mathbf{r}$ and an atom $\mathbf{d}_i$ ($\mathbf{d}_i \in \mathbf{D}$) as vectors in $\mathbf{R}^2$ and also let the atom be of unit norm (i.e. $\|\mathbf{d}_i\| = 1$). At the i^{th} iteration, the algorithm tries to approximate the residual $\mathbf{r}_{i-1}$ in terms of the atom $\mathbf{d}_i$. This means that the residual $\mathbf{r}_i$ is the error of the projection of the residual of the previous iteration $\mathbf{r}_{i-1}$ onto the atom $\mathbf{d}_i$ and from the orthogonality principle that was discussed in section 2.1.8 we know that this error is minimum if it is orthogonal to the atom. Also the projection $\mathbf{p}$ will be a multiple of the atom vector $\mathbf{d}_i$. This process is depicted in figure 2.26 below.

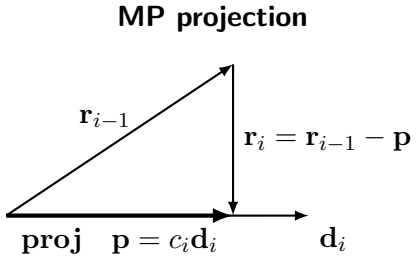


Figure 2.26: MP projection. Residual $\mathbf{r}_{i-1}$ is approximated in terms of atom $\mathbf{d}_i$

From figure 2.26 and by substituting for the projection $\mathbf{p}$ we get the projection error:

$$\boxed{\mathbf{r}_i = \mathbf{r}_{i-1} - c_i\mathbf{d}_i} \tag{2.131}$$

From the orthogonality principle we know that the projection error will be minimum if it is orthogonal to the atom vector, which means that the dot product between the projection error and the atom must be zero:

$$\mathbf{d}_i^T(\mathbf{r}_{i-1} - c_i\mathbf{d}_i) = 0 \Rightarrow c_i\mathbf{d}_i^T\mathbf{d}_i = \mathbf{d}_i^T\mathbf{r}_{i-1} \tag{2.132}$$

In equation (2.132) the dot product of the atom with itself in the left hand side produces a scalar, but for unit norm vectors this dot product reduces to unity so equation (2.132) simplifies to:

$$\boxed{c_i = \mathbf{d}_i^T\mathbf{r}_{i-1} = \langle \mathbf{d}_i, \mathbf{r}_{i-1} \rangle} \tag{2.133}$$

where c_i is the multiple of the projection $\mathbf{p}$. Now by using Pythagoras theorem as expressed in equation (2.109) and applying it to figure 2.26 we can get the following:

$$\|\mathbf{r}_{i-1}\|^2 = \|\mathbf{r}_i\|^2 + \|c_i\mathbf{d}_i\|^2 \Rightarrow \|\mathbf{r}_i\|^2 = \|\mathbf{r}_{i-1}\|^2 - \|c_i\mathbf{d}_i\|^2 \tag{2.134}$$

Again, since the atom vector is of unit norm the term $\|c_i\mathbf{d}_i\|^2$ simplifies to $|c_i|^2$ and equation (2.134) becomes:

$$\boxed{\|\mathbf{r}_i\|^2 = \|\mathbf{r}_{i-1}\|^2 - |c_i|^2} \tag{2.135}$$

It should be noted that the square of the ℓ^2 norm is the energy and as such equation (2.135) implies that in order to minimise the energy of the residual $\mathbf{r}_i$ the term $|c_i|^2$ must be maximised. The maximum c_i corresponds to the atom $\mathbf{d}_i$ that has the maximum absolute correlation with the residual. That is:

$$\boxed{\mathbf{d}_i = \underset{\mathbf{d}_i \in \mathbf{D}}{\arg\max} |\langle \mathbf{r}_{i-1}, \mathbf{D} \rangle|} \tag{2.136}$$

Although the treatment was done with vectors in $\mathbf{R}^2$ the same steps can be performed for vectors in $\mathbf{R}^n$. The above procedure describes a single iteration of MP as the projection of the residual onto a single atom. Such projections though occur for every atom in the dictionary. Therefore the MP algorithm comprises two main steps. The first step is to find the correlations of the residual with all the atoms in the dictionary (i.e. compute all projections of the residual into the atoms) and the second step is to select the correlation coefficient with the maximum absolute value (i.e. select the best optimal solution). By selecting the atom with the maximum absolute correlation coefficient the energy of the residual is monotonically decreased (with an exponential decay) at each iteration step. The algorithm continues until the energy of the residual has reached a certain threshold or until another stopping criterion has been met (e.g. the desired number of iterations has been reached). After I iterations the algorithm produces the following signal decomposition:

$$\boxed{\mathbf{x} = \sum_{i=0}^{I-1} c_i\mathbf{d}_i + \mathbf{r}_I} \tag{2.137}$$

and the energy conservation equation is satisfied:

$$\boxed{\|\mathbf{x}\|^2 = \sum_{i=0}^{I-1} \|c_i\mathbf{d}_i\|^2 + \|\mathbf{r}_I\|^2} \tag{2.138}$$

Energy conservation is satisfied despite the fact that MP is a highly non-linear procedure and the algorithm converges when $I \to \infty$, that is $\lim_{i\to\infty} \|\mathbf{r}_i\| = 0$. The proof of converge for infinite-dimensional vectors can be found in [68]. Proving the convergence of MP is mathematically involved and replicating it is beyond the scope of this thesis although the proof is simplified a lot when considering finite-dimensional vectors in $\mathbf{R}^N$. An informal proof of convergence for signals in $\mathbf{R}^n$ can be found in [112]. It will just be mentioned that the necessary condition for converge is having an over-complete dictionary that spans the space in $\mathbf{R}^n$.

Denoting an over-complete dictionary as $\mathbf{D}$, the atoms as $\mathbf{d}$, the input signal as $\mathbf{x}$, the approximation as $\mathbf{y}$ and the residual as $\mathbf{r}$, then a possible MP algorithm could be performed as follows:

Algorithm 3 Matching Pursuit

1: **Create/choose** an over-complete dictionary $\mathbf{D}$
2: **Initialise** $i = 1$, $\mathbf{y}_{i-1} = \mathbf{0}$, $\mathbf{r}_{i-1} = \mathbf{x}$
3: **Repeat**
4: Compute the inner products $\langle \mathbf{r}_{i-1}, \mathbf{d} \rangle$, $\forall \mathbf{d} \in \mathbf{D}$
5: Select best atom $\mathbf{d}_i = \arg\max_{\mathbf{d}\in\mathbf{D}} |\langle \mathbf{r}_{i-1}, \mathbf{d} \rangle|$
6: Get expansion coefficient $c_i = \langle \mathbf{r}_{i-1}, \mathbf{d}_i \rangle$
7: Update residual $\mathbf{r}_i = \mathbf{r}_{i-1} - c_i \mathbf{d}_i$
8: Update approximation $\mathbf{y}_i = \mathbf{y}_{i-1} + c_i \mathbf{d}_i$
9: Increase iteration number $i \coloneqq i + 1$
10: **Until** stopping condition

Matching pursuit it a widely used algorithm and its theoretical aspects have been studied extensively such as for example in [68], [151] and [137]. A list of software implementations is given in chapter 4.

2.3.2 MP dictionaries

MP deals with over-complete dictionaries that consist of time-frequency atoms. According to [68] such atoms can be generated by scaling, translation and modulation of a single window function $w(t)$. The window function is usually real, continuously differentiable with a non-zero integral and of unit norm. Such a family of atoms can be defined as:

$$\varphi_\gamma(t) = \frac{1}{\sqrt{s}} w(\frac{t-u}{s}) e^{i\omega t} \tag{2.139}$$

where s is the scale, u the location, and ω the modulating frequency and these parameters form the parameter index (i.e. a vector) $\gamma = (u, s, \omega)$ where $\gamma \in \Gamma$ with Γ being a set of parameters indices. The factor $\frac{1}{\sqrt{s}}$ is used to produce a unit norm atom. If $w(t)$ is even then the atom $\varphi_\gamma(t)$ is centred at u, its energy is concentrated around u and its time bandwidth is proportional to s and its Fourier transform is centred at ω with the energy concentrated around ω and its frequency bandwidth is proportional to $\frac{1}{s}$ [68]. Note that although the modulator in that case is the complex exponential, equation (2.139) is generic and we are free to choose any kind of window and any modulator. A set of such time frequency atoms form a redundant dictionary $\mathcal{D} = \{\varphi_\gamma(t)\}_{\gamma\in\Gamma}$.

Regarding the parameter index γ it doesn't have to comprise only three parameters; the number of parameters is dictated by the type of the atoms. For example if we choose $w(t)$ to be a Gaussian window and instead of the complex exponential we modulate with a real sinusoid then we get the well known family of real Gabor atoms:

$$\varphi_\gamma(t) = K_\gamma w(t) \cos(\omega(t-u) + \phi) \tag{2.140}$$

where K_γ is a normalisation constant and $w(t)$ is a Gaussian window:

$$w(t) = 2^{\frac{1}{4}} e^{-\pi(t-u)^2} \tag{2.141}$$

In this case we have to account for phase ϕ thus the parameter index becomes $\gamma = (u, s, \omega, \phi)$. Of course in practical software implementations, discrete versions of atoms are required and these can be either real or complex, although it should be noted that real and complex atoms generally lead to different decompositions. Section 3.6 describes in more detail several families of discrete-time real atoms.

2.3.3 MP realisation

For infinite dimensional spaces the dictionary $\mathcal{D} = \{\varphi_\gamma(t)\}_{\gamma\in\Gamma}$ is overly redundant and may have an infinite number of atoms. For finite dimensional spaces in practical implementations we are interested in a subset of the atoms in $\mathcal{D}$ like the set $\mathcal{D}_a = \{\varphi_\gamma(t)\}_{\gamma\in\Gamma_a}$ which is a subset of $\mathcal{D}$ and Γ_a is a finite index set included in Γ [68]. When dealing with signals in $\mathbf{R}^n$ the dictionary can be considered as a matrix with its columns being the atoms $\mathbf{d}_\gamma$ with $\gamma \in \Gamma_a$ so that $\mathcal{D}_a = \mathbf{D} \in \mathbf{R}^{N\times M}$. A typical MP implementation will start by computing all the inner products between the initial residual $\mathbf{r}_0$ (which is initialised to the original input signal $\mathbf{x}$) and the atoms $\mathbf{d}_\gamma$. Assume that the i^{th} residual has been computed, then MP will try to find an atom $\mathbf{d}_{\hat{\gamma}_i}$ so that [68]:

$$|\langle \mathbf{r}_i, \mathbf{d}_{\hat{\gamma}_i}\rangle| = \max |\langle \mathbf{r}_i, \mathbf{D}\rangle| \tag{2.142}$$

At this stage it is possible to find a better estimate of $\mathbf{d}_{\hat{\gamma}_i}$ by using a Newton method to search for a parameter index γ_i in the neighbourhood of $\hat{\gamma}_i \in \Gamma$ where the inner product of the residual with that new atom reaches a local maximum with $|\langle \mathbf{r}_i, \mathbf{d}_{\gamma_i}\rangle| \geq |\langle \mathbf{r}_i, \mathbf{d}_{\hat{\gamma}_i}\rangle|$ [68]. This double search approach can be used to provide better estimates of the selected atoms thus leading to better approximations. Having obtained $\mathbf{d}_{\gamma_i}$ (which will be denoted as $\mathbf{d}_i$ for short) then the inner products of the residual with the atoms in $\mathbf{D}$ can be calculated using an update formula derived from equation (2.131). In particular, starting from equation (2.131) and by applying the rules of transpose matrices and matrix multiplication we get the following:

$$\begin{aligned}
\mathbf{r}_i &= \mathbf{r}_{i-1} - c_i \mathbf{d}_i \Rightarrow \\
\mathbf{r}_i^T &= \mathbf{r}_{i-1}^T - c_i \mathbf{d}_i^T \Rightarrow \\
\mathbf{r}_i^T \mathbf{D} &= \mathbf{r}_{i-1}^T \mathbf{D} - c_i \mathbf{d}_i^T \mathbf{D} \Rightarrow \\
\langle \mathbf{r}_i, \mathbf{D}\rangle &= \langle \mathbf{r}_{i-1}, \mathbf{D}\rangle - \langle \mathbf{r}_{i-1}, \mathbf{d}_i\rangle \langle \mathbf{d}_i, \mathbf{D}\rangle
\end{aligned} \tag{2.143}$$

In equation (2.143) the terms $\langle \mathbf{r}_{i-1}, \mathbf{D}\rangle$ and $\langle \mathbf{r}_{i-1}, \mathbf{d}_i\rangle = c_i$ were computed and stored in the previous iteration so the only new computation required is $\langle \mathbf{d}_i, \mathbf{D}\rangle$ which can be pre-computed

and stored. In practice many dictionaries are designed in such a way so that the inner products $\langle \mathbf{d}_i, \mathbf{D} \rangle$ for $i = 1...M$ can be calculated using very few operations thus leading to increased computational speeds. Note that this procedure is only feasible if the dictionary and the input signals are small in size. For large number of atoms and lengthy input sequences pre-computing all inner products in $\langle \mathbf{d}_i, \mathbf{D} \rangle$ would not be possible due to insufficient memory.

2.3.4 MP visualisations

The flexibility of MP extends into the ways that a certain decomposition can be visualised. The most common MP visualisation is produced by adding together the Wigner distribution of the extracted atoms leading to the MP time-frequency energy distribution. This approach is explained in detail in [68] and independently in [150]. In particular the MP time-frequency distribution is given by [68]:

$$\mathcal{E}_x(t,\omega) \stackrel{\text{def}}{=} \sum_{i=0}^{\infty} |c_i|^2 W d_{\gamma_i}(t,\omega) \tag{2.144}$$

where c_i is given by equation (2.133) and $W d_{\gamma_i}(t,\omega)$ is the Wigner distribution of atoms d_γ. The Wigner distribution of two functions $f(t)$ and $h(t)$ is given by [68]:

$$W[f,h](t,\omega) \stackrel{\text{def}}{=} \frac{1}{2\pi} \int_{-\infty}^{\infty} f\left(t+\frac{\tau}{2}\right) h^*\left(t-\frac{\tau}{2}\right) e^{-i\omega t} d\tau \tag{2.145}$$

and the Wigner distribution of $f(t)$ is given by $W_f(t,\omega) = W[f,f](t,\omega)$. The derivation of (2.144) from (2.145) can be found in [68] and [150]. Figure 2.27 depicts the MP energy distribution of the quake signal. The signal was decomposed into 256 atoms, using a multi scale Gabor dictionary. Compared to the STFT representation of figure 2.14 it can be clearly seen that the MP representation is sparser. Also the extracted atoms seem to have captured well the most prominent features of the signal. The colour represents atom energy with darker colours denoting higher energy atoms.

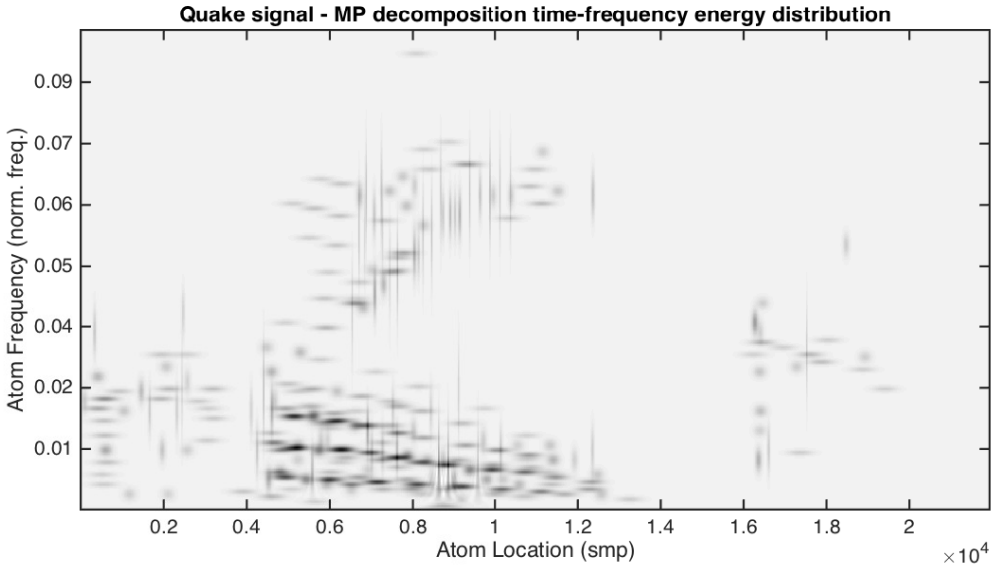


Figure 2.27: MP time-frequency energy distribution of quake signal. This distribution is produced by adding together all the Wigner distributions of the extracted atoms. Colour represents the energy of the expansion coefficients with darker colours denoting higher energy atoms.

One problem with the energy distribution in (2.144) is that in practical implementations it can be very slow to produce. A faster way to represent the same information is by using a patch plot as depicted in figure 2.28. In this plot each rectangle is a time-frequency tile with a time and frequency bandwidth calculated by equations (2.57) and (2.58). The product of the time and frequency bandwidths of each atom obeys the uncertainty principle. This representation clearly demonstrates the arbitrary segmentation of the time-frequency plane that is produced by an atomic decomposition. in this particular example a Gabor dictionary with five scales was used and all kinds of atoms can be observed. Although in this figure the relationships between the atoms are not clear the generic pattern of atom distribution is distinguishable and comparable to the one produced by the Wigner distribution and the STFT.

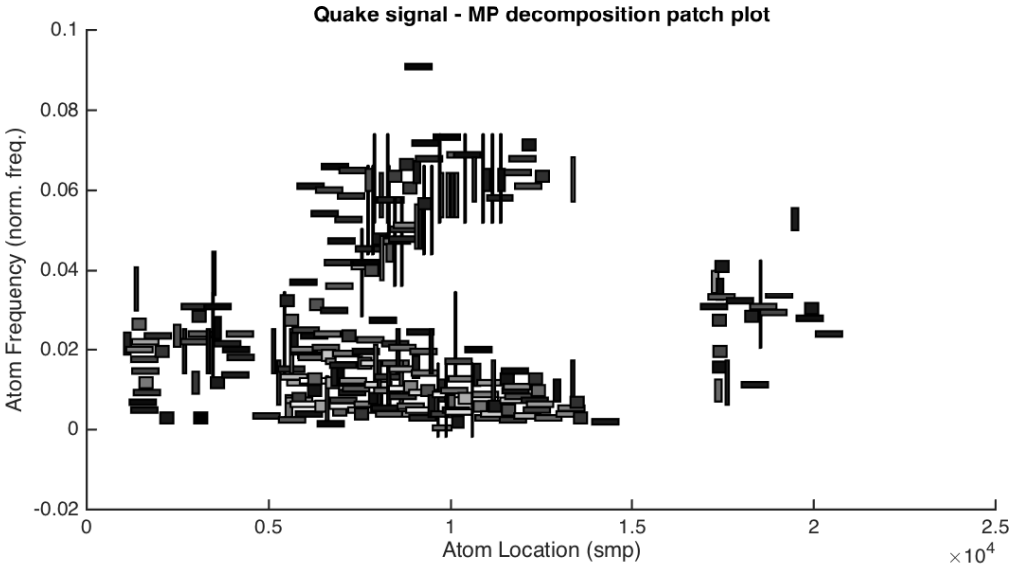


Figure 2.28: Patch plot of MP decomposition. In this view each tile represents an atom with certain time and frequency bandwidths. Colour represents the magnitude of the expansion coefficients with darker colours denoting higher energy atoms. This is a demonstration of an arbitrary segmentation of the time-frequency plane. The product of the time and frequency bandwidths obey the uncertainty principle.

Yet another alternative representation is the scatter plot depicted in figure 2.29. In this time-frequency plot the atoms are represented by circles the size of which represents the scale of the atoms, with larger circles denoting large scaled atoms and small circles small scaled atoms. Colour represents energy in the same way as in the previous plots. Although this plot contains the same information as the patch plot and the energy distribution plot, it is a bit clearer and certain patterns among atoms are more visible.

Of course other representations are possible, such as the typical time-domain representation in figure 2.30. In this plot the dots represent the expansion coefficients, which are sorted by location and normalised. The dashed lines connect the coefficients but in reality this doesn't happen; the lines are used here just to improve the graphic. Observing this time domain plot it can be seen that it follows a very similar pattern as the one seen in the time domain waveform of the signal in figure 2.1.

MP allows for novel visualisations of the decomposed data. The plots presented here are the most basic but there are many other possible visualisations. Recall that the atoms of the decomposition belong to a parameter space, that is each atom has its own parameter vector and any parameter

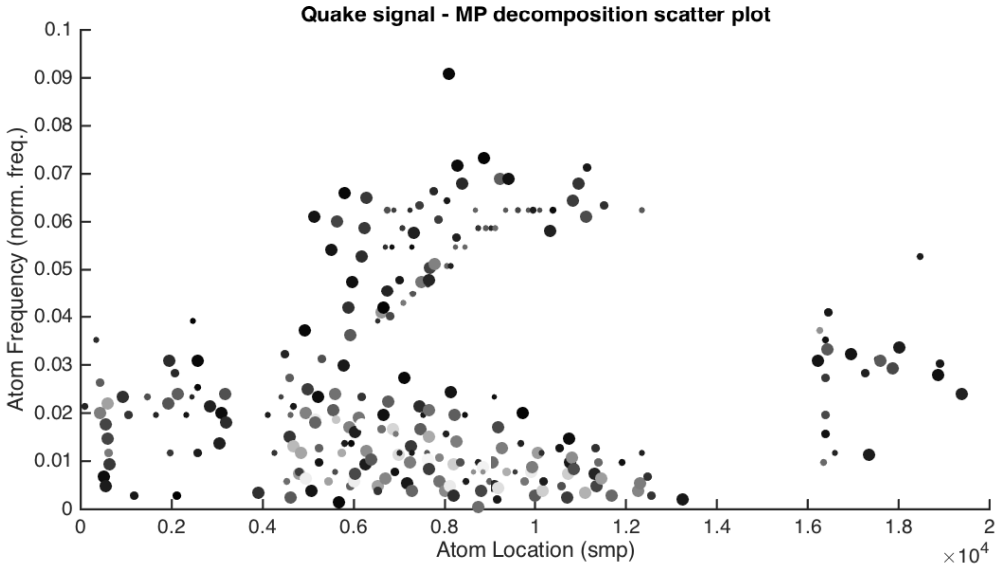


Figure 2.29: Scatter plot representation of MP decomposition. This view is again int the time-frequency plane with each circle representing an atom and the size of each circle represents the scale of the atom, with large circles denoting large scales and small circles small scales. Colour represents the magnitude of the expansion coefficients with darker colours denoting higher energy atoms.

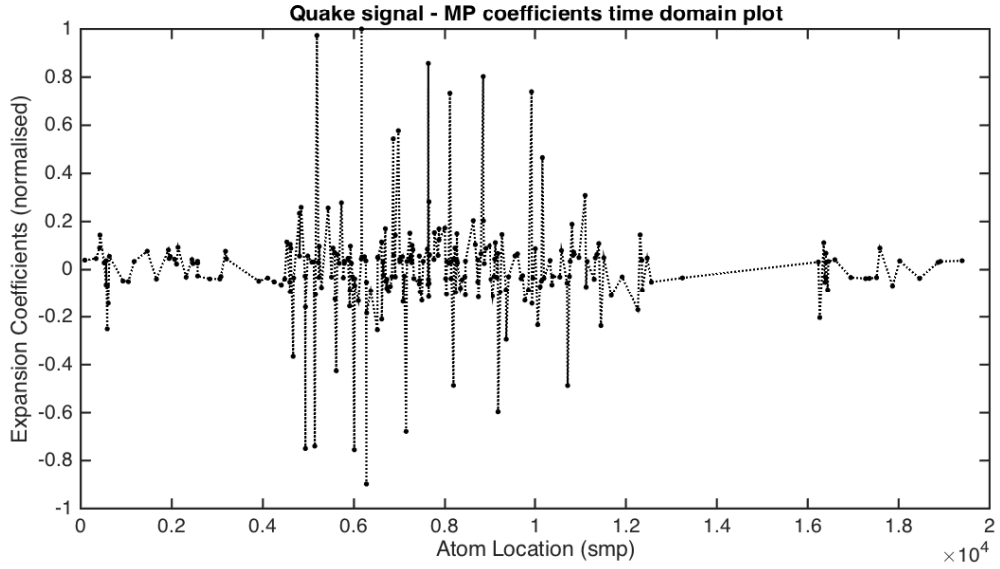


Figure 2.30: Time domain plot of MP decomposition of quake signal. This plot show the expansion coefficients, sorted in time, and normalised. The resulting pattern matches the general pattern of the quake signal waveform in figure 2.1

.

can be essentially plotted. Depending on the parameter space, any set of parameters can be plotted together like for example in a 3-D scatter plot of location (i.e. time), frequency and scale with colour representing the magnitude of each coefficient. Interesting observations also occur when certain atom parameters are plotted in a histogram. An easy to use plotting utility is provided with the software implementation of MP that is part of this work.

2.3.5 MP performance metrics

When dealing with MP it is important to know if a certain decomposition has 'behaved' as expected, therefore certain performance metrics are required. For the following let us assume typical notation and use $\mathbf{x} \equiv x[n]$ to denote the input signal, $\mathbf{y} \equiv y[n]$ the approximation and $\mathbf{r} \equiv r[n]$ the residual and assume an 'I-term' decomposition meaning that the number of iterations is set to I.

MP is a greedy algorithm, in an energy sense, where at each iteration the 'best' atom is the one that absorbs the most energy from the residual, therefore the first metric is based on the reduction of energy at each iteration. In particular the energy of a discrete-time sequence is given by equation 2.21 but for completeness it will be redefined here in terms of a residual $r_i[n]$ of length N:

$$E_{r_i} \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} |r_i[n]|^2 \tag{2.146}$$

At every iteration the residual energy decreases in an exponential fashion and by recording these values an energy decay curve (EDC) can be obtained. For an 'I-term' approximation, the EDC is a vector in $\mathbf{R}^I$ and will be defined by:

$$EDC \stackrel{\text{def}}{=} \mathbf{EDC} \coloneqq (E_{r_1}, \ldots, E_{r_I}) \tag{2.147}$$

A typical EDC is depicted in sub-plot $a)$ of figure 2.31. This EDC corresponds to the decomposition of the 'quake' signal. Observing this sub-plot it can be seen that the residual energy decreases rapidly in the first iterations but the asymptotic decay rate is slow. This behaviour is typical of an MP decomposition and has been studied in detail in [137] where the authors show that matching pursuits are chaotic ergodic maps. The decay rate of an EDC depends on the selected dictionary and the authors in [137] explain ways to estimate it and derive the notion of coherence of a signal with respect to a given dictionary. This issue will not be pursued any further but the important observation is that the atoms selected during the initial steps of the decomposition correspond to coherent structures within the signal whereas the atoms in the tail of the decomposition correspond to the 'noise' element of the signal.

Another metric which is directly related to the EDC is the signal to residual ratio (SRR) which for a single iteration is defined by:

$$SRR_i \stackrel{\text{def}}{=} 10log_{10}\left(\frac{E_x}{E_{r_i}}\right) \tag{2.148}$$

where E_x and E_{r_i} are the energies (or the squared norms since all sequences are real) of the input signal and the residual at the i^{th} iteration respectively. The output is in dB units. In a similar fashion to the EDC the SRR curve can be obtained by recording the SRR values of each iteration and will be defined by:

$$SRR \stackrel{\text{def}}{=} \mathbf{SRR} \coloneqq (SRR_1, \ldots, SRR_I) \tag{2.149}$$

The SRR for the 'quake' example is depicted in sub-plot $b)$ of figure 2.31. This metric is in units of dB therefore it is much easier to observe the SRR's increasing rate. Generally speaking the

greater the SRR at the end of the decomposition the better the quality of the approximation signal will be.

Another metric related to the residual norm is the ℓ^2 error norm which expresses the ℓ^2 norm of the residual relative to the ℓ^2 norm of the input signal as a percentage. This can be defined as:

$$L2err_i \stackrel{\text{def}}{=} 100\left(\frac{\|\mathbf{r}_i\|}{\|\mathbf{x}\|}\right) \tag{2.150}$$

where $\|\cdot\|$ denotes the ℓ^2 norm and similarly to the previous metrics the L2err curve is obtained by:

$$L2err \stackrel{\text{def}}{=} \mathbf{L2err} \coloneqq (L2err_1, \ldots, L2err_I) \tag{2.151}$$

The L2err curve is depicted in sub-plot $c)$ of figure 2.31 where it can be seen that it is decreasing at each iteration step in a similar fashion to the EDC.

A final metric is the coefficient cumulated quality (CCQ) which can be found in MATLAB®'s MP implementation in [152]. CCQ measures the cumulated energy of the selected expansion coefficients relative to the energy of the input signal and essentially measures the 'quality', in an energy sense, of the extracted coefficients at each iteration. This is defined as:

$$\begin{aligned} CCQ_i &\stackrel{\text{def}}{=} \frac{\|\mathbf{C}_i\|^2}{\|\mathbf{x}\|^2} \\ \mathbf{C}_i &\coloneqq (c_1, \ldots, c_i), \quad 1 \leq i \leq I \end{aligned} \tag{2.152}$$

where $\mathbf{C}_i$ is the vector of extracted coefficients up to the i^{th} iteration step. The CCQ curve is obtained by recording the metric at each iteration:

$$CCQ \stackrel{\text{def}}{=} \mathbf{CCQ} \coloneqq (CCQ_1, \ldots, CCQ_I) \tag{2.153}$$

The produced CCQ curve for the 'quake' example is depicted in sub-plot $d)$ of figure 2.31 where it can be seen that the quality increases at each iteration and converges towards one as the decomposition progresses.

All aforementioned metrics can be used to quantify the 'quality' and 'behaviour' of a given MP decomposition but the most commonly used is the SRR because of its clear interpretation; that is, it is much easier to compare the SRR curves of two decompositions rather than their EDC. The EDC becomes difficult to read when the energy approaches zero whereas in the SRR curves even differences below 1dB are visible. As a rule of thumb a good quality and well behaved decomposition can be thought of as one where the EDC decreases as rapidly as possible in the fewest possible iterations and with a negligible amount of energy left in the residual, which translates to a rapidly increasing SRR curve with a high final SRR value. Such metrics will imply that the dictionary used for the decomposition is coherent with respect to the inner structures of the input signal and the resulting decomposition is expected to be compact [137]. Finally, any or all of these metrics, along with the maximum number of iterations allowed, can be used as stopping conditions for a decomposition task (i.e. in step 10 of algorithm 3).

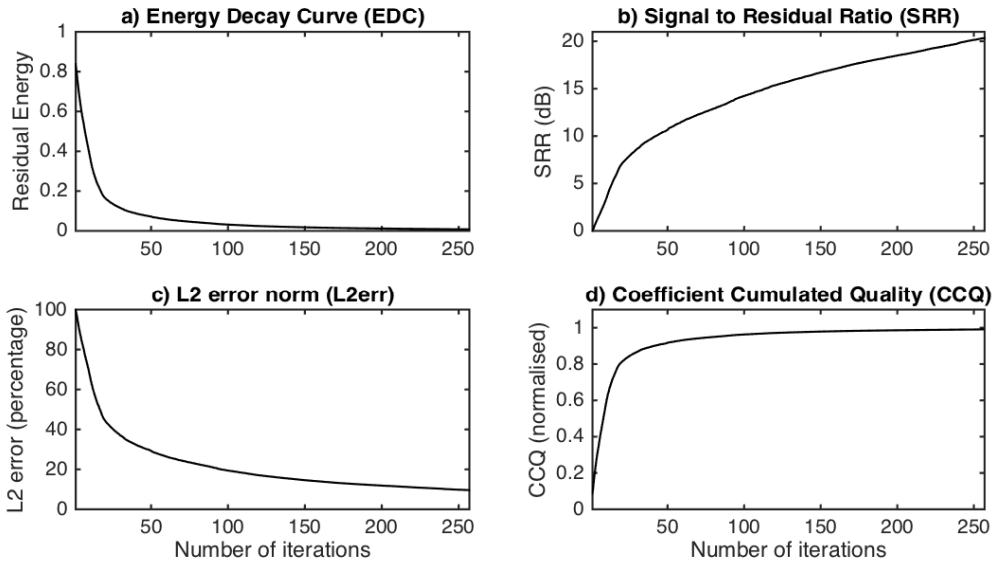


Figure 2.31: MP decomposition metrics for 'quake' example. Sub-plot $a)$ depicts the residual energy decay curve (EDC). The residual error decreases exponentially where in the first few iterations the decrease is rapid but the asymptotic decay rate is slow. This is typical behaviour of an MP decomposition and the atoms of the first few iterations correspond to coherent structures within the signal whereas atoms belonging to the tail of the curve correspond to noise. Sub-plot $b)$ depicts the signal to residual ratio (SRR) in dB units, which is the ratio between the energy of the residual and the energy of the input signal at each iteration. Sub-plot $c)$ depicts the L2 error norm which expresses the norm of the residual relative to the norm of the input signal as a percentage. Sub-plot $d)$ depicts the coefficient cumulated quality (CCQ) which describes the 'quality' of the selected coefficients up to the i^{th} iteration step.

2.3.6 MP drawbacks

As with all algorithms and techniques presented so far, MP has a few drawbacks. The most important one is its inability to super-resolve under certain conditions. In particular, when the dictionary is not orthogonal, the algorithm might choose the 'wrong' atoms in the first few iterations and subsequent iterations will try to rectify the errors introduced by the first selected atoms [64]. As a consequence the decomposition of a signal with simple structures might imply a signal with highly complex structures. For example given a signal with two closely spaced frequency components, in many cases the first atom that will be selected by MP will be an average of the two initial components. In order to rectify the error introduced by that first atom, subsequent iterations will extract atoms that were not present in the original signal which means that the initial simple structure of the signal is missed completely. A similar behaviour occurs when a signal consists of two atoms closely spaced in time. In this case the algorithm might first select a single atom that matches the characteristics of both underlying atoms thus again leading to a 'wrong' first choice, the error of which must be compensated by subsequent atom extractions. Such 'wrong' choices mainly occur because the algorithm itself is myopic [64] and also because of its greedy approach to energy reduction; the algorithm will try to extract the atom that will reduce the energy of the residual the most, regardless of the underlying structure of the signal. Although this is certainly a potential problem it can be argued that this behaviour is not unreasonable depending on the context. For example given a compression task with the aim of signal transmission the inability to super-resolve the underlying signal components is irrelevant. The problem is most relevant though in analysis and feature detection and extraction tasks where the structure of the signal is important. One solution to this problem is proposed in [148] using a

variant of MP called high resolution pursuit (HRP). Another possible solution using the algorithm presented in chapter 3 will be briefly discussed.

Another problem has to do with the convergence of the algorithm. Given certain dictionaries and signals, MP fails to produce a sparse representation and in some cases it fails to converge. Some of these examples can be found in [148]. A solution to such problems has been proposed by yet another variant of MP called orthogonal matching pursuit (OMP) [69] which convergences in a finite number of iterations.

Another potential problem stems from the required discretisation of the continuous space of the dictionary parameters which introduces a statistical bias that becomes relevant when multiple MP decompositions are analysed, like for example in the context of EEG data analysis [153]. The authors in [153] propose a solution to this problem by creating stochastic dictionaries where the parameters of the atoms in each dictionary are randomised before each decomposition. Although this problem does not directly affect the work presented in this thesis, in chapter 3 a solution is presented which is actually a by-product of the algorithm presented in chapter 3, thus eliminating the need for the approach proposed in [153].

A final drawback has to do with the computational speed of MP. Although MP is faster than many other atomic decomposition techniques (faster than BP for example) it is still very slow compared with techniques based on the STFT and especially when considering large scaled problems (i.e. with long duration discrete-time signals). In fact, most literature on MP seems to provide solutions for small sized problems only. A typical 'textbook' MP implementation (where by 'textbook' implementation is meant a direct implementation of algorithm 3) can only deal with signal sizes up to a few thousand samples, like for example MATLAB®'s implementation [152][4](henceforth BMP for basic matching pursuit). The only MP implementation known to the author, that is capable of dealing with real life signals in approximate real time is MPTK [154][5] and even this implementation becomes excessively slow when the dictionaries used are over-sampled or when multiple type dictionaries are used. There are several possible algorithmic optimisations that can improve the computational speed but usually such optimisations lead to increased memory sizes; that is, improvements in speed lead to increased memory usage and vice versa which is a typical trade-off found in many algorithms. It should be noted though that given certain well structured and small dictionaries and by stream-processing the incoming data it is possible to implement a real-time MP algorithm. Some ideas on how to achieve this are briefly described in chapter 3. Also, in an off-line processing context it could be possible to compute an MP decomposition really fast by using specialised hardware such as FPGAs.

Despite the aforementioned potential problems, MP is a generic algorithm with great flexibility and this flexibility outweighs any drawbacks especially when it comes to audio signal processing.

2.3.7 MP variants

The flexibility of MP has led to the development of many algorithms that either improve certain aspects of the standard matching pursuit algorithm or take advantage of the generality of the algorithm to create hybrid solutions to certain problem domains.

[4] http://uk.mathworks.com/help/wavelet/ref/wmpalg.html
[5] http://mptk.irisa.fr/

Probably the first, most well known and most studied MP variant is the orthogonal matching pursuit (OMP) proposed in [69] and independently in [151] and [137]. The main difference with MP is that given a N-dimensional vector, OMP is guaranteed to converge in N or less iterations. In MP, the atom selected at each iteration is orthogonal to the residual at that iteration but generally is not orthogonal to the previous selected atoms and as such when the projection is subtracted from the residual then new components are introduced in the direction of all the previously selected atoms. A consequence is slow asymptotic convergence. To avoid this problem OMP, at each iteration, orthogonalises all previously selected atoms using a Gram-Schmidt orthogonalisation procedure [137]. The main drawback of OMP is the increase in computational complexity since the orthogonalisation procedure has to be performed at each iteration (Gram-Schmidt orthogonalisation involves computation of inverse matrices which is generally a computationally expensive process). There are several approaches that approximate OMP and lead to faster algorithms but usually such approaches take advantage of the structure of special dictionaries which in itself might be limiting to the flexibility of MP.

Another popular variation is the so called weak matching pursuit (WMP). WMP is useful when dealing with very large dictionaries and the atom selection criterion, that is, selecting the atom that maximises the absolute value of the inner product, is relaxed, with the selected atom being chosen from a subset of the original dictionary (as explained in section 2.3.3) [155]:

$$|\langle \mathbf{r}, \mathbf{d}_i \rangle| \geq \alpha \max_{\gamma} |\langle \mathbf{r}, \mathbf{d}_\gamma \rangle|, \quad \alpha \in (0, 1] \tag{2.154}$$

where α is called the relaxation factor and $\mathbf{d}_\gamma$ are the atoms in $\mathbf{D}$.

In [156] and [157] the authors propose an MP variant based on conjugate subspaces. In standard MP only a single atom is selected at each iteration. In subspace pursuit in the i^{th} iteration the algorithm searches for a $N \times R$ matrix $\mathbf{G}$ that minimises the ℓ^2 norm of the residual $\mathbf{r}_i = \mathbf{r}_{i-1} - \mathbf{G}\mathbf{c}_i$, where $\mathbf{c}$ is now an $R \times 1$ vector of expansion coefficients. The solution for $\mathbf{c}$ follows again from the orthogonality principle as is the case in MP. This process is generally computationally expensive unless $\mathbf{G}$ has some special structure like for example when the columns of $\mathbf{G}$ comprise an atom and its complex conjugate (thus leading to conjugate subspaces). A full derivation of this approach can be found in [156] and [157]. The main benefit of this approach is that given a real signal then a real decomposition using a complex dictionary becomes possible. A direct consequence of this for audio signals is that a dictionary of complex atoms allows the estimation of phase without explicitly including it in the parameter set. The authors continue the work and show how to use this approach to decompose a signal against a dictionary of damped sinusoids which are better suited to describe transient phenomena that are asymmetric.

Another variant is the high resolution pursuit (HRP) in [148] which was proposed as a solution to the super-resolution problem of MP which was discussed earlier. HRP essentially introduces a new similarity measure, different from the usual inner-product, which emphasizes local fit over global fit [148]. This method is yet another example of the flexibility of MP.

Other notable variants that demonstrate the flexibility of MP are the fast ridge pursuit [158] where MP is extended to use a dictionary of Gaussian chirps, the harmonic matching pursuit (HMP)[159] where at each iteration the residual is projected onto the subspace spanned by a

dictionary of harmonic atoms which is an extension of the standard Gabor dictionary, and the molecular matching pursuit (MMP) [160] where a signal is decomposed into tonal and transient 'molecules' , that is groups of similar type atoms, where tonal molecules can be obtained via the modified discrete cosine transform (MDCT) and transient molecules via the DWT. The HMP and MMP algorithms demonstrate the possibility of extracting more than one atoms per iteration. Of particular interest are also hybrid algorithms like for example the one presented in [123] and [112] where a sinusoidal modelling analysis/synthesis system is used where each frame is decomposed by a perceptually weighted matching pursuit algorithm. Finally a post-processing technique known as agglomerative clustering (AC) [161] can be applied to the extracted expansion coefficients in order to impose structure and create coherent groups of similar atoms that might better characterise certain aspects of a signal.

2.4 Matching Pursuit in Source Separation of Music

In the previous section the MP algorithm was discussed in detail and several variations were presented dealing with problems such as convergence in a finite number of steps (OMP), feature extraction (HRP), speed optimisations (WMP and subspace pursuits), signal decomposition into various dictionaries (Gaussian chirps, HMP with harmonic atoms, MMP with tonal and transient molecules) and post-processing techniques (AC). This section presents MP variants that are aimed at source separation problems.

2.4.1 Stereo Matching Pursuit

The first application of MP into the source separation of linearly mixed stereophonic recordings with more sources than mixtures, was presented by Gribonval in [162]. Gribonval proposed an MP variant that is able to decompose a stereo signal using a stereo dictionary, called stereo matching pursuit (SMP). Although the paper focuses on stereo signals, the algorithm and the idea of the stereo dictionary can be extended to signals with more than two channels. The following equations are taken from [162]. A linearly mixed stereo signal $x(t)$ comprising P sources, can be expressed as:

$$x(t) = \sum_{p=1}^{P} \lambda_p [\cos\Theta_p \, s_p(t - \tau_{l,p}),\ \sin\Theta_p \, s_p(t - \tau_{r,p})] + n(t) \tag{2.155}$$

where s_p are the individual sources, $0 \leq \Theta \leq pi/2$ a pan-pot parameter, $\tau \leq \tau_{max}$ a delay parameter with τ_{max} being the maximum delay between the channels and λ_p a gain parameter. By substituting the linear expansion equation (2.113) for each source s into equation (2.155) we get:

$$x(t) = \sum_{p=1}^{P} \sum_{m=1}^{M} c_{p,m} \lambda_p [\cos\Theta_p \, \varphi_m(t - \tau_{l,p}),\ \sin\Theta_p \, \varphi_m(t - \tau_{r,p})] + n(t) \tag{2.156}$$

A stereo dictionary can be denoted by:

$$\mathcal{D}_{stereo} = [\cos\theta \, \varphi(t),\ \sin\theta \, \varphi(t - \tau)] \tag{2.157}$$

with $\varphi \in \mathcal{D}$ being a mono atom, θ a pan-pot parameter and τ the delay parameter.

Stereo matching pursuit decomposes (2.156) against the stereo dictionary in (2.157) into an M-term approximation similar in form to equation (2.113):

$$x(t) = \sum_{m=1}^{M} [\cos\theta_m \, \varphi_m(t), \ \sin\theta_m \, \varphi_m(t - \tau_m)] + r^M(t) \tag{2.158}$$

Note that each stereo atom in (2.158) represents an atom of a source p in (2.156) and therefore every pair (θ_m, τ_m) represents an estimate of the pan-pot and relative delay parameters $(\Theta_p, \tau_{r,p} - \tau_{l,p})$ [162]. In order to perform source separation, the pairs (θ_m, τ_m) (which imply the atom indices) are partitioned into K clusters which can be used to obtain the estimated separated sources up to a gain and a delay:

$$\widehat{\lambda_p s_p} = \sum_{m \in K_i} c_m \varphi_m \tag{2.159}$$

where $i = 1...K$. The mixing matrix coefficients and the clustering of the expansion coefficients could be obtained manually by the histogram of the θ_m parameters as is the case in [162]. The aforementioned source separation procedure is exactly the same as the DUET algorithm [57], with the main difference being that the approach presented here is using the MP instead of the STFT as the front-end to the source separation algorithm. Since MP using a multi-scale Gabor dictionary is similar to a multiple STFT with different window sizes, the algorithm presented here can be thought of as an extension to the DUET algorithm where the window sizes are chosen in an adaptive manner. It should be noted that this algorithm attributes each extracted stereo atom to a single source only. The authors in [163] provide another variant where the extracted atom is attributed to the N closest sources leading to an algorithm similar in nature to Boffil's and Zibulevsky's algorithm in [164].

Regarding the actual SMP algorithm, it is very similar to the original MP with the main difference being that the stereo signal is projected onto the stereo subspace $\mathcal{V}_{\varphi,\tau}$ spanned by the stereo dictionary in (2.157) and the 'best' atom (in an energy sense), is given by [162]:

$$(\varphi_M, \, \tau_M) \coloneqq \underset{\varphi \in \mathcal{D}, |\tau| \leq \tau_{max}}{\arg\max} \| P_{\mathcal{V}_{\varphi,\tau}} \, \mathbf{r}^{M-1} \| \tag{2.160}$$

where $\| P_{\mathcal{V}_{\varphi,\tau}} \, \mathbf{r}^{M-1} \|$ is the projection of $\mathbf{r}^{M-1}$ onto the stereo subspace given by [162]:

$$\| P_{\mathcal{V}_{\varphi,\tau}} \, \mathbf{r}^{M-1} \|^2 = |\langle r_l^{M-1}(t), \, \varphi(t) \rangle|^2 + |\langle r_r^{M-1}(t), \, \varphi(t - \tau) \rangle|^2 \tag{2.161}$$

which essentially is the maximisation of the sum of squared dot products across both channels. The full SMP algorithm can be found in [162] with an application to the source separation of a three source linearly mixed stereo signal.

2.4.2 Demixing Pursuit

In [163] the authors present another source separation algorithm that is based on MP which they term de-mixing pursuit (DP). DP works by having a priori knowledge of the mixing matrix $\mathbf{A}$ in contrast to the SMP approach where the mixing matrix is unknown. By letting $\mathbf{A} = (\cos\boldsymbol{\theta}^T, \sin\boldsymbol{\theta}^T)$ be a $J \times P$ mixing matrix (where J are the number of channels and P the

number of sources) with $\mathbf{a}_p$ the mixing coefficients of the p^{th} column of $\mathbf{A}$ and by setting the delay parameter $\tau = 0$ and dropping the noise term $n(t)$, equation (2.156) can be written as:

$$x(t) = \sum_{p=1}^{P} \sum_{m=1}^{M} c_{p,m}\, \mathbf{a}_p\, \varphi_m(t) \tag{2.162}$$

where all other symbols have their usual meaning. The atoms $\mathbf{a}_p\, \varphi_m(t)$ constitute a 'directional' multi-channel dictionary [163] and can be used to get an M-term MP decomposition where the inner-products are computed by [163]:

$$\langle \mathbf{r}^M,\, \mathbf{a}_p \varphi_m(t) \rangle = \mathbf{a}_p^T\, \mathbf{r}^M\, \boldsymbol{\varphi}_m^T \tag{2.163}$$

and a source s_p can be reconstructed by [163]:

$$\hat{s}_p(t) = \sum_{m} c_{p,m} \varphi_m(t) \tag{2.164}$$

where the expansion coefficients $c_{p,m}$ are proportional to $\mathbf{a}_p$.

The theoretical properties of DP have been studied in [165]. DP is also the main algorithm used in the MPTK software package and is the baseline, MP based, source separation algorithm that we compare against in chapter 6.

2.4.3 Dual Matching Pursuit

Another MP-based source separation algorithm is presented in [166] which is similar in operation with the SMP approach presented earlier. In this algorithm though instead of minimising the residual across both channels simultaneously, as is done in SMP (equation (2.161)), each channel of the mixture is decomposed separately, therefore the name dual matching pursuit (DMP). The main assumption is that the first extracted atoms between the two decompositions will be highly correlated, since these are the atoms that contain most of the energy of the signal and will most probably correspond to important signal structures. An atom appearing in both channel decompositions can be used to extract information of the pan-pot parameters θ (in equation (2.158)) using the following:

$$\theta_m = \tan^{-1}\left(\frac{c_{l,m}}{c_{r,m}}\right) \tag{2.165}$$

where $c_{l,m}$ and $c_{r,m}$ are the expansion coefficients of the correlated left and right atoms obtained by the left and right decompositions respectively. Having obtained all possible θ_m values the mixing matrix coefficients and the estimated sources can be obtained in exactly the same way as in the SMP approach.

The authors go a bit further and improve the correlation of the extracted atoms between channels by using a trained dictionary. By using an algorithm presented in [167] they create a dictionary by using sources that are similar in nature with the ones founds in the mixture, as training sets. The amount of atoms in the created dictionary is reduced by removing all atoms that are highly correlated. This reduced dictionary is then used with the dual matching pursuit to decompose the stereo signal. In the example provided in [166] this process seems to produce a much cleaner histogram of the θ values which lead to a better mixing matrix estimation and also improved the evaluation scores of the separated sources.

2.4.4 Matching Pursuit With Source Specific Dictionaries

One more interesting MP variant which is aimed at source separation of single-channel mixtures is presented in [168], [169] and the same algorithm applied to the sparse representation of musical signals in [170]. There seems to be an inconsistency with the naming of the algorithm where the authors term the algorithm as matching pursuit with content adaptive dictionaries (MP-CAD) in [168] and matching pursuit with source specific dictionaries (MP-SSD) in [170] and [169]. The latter naming should be used since it better describes the internal mechanics of the algorithm. Naming inconsistencies aside, the proposed algorithm is very interesting and demonstrates the ability of MP to be used in source separation of single-channel mixtures, although there is no reason why it couldn't or shouldn't be used for stereo mixtures as well.

The main idea is the creation of source-specific dictionaries, that is trained dictionaries, which are able to efficiently capture the internal structures of specific classes of signals. The approach taken to create those specific-dictionaries is not based on traditional dictionary learning techniques such as for example K-SVD [171], but on MP exclusively. In particular there are three main steps for creating source-specific dictionaries. Firstly a set of musical components of a particular type of signal (e.g. musical notes of flute, clarinet, piano etc.) is decomposed against standard source-independent dictionaries (e.g. Gabor dictionary). Secondly the extracted atoms are prioritised based on their ability to approximate the desired signal type. Thirdly the prioritised atoms are used to synthesize new atoms that eventually form compact source-specific dictionaries. Source separation can then take place by decomposing a given single-channel mixture against several of such source-specific dictionaries. The idea is that since the dictionaries are created in such a way as to only capture structures of particular types of signals, a decomposition of a mixture using such a dictionary will only yield an approximation of the type of signal that the particular dictionary represents.

The results presented in [168] and [169] are positive and although the current work presented in this thesis does not deal with single-channel mixtures, MP-SSD and especially its extension to stereo mixtures should be considered for future work.

2.5 Summary

This chapter introduced all technologies relevant to this thesis along with important notation and nomenclature. The most common traditional signal representation techniques in the time, frequency, time-frequency and time-scale domains were presented. The pivotal idea of representing a signal as linear combinations of elementary signals, called atoms, was described and the main limitation of representing a signal against a single basis was emphasized and demonstrated with example representations of a speech signal. That led to the discussion of atomic decompositions that produce sparse adaptive representations and some of the most well known techniques were briefly discussed. The matching pursuit algorithm was introduced and various aspects of it were discussed in detail. Finally variants of the matching pursuit algorithm that deal with source separation problems were presented. The chapter also introduced fundamental concepts in signals & systems theory and linear algebra that are required for the implementation of all the algorithms that are described in chapters 3 and 4. The convolution sum was derived from first principles and its connection to cross-correlation was outlined along with a way to perform fast convolution using the Fourier transform.

Chapter 3

Guided Matching Pursuit (GMP)

In this chapter a new variant of matching pursuit, called guided matching pursuit (GMP) is presented. In section 3.1 the main motivations behind this new technique are discussed before moving onto the description of the algorithm in section 3.2. Finally sections 3.3 to 3.7 describe in detail various properties and implementation aspects of GMP.

3.1 Motivation

Section 2.3 presented the advantages of MP against other sparse decomposition techniques, most importantly its relative simplicity and flexibility which can be backed up by the wealth of MP variations that exist. MP as an algorithm has been used for the solution of several problems such as data compression, audio coding, feature extraction, analysis-transformation-synthesis of audio signals and source separation. The suitability of MP for the solution of source separation problems was also made clear in section 2.4 where several MP variants focused on that problem domain, were discussed.

In terms of source separation and especially the SCA family of techniques presented in section 1.4.3, MP is a very attractive candidate to be used as the front end to a SCA based algorithm since it produces sparse adaptive representations, albeit sub-optimal. Regardless of the evidence that sparse representations can lead to better quality source separation, many state of the art algorithms still use the ubiquitous STFT as the front end. Probably the main reason for choosing this representation has to do with the fast computational speed of the STFT compared to all state of the art sparse decompositions which allows it to be used in real-time systems. Speed is a requirement and a main driving factor behind the design of many algorithms. Unfortunately the speed requirement is in conflict with the sparsity requirement of sparse decomposition techniques. In section 2.2.1 it was explained that the problem of approximating a signal in terms of atoms selected from an over-complete dictionary has infinite solutions and finding the 'optimal' solution is an intractable problem (in terms of computational complexity theory finding the optimal solution is considered an NP-hard problem [137]). Algorithms that deal with such problems are usually iterative in nature and iterations usually imply a slow algorithm; it could be argued that sparse decompositions over highly redundant dictionaries are inherently slow. Nonetheless there are ways to make such problems tractable and get acceptable results, both in terms of computational time

and quality of decomposition, as is the case with MP that produces sub-optimal representations, relatively fast. Although computational speed of an algorithm is indeed an important requirement (an intractable problem is not worth solving) in certain cases it should not be the driving factor, as for example when dealing with high quality source separation. The point being that given an algorithm that is capable of perfectly separating individual sources from a mixture, then execution speed should not be the main concern. Relaxing the speed requirement allows an algorithm to use a 'brute-force' approach to the solution of a problem meaning that highly complex processing can take place. And this is one of the main ideas behind GMP and in fact a requirement, that is, the requirement of unlimited computing resources which is in contrast to real-time systems where an algorithm should use as few resources as possible. Without execution time being a main concern, one might ask what kind of technologies could be potentially useful when dealing with source separation?

At the end of section 1.3, source separation problems were categorised into over-determined, determined and under-determined. The over-determined and determined cases can be successfully solved by ICA techniques. Under-determined cases though require different approaches. No matter the case, when dealing with source separation, certain assumptions have to be made that lead to a mathematical model which describes the processes that occur within the mixture. The most difficult source separation case is that of a single channel, convolutive mixture comprising multiple sources, without having any prior information. As such, the first possible simplification is to consider linear instantaneous mixtures instead. Although this is a major simplification, when developing algorithms it is normal to start working with simplified versions of problems before trying solving more complex ones. But even for the instantaneous mixture, the single-channel case is very tricky, although there are several proposed approaches.

An easier problem to solve, is when the mixture is stereo and the configuration of a stereo mixture comprising multiple sources is of special interest since most professional recordings are in this format, so there is a big incentive in trying to separate stereo mixtures. The main advantage of having a stereo mixture is that the spatial information within the mixture can be used to determine the direction of arrivals of the individual sources. Depending on the recording technique used, the spatial information can be estimated either from inter-channel level differences or inter-channel time differences or both. SCA based techniques heavily depend on such information. The process of estimating the directions of arrivals is called mixing matrix estimation and is usually treated as a separate problem to source separation.

Another technology that can be used to our advantage is that of sparse adaptive representations. It is a fact the sparse representations lead to better quality source separation and this is the main reason why a time-domain mixture is always transformed into a new domain where the sources are sparsely represented and are easier to distinguish and separate. One main assumption regarding sparsity is that each decomposition coefficient (or the atom related to it) is attributed to a single source only. This is a crude assumption and is often violated, therefore some algorithms try to spread the contribution of an extracted atom to a source and its neighbouring sources. The task of atom extraction and allocation can be aided further by the use of a musical score. The musical score could be obtained, either manually or by estimating it. When considering automatic score estimation, technologies such as pitch and partial-tracking, note onset/offset

detection, instrument identification and rhythmic information extraction could be used. Score estimation is part of the field of music information retrieval (MIR) and many algorithms for all aforementioned subtasks already exist, so all these technologies could be incorporated into a source separation algorithm.

It is clear that the solution to the problem of high quality source separation cannot be obtained by a single technology but rather by the amalgamation of several technologies which leads to hybrid processing systems. And this is the main motivation behind the development of GMP; the desire to use as much a priori information as possible by taking advantage of already established and clever algorithms. In the past century the field of signal processing has evolved tremendously and at the moment we have a huge amount of analysis tools at our disposal, accompanied by incredibly fast computing systems. That is not to say that we should stop researching new technologies; this is a naive thought and progress into that domain is inevitable but maybe it is the right time to focus more on the development of hybrid solutions comprising well established technologies and algorithms.

Research into source separation tells us that sparsity is the way forward and the SCA framework is well suited to source separation of stereo mixtures. An important step to any SCA approach is the front-end where the signal is transformed from the time-domain into another domain where the individual sources can be sparsely represented. For this step a sparse decomposition technique is required. Many sparse decompositions depend on over-complete dictionaries which lead to sparse adaptive representations and as such, the dictionary used affects the sparsity and adaptivity of the produced decomposition. An important observation is that regardless of the type of the dictionary most sparse decomposition techniques start by either defining or training one. In certain cases, the implementation of the algorithms is dictated by the type of the dictionary where specific structuring of the dictionary can lead to algorithmic optimisations that reduce the computational complexity. This is particularly true for analytic dictionaries such as the wavelet and cosine packets dictionaries which are specifically designed to work with the wavelet packet decomposition, wavelet dictionaries which in practice are implemented using quadrature mirror filter banks and critically sampled multi-scale Gabor dictionaries which can be used for fast implementations of the MP algorithm.

The approach of deciding on a dictionary before the actual decomposition takes place, can pose problems especially when considering the implementation of the algorithm. For example assume a signal decomposition over a multi-dictionary, where a multi-dictionary comprises atoms of various types, like for example, Gabor atoms, Gaussian chirps and wavelets. Such a decomposition makes perfect sense, especially when musical signals are considered. For example, drums in a piece of music will be better represented by wavelets whereas instruments with harmonic content (or almost harmonic) could be better represented by Gabor atoms. One way to achieve such a decomposition would be to use specific types of dictionaries that lead to fast algorithmic implementations, as for example the MMP algorithm does, but this approach in itself poses new problems. First of all, it might be difficult to devise a generic algorithm that takes advantage of all the different types of dictionaries and secondly there are certain types of dictionaries that just cannot be designed so as to produce fast algorithmic implementations. Another way of using multi-dictionaries would be the brute-force approach where a dictionary matrix comprising all different types of atoms is created

and the decomposition algorithm acts on that matrix. This approach is used by MATLAB®'s MP implementation but leads to dictionary matrices that are impractical to use and store, especially when high-dimensional vectors are involved. The theoretical aspects of this problem have been discussed in the literature but not many practical implementations are available. The only software implementation, known to the author, that is capable of processing real-world audio signals, using multi-dictionaries is the MPTK software package. Although MPTK is excellent software, in its current form, it has certain limitations that did not allow it to be used in the context of this research. As such a new approach to the problem of using multi-dictionaries was desired and one that could take advantage of a priori information about the signal. The search for such an approach eventually led to GMP.

To conclude, a new sparse decomposition technique is desired with the aim of achieving the following main goals:

- *Adaptivity*. The decomposition should produce atoms that are well adapted to inherent signal structures.
- *Sparsity*. The decomposition should produce sparse representations, i.e. the signal should be expressed in terms of few significant atoms.
- *Use of a priori information*. The decomposition should make use of a priori signal information which will guide the update step of the algorithm.
- *Use of unlimited resources*. For practical implementations, the speed and the amount of memory used for the decomposition, should not be of main concern (of course within reasonable limits).

The following section describes in detail this new MP variant.

3.2 GMP Algorithm

The GMP algorithm is a simple modification of the MP algorithm presented in section 2.3.1, where a pre-processing step is included as the first step in the main sequence of events [172]. The initialisation of the algorithm is the same as with MP, where the residual signal $\mathbf{r}$ is initialised to the input signal $\mathbf{x}$ and the approximation signal $\mathbf{y}$ is initialised to zero. The main difference is that no dictionary is created or selected at this point; a new dictionary is created at every iteration of the algorithm. In particular, at every iteration the pre-processing step performs some kind of analysis on the residual and extracts useful information which is then used to create a mini-dictionary $\mathbf{D}_i$ that contains a small number of atoms. For example the pre-processing step could perform a Fourier analysis on the residual and extract the frequency and phase of the component with the maximum magnitude (for example by using equations (2.64) and (2.65)). This information could then be used to create a mini-dictionary containing Gabor atoms of that particular frequency and phase and of possibly different scales or even atoms with neighbouring frequencies that are greater than the frequency resolution of the Fourier analysis (i.e the resolution as given by equation (2.66)). In this example the component of maximum magnitude is selected since it most likely represents an important signal feature. The idea is that the newly created

atoms will correlate well with the corresponding frequency components of the residual. In this way, the pre-processing step acts as a guide for creating atoms that might best correlate with the features of the signal we are interested in, therefore this new approach is called guided matching pursuit [172]. Having obtained the mini-dictionary the algorithm continues in a similar fashion to the basic MP algorithm and the process is repeated to the newly obtained residual.

Using typical notation, let $\mathbf{x}$ be the input signal, $\mathbf{r}$ the residual and $\mathbf{y}$ the approximation signal, then the GMP algorithm goes as follows:

Algorithm 4 Guided Matching Pursuit

1: **Initialise** $\mathbf{r}_{i-1} \coloneqq \mathbf{x}$, $\mathbf{y}_{i-1} \coloneqq \mathbf{0}$, $i \coloneqq 1$
2: **Repeat**
3: *Pre-processing step*: perform some kind of analysis to the residual and extract important information. Use the extracted information to create a mini-dictionary $\mathbf{D}_i$ (i.e. with small number of atoms).
4: Compute the cross correlations between the residual and all atoms in $\mathbf{D}_i$ and get the coefficient matrix $\mathbf{C}_i \coloneqq XCORR\left(\mathbf{r}_{i-1}, \mathbf{D}_i\right)$
5: Select best atom $\mathbf{d}_{\gamma_i} \coloneqq \underset{\mathbf{d}_{\gamma_i} \in \mathbf{D}_i}{\arg\, max} |\mathbf{C}_i|$
6: Get expansion coefficient $c_i \coloneqq \langle \mathbf{r}_{i-1}, \mathbf{d}_{\gamma_i} \rangle$
7: Update residual $\mathbf{r}_i \coloneqq \mathbf{r}_{i-1} - c_i \mathbf{d}_{\gamma_i}$
8: Update approximation $\mathbf{y}_i \coloneqq \mathbf{y}_{i-1} + c_i \mathbf{d}_{\gamma_i}$
9: Increase iteration number $i \coloneqq i + 1$
10: **Until** stopping condition

Note that step 1 of the MP algorithm (section 2.3.1) is moved into step 3 in the GMP algorithm with the extra addition of a pre-processing step. The dictionary is not fixed before the decomposition but rather created dynamically at each iteration. Although this modification is minor (just a re-arrangement of the steps) it has major consequences, especially regarding the implementation aspects of the algorithm. A direct consequence of this approach is that we are not concerned about the creation of static dictionaries, either analytic or trained ones, but dynamic dictionaries, the creation of which is dictated by the actual input data to be processed. And this is where the pre-processing step comes into play. For example imagine that we want to decompose a piece of music, where information about the musical score, the number and type of instruments and the way they are mixed, is available a priori, either because they were already available or estimated in some way. A GMP algorithm could use all this information to find for example a location in the signal where a particular instrument is active and plays a specific note. Knowing the location and the musical key of the note and also the type of instrument is was generated from, then a dictionary can be created that comprises atoms which are expected to characterise that particular note event and therefore this dictionary will correlate well with the particular section of the signal during the inner-product computation step of the algorithm. So rather than letting the algorithm blindly extract atoms based on a greedy, in an energy sense, cost function we guide it to regions of interest within the signal and instruct it to extract atoms based on rules other than maximum energy reduction alone. The musical note example presented here is of course ideal and one could argue that the musical score and similar information is difficult to obtain or infer but first of all even an imperfect estimation of such information could be of benefit and secondly there is already a wealth of available processing techniques that could help guide a decomposition some of which will be discussed in subsequent sections of this chapter.

The idea of guidance is not new and has been proposed and implemented in some way or another but, as far as the author knows, certainly not in the way presented here. For example, the idea of guidance can be found in spectral modelling techniques where information extracted from the STFT decomposition of a signal is used to create quasi-stationary partials. Another example is given in the original MP paper where Mallat and Zhang suggest the use of a Newton search strategy in order to refine the estimated parameters of an already selected atom, that is, the initial correlation step acts as a guide by providing approximate atom parameter estimates which are then refined in a subsequent step, leading to a double search strategy [68]. A similar approach is followed in Gribonval's ridge pursuit [158] where a two-step pursuit strategy is used in identifying a locally optimal chirp atom. In this two-step approach the algorithm firsts selects the best Gabor atom and then the local behaviour of that atom's frequency is used to estimate a chirp-rate parameter and a refined scale parameter. Again the idea of guidance is clear; the search is guided to select the best Gabor atom and then information around that atom is used to select an optimal chirp atom. Another approach to chirp atom estimation is presented in [173] where the authors propose an algorithm that is more on par with GMP. In particular the proposed algorithm starts with the STFT representation of the input signal and selects the time-frequency point with the maximum magnitude and then iteratively selects neighbouring STFT points based on a couple of extension rules. This procedure results in time and frequency vectors that essentially hold the time-frequency coordinates of consecutive maxima of the STFT. The chirp rate is finally estimated by performing linear regression using those coordinates. Again the idea of guidance is evident but not stated. Another example can be found in [123] where Verma and Meng propose a perceptually weighted matching pursuit algorithm, where the atoms in the dictionary are weighted by psycho-acoustically derived perceptual masks; that is, the perceptual masks guide the selection of the atoms in the decomposition. The main difference with GMP is that all aforementioned algorithms try to solve a specific problem and as such authors speak about double search strategies, two-step pursuits or coefficient weightings which all suggest a guiding strategy of some sort; although the idea of guidance is implied it is never explicitly expressed. In this context, GMP provides a generic framework that could potentially be used to describe all aforementioned algorithms and more. Several examples of how to do so will be provided throughout this chapter.

3.3 Online vs Offline Processing

So far the treatment of MP and GMP has been mostly theoretical. When considering practical implementations there are several issues that need to be addressed. The first potential problem arises when we consider the effect of the decomposition on the boundaries of the signal. To explain further, an MP algorithm extracts atoms from various locations within the signal and some times the atom to be extracted happens to be at either edge of the signal. This issue is depicted in figure 3.1. The left sub-plot of figure 3.1 is an input signal, the middle sub-plot depicts an atom to be extracted and the sub-plot on the right shows the atom at an extraction location which for this example is at the left edge of the signal. Note that the atom in this example, in reality would not have correlated with the signal at that particular point in time; this example is exaggerated for illustration purposes.

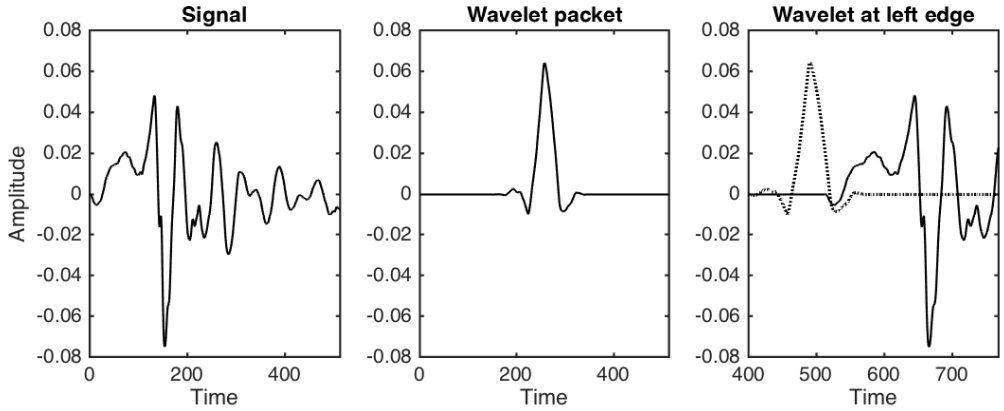


Figure 3.1: Demonstrating the MP boundary problem. The left subplot shows the original signal, the middle sub-plot a wavelet packet atom and the right sub-plot the atom plotted on top of the signal, at the location where it is going to be extracted from.

Carrying on with the decomposition of this example if the atom is extracted from this location then a so called pre-echo artifact will be introduced. This is demonstrated in figure 3.2. Sub-plot $a)$ depicts the original signal but zero padded from both sides; zero-padding is necessary in order to accommodate the extracted atom. Sub-plot $b)$ shows the new signal after the atom extraction and the introduction of extra energy, i.e. the pre-echo artifact, can be clearly seen both in sub-plot $b)$ and in more detail in sub-plot $c)$. Given this situation what is the best or correct way to treat this issue? This problem is barely addressed in the literature and it seems that the solution is implementation specific.

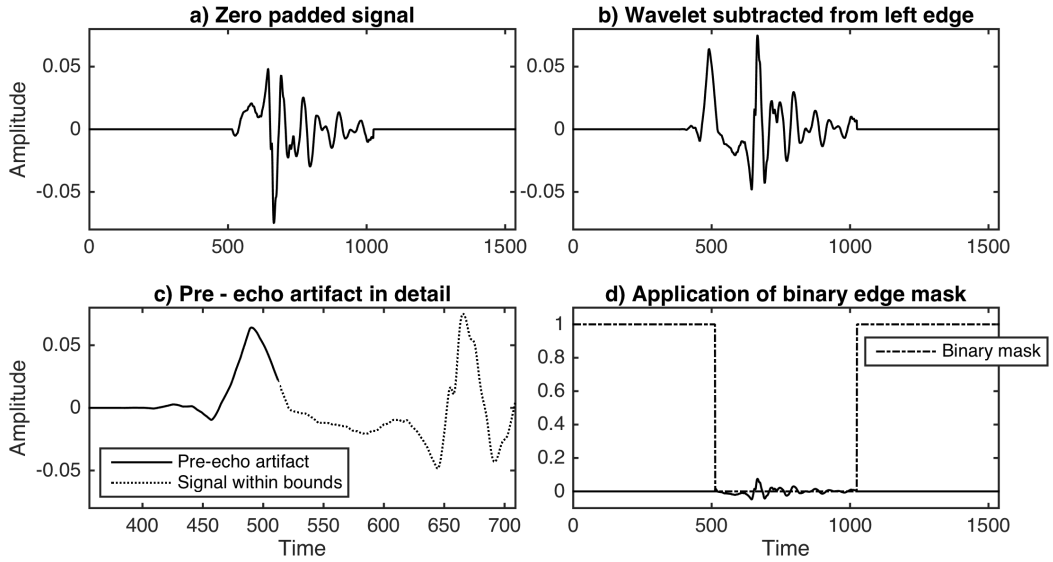


Figure 3.2: Demonstration of pre-echo artifact. An atom is subtracted from the original signal (sub-plot $a)$) to produce a new signal (sub-plot $b)$) where the pre-echo artifact can be seen. Sub-plot $c)$ shows a detailed version of the artifact. Sub-plot $d)$ is a potential solution; a binary mask is applied setting the edges to zero thus removing the artifact.

A potential solution would be to constrain the region of the decomposition towards the middle of a signal, thus never extracting an atom from the edges, but with this approach the signal will never be fully decomposed since the data of the signal at the edges will be left untreated. A similar approach would be to apply a window function to the signal in order to smooth the edges and produce similar results as the previous approach. Again though applying a window function would leave the edges of the signal untreated, unless the signal is long enough and is segmented

into overlapping frames, in which case the untreated potions of one frame will be treated by the next frame, a process similar to an STFT with overlap and add synthesis step. This is an interesting approach and will be discussed in the next section. MATLAB®'s MP implementation offers another solution by folding certain atoms around the edges. This folding is depicted in the middle sub-plot of figure 3.3. The left sub-plot of figure 3.3 depicts a centred wavelet packet atom and the right sub-plot shows the signal overlaid with the folded atom before extraction. Note that this approach is only relevant if the maximum length of the atoms in the dictionary is the same as the length of the signal so it is not feasible if the input signal is of much greater length.

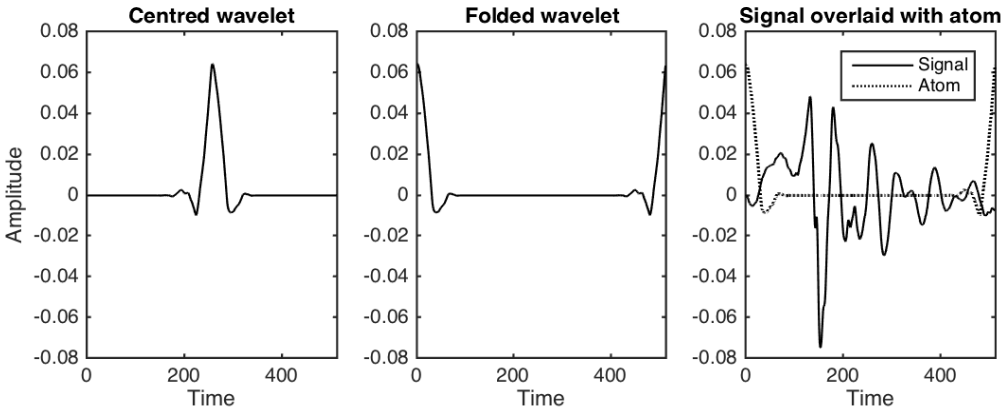


Figure 3.3: MATLAB®'s dictionaries contain atoms at several locations, and atoms found at the edges are folded around. The left sub-plot depicts an atom located at the center, the middle sub-plot shows a folded atom and the right figure shows a signal overlaid with the folded atom.

Yet another solution can be found in Mineault's MP implementation, which will be called temporal matching pursuit (TMP) [174][1], which uses a binary mask that essentially sets the zero-padded edges of the signal to zero after each iteration thus eliminating any introduced artifacts. This approach does not affect the reconstructed signals as long as the binary mask is also applied during the synthesis stage. Sub-plot $d)$ of figure 3.2 depicts this binary mask approach where the pre-echo artifact introduced by the atom subtraction is set to zero. Finally, Goodwin in [156] proposes the use of damped sinusoids for preventing the pre-echo artifact to be introduced in the first place. Although damped sinusoids are an interesting atom type capable of capturing transient phenomena within a signal and do indeed, in many cases, prevent pre-echo artifacts to be introduced, the solution they provide is not generic enough. Out of all the solutions presented here, Mineault's approach seems to be the most generic and can be applied to signals of any length. Of-course damped sinusoids and gamma-tone atoms (which will be introduced later) are important atom types and should be used when decomposing signals containing transients, since they can potentially eliminate any pre-echo artifacts introduced within the signal in the case the signal length is much greater than the atoms length.

Having considered all the aforementioned options it seems that the best generic solution to the boundary problem would be to first zero pad the signal and then apply a binary mask at each iteration of the algorithm. To formalise this process, let $x[n]$ be the input signal, N_x be the length of the input signal in samples, N_{max} be the maximum length of the atoms in samples, and define

[1] `http://uk.mathworks.com/matlabcentral/fileexchange/32426-matching-pursuit-for-1d-signals`

the padding size as:

$$N_{pad} = \lceil c_{zp} N_{max} \rceil \tag{3.1}$$

where $c_{zp} \geq 1$ is a constant that affects the length of the zero padding and the ceil function notation is used because the multiplication $c_{zp}N_{max}$ must produce an integer number, then a zero-padded signal will be given by:

$$x_{zp}[n] \coloneqq \begin{cases} 0, & n = 0, \ldots, N_{pad} - 1 \\ x[n], & n = N_{pad}, \ldots, N_{pad} + N_x - 1 \\ 0, & n = N_{pad} + N_x, \ldots, 2N_{pad} + N_x - 1 \end{cases} \tag{3.2}$$

Note that the length of the zero padding is expressed in terms of the maximum atom length N_{max}. Also the new length of the zero-padded input sequence will be:

$$N_{zp} = 2N_{pad} + N_x \tag{3.3}$$

In a similar fashion to equation (3.2) the binary mask can be defined as:

$$m[n] \coloneqq \begin{cases} 1, & n = 0, \ldots, N_{pad} - 1 \\ 0, & n = N_{pad}, \ldots, N_{pad} + N_x - 1 \\ 1, & n = N_{pad} + N_x, \ldots, 2N_{pad} + N_x - 1 \end{cases} \tag{3.4}$$

The binary mask can be applied to a signal by:

$$x_{zp}[n \cdot m[n]] = 0, \quad n = 0, \ldots, N_{zp} - 1 \tag{3.5}$$

where the symbol '$\cdot$' denotes multiplication in this case. When used in the GMP algorithm, the binary mask is applied after the update steps to both the residual and the approximation signals.

Another potential issue has to do with the length of the input sequences. To put things into perspective, a typical three minute musical signal, sampled at the standard sampling frequency of 44100 Hz will be $3 \times 60 \times 44100 = 7938000$ samples long, whereas the length of atoms, although not restricted to a particular value, usually ranges from a few hundred to a few thousand samples. So the question of how to treat such long sequences naturally arises. But yet again this issue is not widely discussed in the literature and is implementation specific. In fact all early literature on MP, presents examples where the input sequences are very small and all treatments are mathematical without any practical suggestions on how to extend the algorithm for long sequences. Even recent literature follows similar approaches with most provided implementations not being able to cope with long sequences; it seems that the main concern is the theoretical development of algorithms and the supplied implementations act as proof of concepts rather than production ready code. For example MATLAB®'s BMP implementation cannot deal with input sequences more than a few thousand samples long because the lengths of the input and the atoms in the dictionary are coupled. BMP is a 'textbook' implementation of MP and the produced dictionary is a matrix where its columns contain atoms at various locations, scales and frequencies so the calculation of the inner-products at each iteration reduces to a simple vector-matrix multiplication. Although this leads to a very straightforward implementation it cannot be used with long input sequences;

BMP is definitely not tailored to process audio signals.

Mineault's TMP implementation uses computational tricks similar to MPTK, that speed up the MP decomposition and allows the processing of longer sequences but there is a limit to the length of sequences and also the size of the dictionary that can be used and TMP cannot cope when these become very large in size. It seems that the only, publicly available, implementation able to cope with real life musical signals is the MPTK software package described in [154]. MPTK makes use of the STFT, MDCT and other transforms, for the fast implementation of MP. Each such transform constitutes a 'block'. For example a Gabor block is equivalent to a single scale STFT of the signal. Each block can find its maximum correlated atom and also update its inner-products along particular supports of the signal. Several such blocks can be used simultaneously to provide multi-scale or multiple-basis analysis [154]. Blocks are contained within another object called the dictionary which is responsible for finding the 'best' atom by using fast search tree algorithms. In short, the main idea is to compute all inner products during initialisation by using fast transforms or fast convolution algorithms and at each iteration the computed inner-products are updated along a particular support of the signal, i.e. locally. There is a lot to be learned from the MPTK architecture but still this is not the only way to treat long duration sequences. Other interesting approaches involve frame based techniques used in spectral modelling and specifically hybrid techniques that use sinusoidal modelling and MP, like for example in [123], [112], [175], [176] and [177]. The following subsections present a few approaches on how to treat long duration signals, some of which are generic enough to be used with pretty much any MP implementation.

3.3.1 Short-Time Matching Pursuit (STMP)

The short-time matching pursuit, as the name suggests, is similar in nature with the STFT, where in the analysis stage, a signal is segmented into possibly overlapping frames, but instead of computing the Fourier transform of each frame an MP decomposition takes place. The synthesis stage can be performed by an overlap-add technique. This approach is of particular interest since it can lead to real-time implementations of MP. A real-time implementation is based on stream-processing, which is a memory-efficient technique for handling large streams of data, like audio for example [178]. In stream-processing, also referred to as on-line processing, the input signal is segmented into frames, which could overlap or not, and each frame is processed fully before the next one arrives [178]. This is opposite to off-line processing where a physically stored signal is processed as a whole. Traditionally, stream-processing applications use the FFT or other fast transforms for processing the incoming data, but given a relatively small dictionary and powerful enough hardware a real-time MP decomposition is certainly possible. Of course the idea of frame-based processing using MP is not new, but the real-time application of MP has not been researched and the term STMP has not been used before (at least according to Google) despite the fact that MP has been applied to sequential short segments of data and STMP exactly characterises this technique. Stream-processing is not the only reason to follow the STMP approach. Even in an off-line setting the decomposition of a signal can be sped up by processing frames in parallel using multi-threaded programming techniques. In off-line processing the order of how the frames are processed does not matter and as long as they are overlapped and added in the correct sequential order, the resulting decomposition will be successful and meaningful.

Implementation wise, STMP is similar to the STFT as described in section 2.1.5 but unlike the STFT, the FFT processing is not necessary; instead an MP decomposition takes place. Therefore we are only interested in the overlap-add decomposition described in [179][2]. First of all, let us define the matching pursuit operator for a sequence x:

$$\mathcal{MP}[x](\gamma) \stackrel{\text{def}}{=} MP\{x[n]\} \tag{3.6}$$

where $x[n]$ is an input sequence, MP can be any matching pursuit variant and $\gamma \in \Gamma$ is the parameter index. Now assume we want to segment an input sequence x into frames using a window w of length M. Then the s^{th} windowed frame can be defined by [179]:

$$x_{sR_a}[n] \stackrel{\text{def}}{=} x[n]w[n - sR_a], \quad n \in (-\infty, +\infty) \tag{3.7}$$

where s is the frame index and R_a is the analysis frame step, or hop size. The hop size is essentially the number of samples that each successive window is advanced for and also dictates the amount of overlap. For example in order to achieve a 50% overlap then $R_a = M/2$ (where M is the window length).

The STMP analysis can be expressed as:

$$\{B(sR_a, \gamma), r_{sR_a}[n]\} = MP\{x_{sR_a}[n]\}, \quad n \in (-\infty, +\infty) \tag{3.8}$$

or

$$\{B(sR_a, \gamma), r_{sR_a}[n]\} = \mathcal{MP}[x_{sR_a}](\gamma) \tag{3.9}$$

where the output $B(sR_a, \gamma)$ is a structure that stores the parameters of all extracted atoms for each frame and $r_{sRa}[n]$ is the residual of each frame. In MP literature the structure B is called a 'book'. In practice the 'book' can be a $Q \times P \times S$ matrix where $Q \geq 0$ denotes the number of atoms, P the length of the parameter index γ, and S the number of data frames, therefore the 'book' can be defined as:

$$\begin{aligned} B(sR_a, \gamma) \stackrel{\text{def}}{=} \mathbf{B}_{p,q,s} \equiv \mathbf{B}(p, q, s) \\ q = 1, \ldots, Q \\ p = 1, \ldots, P \\ s = 1, \ldots, S \end{aligned} \tag{3.10}$$

The original signal can be obtained by first summing all atoms of each decomposed frame and adding the residual (if any) to produce the reconstructed frames and then by summing the reconstructed overlapping frames in their original time positions. The synthesis stage can be expressed as:

$$\hat{x}[n] = \sum_{s=-\infty}^{\infty} x_{sR_s}[n] \tag{3.11}$$

$$x_{sR_s}[n] = \sum_{q=0}^{Q-1} c_q d_{p_q}[n] + r_{sR_s}[n] \tag{3.12}$$

[2] http://ccrma.stanford.edu/~jos/sasp/Overlap_Add_Decomposition.html

where s is the frame index, R_s the synthesis hop size, c_q and d_{p_q} are the expansion coefficients and the atoms in the book respectively and $r_{sR_s}[n]$ the residual of each frame. From equation (3.11) we get the following:

$$\begin{aligned}\hat{x}[n] &= \sum_{s=-\infty}^{\infty} x_{sR_s}[n] \\ &= \sum_{s=-\infty}^{\infty} x[n]w[n-sR_s] \\ &= x[n] \sum_{s=-\infty}^{\infty} w[n-sR_s]\end{aligned}$$

therefore $\hat{x}[n] = x[n]$ if and only if:

$$\boxed{\begin{gathered}\sum_{s\in\mathbf{Z}} w[n-sR_s] = 1, \quad \forall n \in \mathbf{Z} \\ \text{and} \\ R_a = R_s\end{gathered}} \qquad (3.13), (3.14)$$

Equation (3.13) is known as the constant-overlap-add (COLA) constraint for the window w [179] and equation (3.14) requires the analysis and synthesis hop sizes to be equal. As long as equations (3.13) and (3.14) are obeyed then the overlap and add procedure will yield perfect reconstruction of the original signal x [179]. In case the window w does not obey the COLA constraint exactly, then modulation artifacts will appear in the output signal but in practice these can be reduced to a certain extent by an element-by-element division of the output signal with the window envelope produced by the COLA equation. More specifically, let the window envelope be assigned as:

$$\mathbf{w}_{env} \coloneqq \sum_{s=0}^{S-1} w[n-sR_s] \qquad (3.15)$$

where s is the frame index, S the maximum number of frames and R_s the synthesis hop size and let $\mathbf{y}$ be the modulated approximation output produced by the synthesis stage of the STMP, then $\mathbf{y}$ can be demodulated by:

$$\mathbf{y} \coloneqq \mathbf{y} \oslash \mathbf{w}_{env} \qquad (3.16)$$

Aside the computational benefits, the STMP analysis/synthesis framework is a valuable technique in its own right. First of all and STMP analysis/synthesis system is an identity system. Figure 3.4 shows the results of an STMP of the 'quake' signal with the top sub-plot depicting the original signal, the middle sub-plot the approximation after synthesis and the bottom sub-plot the residual. It can be clearly seen that the approximation is very similar to the original and the residual has little remaining energy. Audition of the signals also reveals the similarity with one noticeable difference being the lack of some high frequency content in the approximation but certainly not in a degrading way. The reason for this is the low number of atoms used for the decomposition, so increasing this number will increase the quality in the approximation signal.

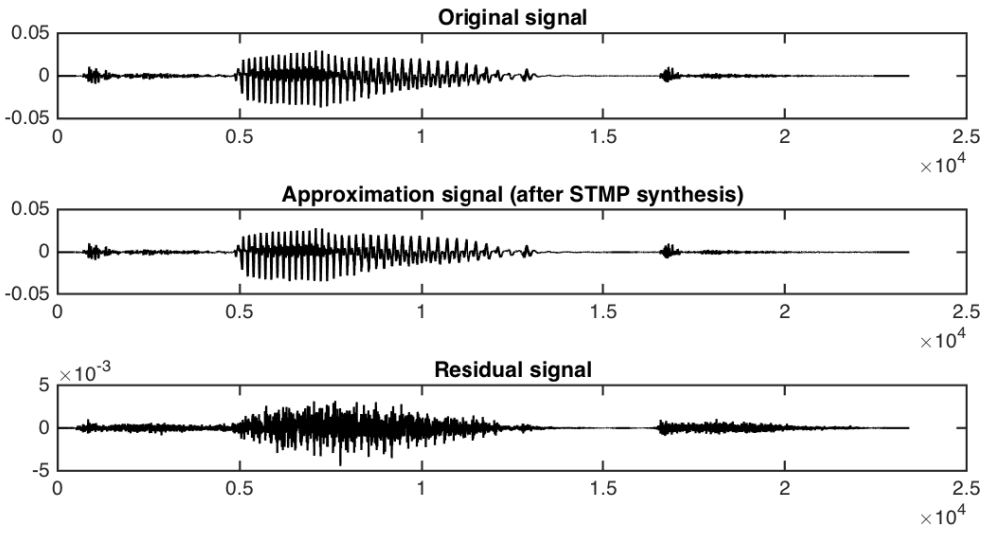


Figure 3.4: STMP decomposition waveforms. The top plot depicts the original signal, the middle plot the approximation signal after the STMP synthesis stage and the bottom plot the residual. It can be clearly seen that the original and the approximation signals are very similar. Audition of the signals also verifies the similarity. When listening to the signals the main noticeable difference is the lack of high frequency content but definitely not in a degrading way. Finally the residual signal has very little remaining energy.

For this particular STMP a 1024 samples long hamming window was used with a hop size of 128 samples which are the same settings used for the STFT representation shown in figure 2.14. The GMP decomposition was performed using 20 atoms per frame. Figure 3.5 depicts the STMP representation of the 'quake' signal. Observing figure 3.5, the generic time-frequency pattern of the 'quake' signal is clearly visible and it seems that the main signal features have been captured successfully. Comparing that to the STFT representation of figure 2.14 it can be seen that the STMP representation is much clearer. This is a direct consequence of the sparse adaptive nature of MP; in STFT pretty much all time-frequency points (or tiles or atoms) contain some energy whereas in the STMP representation the visible energy belongs to extracted atoms only. Using a 1024 point FFT and the aforementioned settings for the window length and the hop size the signal is split into 176 frames and the time-frequency plane is segmented into $176 \times 513 = 90288$ time-frequency points. In the STFT representation most of these points contain some energy whereas in the STMP representation only $176 \times 20 = 3520$ atoms are active, therefore the STMP representation uses only $(3520/90288) * 100 = 3.9\%$ of the time-frequency plane. It is clear that STMP is a much sparser representation compared to the STFT. Although, similarly to the STFT, the STMP is a redundant representation since the analysis frames overlap, so a traditional MP decomposition is even sparser which is evident by figure 2.27 where only 256 atoms were used for decomposing the whole signal.

Although the STMP and STFT representations depicted in figures 3.5 and 2.14 respectively are similar and seem to provide the same information in reality this is not the case. More specifically the STFT output is the complex FFT output of each data frame whereas the STMP output is the MP decomposition of each frame. As already explained in detail in chapter 2 the FFT decomposes a signal against a single fixed basis and MP is an sparse decomposition using multiple bases. In the context of sparse decompositions the FFT output provides us with the frequency, phase and magnitude information of fixed scale atoms whereas the output of MP, that is the 'book' matrix (or structure), contains much more information including frequency, phase and magnitude but

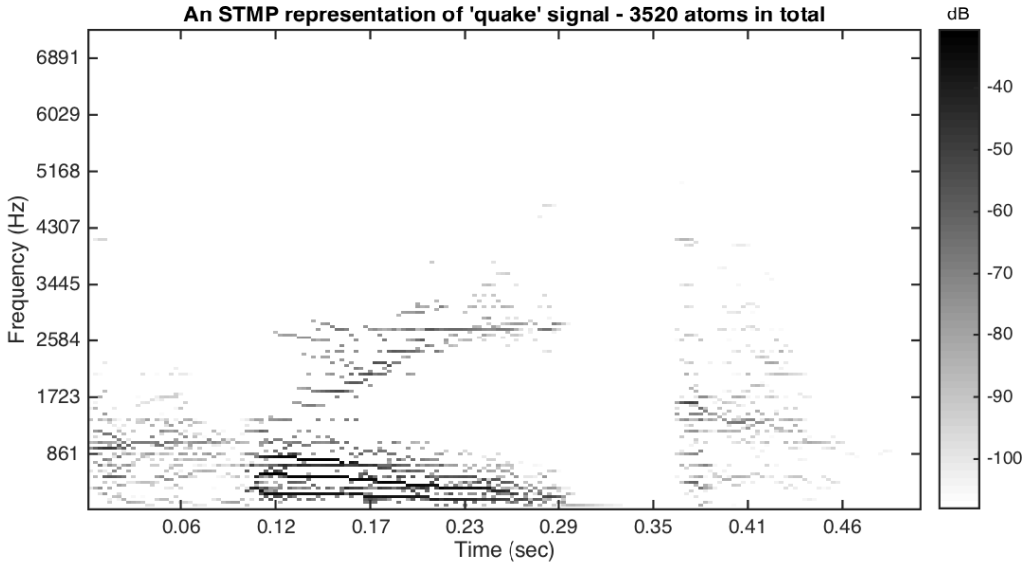


Figure 3.5: An STMP representation of the 'quake' signal. The settings used for this decomposition are the same used for the STFT of figure 2.14. In particular NFFT=1024 with an 8x overlap. Each frame of the STMP was decomposed into 20 atoms. The resulting decomposition has a total of 3520 atoms, which is more than the number of atoms in figure 2.27 but much less than the STFT representation. This figure is more sparse than the STFT equivalent and the quality of the original and the approximation signals is almost the same.

also location of the atom, scale and depending on the dictionary used, the atom types and other parameters related to specific atom types like for example chirp-rates for chirp atoms, damping factors for damped sinusoids and shape and scale parameters for gamma-tone atoms to name a few. So although the STMP representation in figure 3.5 is useful for inspecting how the energy of the signal is distributed in the time-frequency plane it is not correct in the sense that any required modifications cannot take place on that representation directly as is the case with the STFT output. Instead the 'book' matrix has to be manipulated. To explain further in the STFT representation the energy of each time-frequency point is attributed by a single atom but in the STMP representation the energy of each time-frequency point might be attributed by several atoms. This happens because during the MP decomposition an atom of the same frequency, but maybe of different scale, might be selected more than once unless the OMP variant is used. So an STMP representation needs to correctly treat this atom overlap.

One possible solution is the one already discussed, where the energy of atoms of the same frequency is added and these atoms are represented by a single atom in the time-frequency plane (this approach was used for generating figure 3.5). When using this approach though a lot of information is lost, such as the location, scale and phase of the atoms, therefore this representation should be mostly used for visualisation purposes. Having said that, preliminary research into this issue has shown that by directly feeding the STMP representation into the synthesis part of a phase vocoder system produces very interesting robotisation/vocoding effects which could be used creatively. Also by considering the sparseness of this representation it could be possible to directly use it as input to systems that do not produce an audible output but rather meta-parameters. Potential applications include classification using neural networks and audio identification in acoustic fingerprinting technologies. The STMP provides a sparse representation (sparser than the STFT) in a similar format to the STFT and since many systems already are

designed based on the STFT as the front-end it would be interesting to see what happens when the STMP is used instead.

Another possible solution to the atom frequency overlap problem is to separate similar atoms into their own STMP representations. This approach essentially produces a 3-D representation (or 4-D if we consider colour as well) where the x-axis represents time, the y-axis represents frequency and the z-axis represents different layers of STMP representations. These layers will be called 'Z-layers' (for lack of a better name). It should be noted that, although during the MP decomposition we are free to choose any kind of dictionary, it is best if the frequency of the parameter space is discretised in a similar way to the Fourier dictionary, by letting $\omega_k = 2\pi k/N$ for $k = 0, \ldots, N_{o2}$ where N is the maximum atom length and N_{o2} is half of N. Doing so simplifies the implementation considerably. Let us denote $\mathbf{B} \in \mathbf{R}^{Q \times P \times S}$ the output of the STMP, with Q denoting the number of atoms, P the number of parameters and S the total number of frames. Also let us denote with ξ the p^{th} column of $\mathbf{B}$ that represents frequency and with α the p^{th} column that represents the expansion coefficients. Finally let the number of the Z-layers be denoted by Z. Now a 3-D STMP representation can be obtained by algorithm 5.

The results produced by algorithm 5 are depicted in figure 3.6. All sub-plots of figure 3.6 depict a portion of the 'quake' signal, specifically the region where the triphthong occurs. The top three sub-plots show 2D views of each plane (XY, XZ and YZ) and the bottom plot shows a 3D view. Note that the way the Z-layers are ordered depends on step 26 of the algorithm. In this case the same frequency atoms are ordered depending on the magnitude of their expansion coefficients but other orderings are possible, such as by location, scale even atom type if a multi-dictionary was used for the decomposition. Step 32 produces the typical 2D view of a time-frequency representation and the results of this are depicted in figure 3.5. Finally note that if the frequencies of the atoms are not descretised as suggested earlier then extra logic needs to be incorporated into algorithm 5 to accommodate for a variable size of the frequency dimension of $\mathbf{E}$.

From the discussion so far it is clear that the STMP is a versatile representation and could be useful to numerous applications. In fact preliminary research using this technique has shown that source separation following a SCA framework is possible. Despite the redundancy, STMP is sparser than the STFT and provides a similar representation that could be used as a 'drop in' replacement to the STFT. This section has discussed only the fundamentals of this new technique which should be explored in more detail and could be a topic for future research.

Algorithm 5 3-D STMP representation

function FINDZLAYERS($\mathbf{B}$)
  Initialise $Z \coloneqq 0$, $[Q, P, S] \coloneqq size(\mathbf{B})$
  for $s \coloneqq 1 : S$ **do** ▷ for every frame
    for $q \coloneqq 1 : Q$ **do** ▷ for every atom in the current frame
      $k \coloneqq \mathbf{B}_{q,\xi,s}$ ▷ get the first frequency
      if $k == 0$ **then**
        continue ▷ skip to next iteration for zero frequency
      end if
      $\mathbf{k}_c \coloneqq find_q(\mathbf{B}_{q,\xi,s} == k), \quad \forall q$ ▷ find all common frequencies in $\mathbf{B}$
      if $length(\mathbf{k}_c) > Z$ **then** ▷ $length$ returns number of elements
        $Z \coloneqq length(\mathbf{k}_c)$ ▷ update Z
      end if
    end for
  end for
  return Z
end function

function COMPUTESTMP3D($\mathbf{B}$,N_{o2},Z)
  Initialise $[Q, P, S] \coloneqq size(\mathbf{B})$, $\mathbf{E} = zeros(S, N_{o2}, Z)$, $\mathbf{E}_{total} = zeros(S, N_{o2})$
  for $s \coloneqq 1 : S$ **do**
    $\mathbf{c} \coloneqq \mathbf{B}(q, s, \alpha), \forall q$ ▷ get vector containing expansion coefficients
    $\mathbf{f} \coloneqq \mathbf{B}(q, s, \xi), \forall q$ ▷ get vector containing frequencies
    for $k \coloneqq 1 : N_{o2}$ **do**
      $\mathbf{k} \coloneqq find(\mathbf{f} == k)$ ▷ find all same frequencies within $\mathbf{f}$
      $K \coloneqq length(\mathbf{k})$ ▷ get number of similar frequencies
      if $K > 1$ **then**
        $\mathbf{k}_{sorted} \coloneqq sort(|\mathbf{c}(\mathbf{k})|, \text{'descent'})$ ▷ sort in descending magnitude
        $\mathbf{k} \coloneqq \mathbf{k}(\mathbf{k}_{sorted})$ ▷ permute $\mathbf{k}$ vector
      end if
      for $j \coloneqq 1 : K$ **do**
        $\mathbf{E}(s, k, j) \coloneqq |\mathbf{c}(\mathbf{k}_j)|^2$ ▷ assign energy to each time-frequency point
      end for
      $\mathbf{E}_{total}(s, k) \coloneqq sum(|\mathbf{c}(\mathbf{k})|^2)$ ▷ assign summed energy to each TF point
    end for
  end for
  return [$\mathbf{E}$, $\mathbf{E}_{total}$]
end function

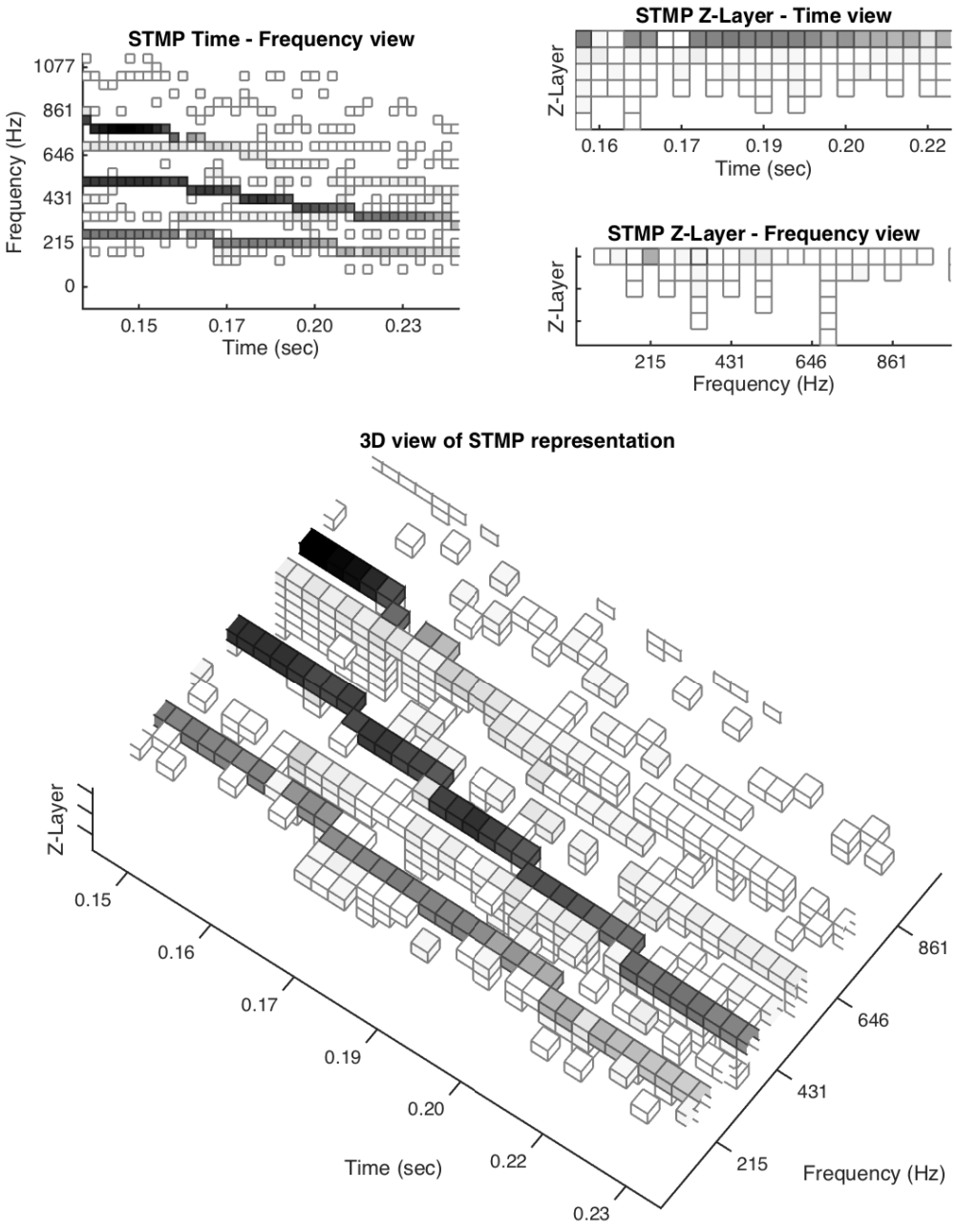


Figure 3.6: STMP 3D representation of triphthong region in 'quake' signal. The way to produce this figure is described in algorithm 5.

3.3.2 Long-Frame Matching Pursuit (LFMP)

Long-frame matching pursuit (LFMP) refers to any MP decomposition that acts on a stored copy of a signal as a whole, therefore such approaches are limited to off-line processing. Many of the implementations discussed at the beginning of section 3.3 are based on this category. The main idea is to pre-compute all inner products between a given dictionary and the signal and at each iteration of the algorithm update the inner-products at particular supports along the signal, meaning locally. By computing the inner-products between all atoms in the dictionary it is possible to use the simple update formula given by equation 2.143 to efficiently update the correlations between the signal and the dictionary. The direct implementation of 2.143 is limited by available memory storage, but certain dictionaries are designed in such a way so that only a few inner-product calculations between the atoms are needed. The analytic formula for fast computation of inner products between Gabor atoms is provided in [68]. Also the MPTK implementation makes use of fast transforms, such as the STFT and the MDCT, as well as fast convolution algorithms to speed up inner-product computations. However certain types of dictionaries do not lead to fast implicit implementations and the only way for fast inner product computation is to use fast convolution algorithms. These algorithms are based on the convolution property of the FT which states that convolution of two sequences in the time domain equals multiplication of their spectra in the frequency domain. The use of the FFT allows for fast convolution implementations and by using the time reversal property of the FT, fast cross-correlation which is the actual computation needed for MP. But even these algorithms start to slow down when the dictionary to be used becomes very large in size.

GMP tries to tackle the issue of decomposing long duration signals by using a guided approach. An important point to be made regarding the pre-processing step of GMP is that although it is included in the main body of the algorithm it is not necessary to be performed at every iteration. To elaborate further, certain processing such as, for example, mixing matrix, musical score and power spectrum estimation only have to be performed once, before the actual signal decomposition starts. Such types of pre-processing result in 'guide maps', or depending on the context and the processing performed 'feature maps', which will drive the rest of the decomposition process and at each iteration only particular supports of these 'maps' need to be updated. Although this approach sounds similar to the MPTK architecture, it is not. The main difference is that the MPTK uses the STFT and other fast transforms to produce its so called 'blocks' that are essentially dictionaries of atoms whereas in GMP the STFT and possibly other transforms and processing, are used as guide maps and at each iteration a small dictionary is used, the creation of which is dictated by those pre-computed maps. The idea of GMP is more similar to the model based MP in [173] where an STFT with a small analysis window is used as a starting point for the creation of chirp atoms that are subsequently correlated with the residual and subtracted. The LFMP approach of GMP along with several types of pre-processing are discussed in detail in section 3.4.

3.3.3 Variable-Length Matching Pursuit (VLMP)

Variable-length matching pursuit (VLMP) refers to the process of segmenting an input signal into non-overlapping frames of variable length and then performing an MP decomposition on each frame. Similarly to LFMP this technique is only suitable for off-line processing but since the signal is segmented the computation of the decomposition can be sped up by processing frames in parallel using multi-threading programming similarly to off-line STMP. The main idea behind this approach is that a signal can be segmented into frames which represent important temporal events within the signal. For example given a music signal, a musically meaningful segmentation could be based on song structure or even smaller events such as note events. These types of processing have been researched in the field of music information retrieval and several techniques are available.

For example signal segmentation can be based on a note-onset detection scheme. Such methods first compute an onset detection envelope showing bursts of energy corresponding to successive pulses and then a pick peaking algorithm is used to estimate the positions of the notes. In certain signals the detecting positions might not be notes in the musical sense but positions of high bursts of energy indicating an important change in the signal. Another approach is based on self-similarity matrix of spectral data [180]. The procedure is based on the construction of a similarity matrix calculated from the inter-frame spectral similarity of spectral data. The similarity matrix can be based on different spectral data such as the FFT spectrum or the Mel-frequency cepstrum. Segmentation can also be based on detection of harmonic changes in audio signals as described in [181]. An excellent MATLAB® toolbox that provides several segmentation algorithms (including the ones mentioned here) is the MIRtoolbox described in [182][3].

A VLMP approach could be used to perform MP decompositions using different dictionaries for each data segment. For example splitting a musical signal into segments that correspond to song structure, an MP decomposition could use dictionaries that are tailored to the audio characteristics of each segment which could lead to better adapted and probably sparser decompositions. A potential application could be variable bit-rate encoding for signal compression. The VLMP approach is not pursued any further in this work but it was mentioned here because it does provide an alternative way to process long duration signals and also fits well with the concept of guidance proposed by the GMP framework.

3.4 The Pre-Processing Step

In section 3.2 the GMP algorithm was introduced and the inclusion of a pre-processing step in the traditional MP algorithm was proposed as a way to achieve a 'guided' decomposition. Also several possible kinds of processing were mentioned for implementing this step such as mixing matrix, musical score and spectrogram estimation. What kind of pre-processing to choose depends on the problem at hand but for audio signals the spectrogram is a reasonable choice. The rest of this section presents several FFT based algorithms that could be used as the pre-processing step of GMP.

[3] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

3.4.1 FFT Processing

Probably the simplest pre-processing step for GMP is the FFT. Of course as explained in section 2.1.3 the FFT would make sense for small length signals since it is an idealised representation. Applying a single FFT to a few seconds of an audio signal would be impractical first of all because the FFT size will be very large but most importantly because the time information of the signal is lost. Therefore the application of FFT is useful for small data segments and thus it is well suited to be used in an STMP setting.

Using typical notation for the input, approximation and residual signals and by denoting the FT of the residual as $R(\omega)$, the frequency bin index as k and the length of the FFT as $N_{fft} \geq N$, where N is the length of the input signal, then the pre-processing step using the FFT could be performed as follows: where the magnitude spectrum can be obtained using equation (2.64).

Algorithm 6 FFT pre-processing

...

1: $R^{i-1}(\omega) \coloneqq FFT(r^{i-1}(t), N_{fft})$ ▷ compute FFT of residual

2: $k_i \coloneqq \underset{k_i \in R(\omega)}{\arg\, max} |R^{i-1}(\omega)|$ ▷ get frequency bin with maximum magnitude

3: Create a mini-dictionary $\mathbf{D}_i$ comprising real atoms, of possibly different types, different scales and normalised frequency k/N_{fft}

...

If desired the phase of the selected component can be obtained from the phase spectrum by using equation (2.65). Phase might be important for example when dealing with the generation of complex atoms or when other kinds of processing need to take place (e.g estimation of direction of arrival given multi-channel signal) but for real atom generation it is not needed. Note that algorithm 6 is not the full GMP algorithm; it only shows a possible pre-processing implementation. In order to get a full GMP algorithm using the FFT, step 3 of algorithm 4 should be replaced with algorithm 6.

A single iteration of GMP using the FFT as the pre-processing step is illustrated in figure 3.7. In this figure the plot in the first row depicts a 1024 samples long excerpt from a spoken vowel 'a'. The plot in the second row depicts the input's magnitude spectrum. For illustration purposes, only the first few frequencies are shown. Step 2 of algorithm 6 finds the frequency bin with the maximum magnitude and in this example it is found to be frequency bin number 17. The next step is to create a mini-dictionary comprising atoms of that frequency but different scales. The plots in the third row of figure 3.7 depict three Gabor atoms of same frequency, different scales and of unit norm. The plots in the fourth row depict the absolute value of the cross-correlations between the atoms and the signal. Note that the output of each cross-correlation is a new signal in the time domain. These correlations are essentially the coefficient matrix of step 4 in algorithm 4. The next step in GMP is to find the absolute maximum correlation value and in this example this maximum occurs for the third atom with 'coordinates' (u, a) where u denotes the location in the time axis and a, the value of the y-axis, is essentially the magnitude of the expansion coefficient. It should be noted that a at this stage is positive since it was obtained from the absolute values of the correlation but before proceeding to the next step the actual sign of a needs to be obtained from the coefficient matrix. Assuming that the coefficient matrix is $\mathbf{C} \in \mathbf{R}^{N \times Q}$ where N is the length of the atoms (and the signal in this case) and Q the number of atoms in the dictionary then the actual value of the expansion coefficient can be obtained from $a \coloneqq \mathbf{C}(u, idx)$, where idx

is called the atom index and indicates the column of $\mathbf{C}$ where the maximum expansion coefficient occurs, which for this example is the third column. Having obtained these parameters the selected unit norm atom is multiplied with the expansion coefficient a and placed at location u in order to be subtracted form the signal. The first plot of the fifth row in the figure depicts the signal overlaid with the 'best' correlated atom. Note how the overlaid atom matches the underlying signal. Comparing this new atom with the original depicted on the right sub-plot of row three, it can be seen that their orientations are different. The different orientation of the new atom is due to the multiplication of the unit norm atom with the expansion coefficient, which in this case happens to have a negative sign thus flipping the atom along the x-axis. Also it is important to note that the phase of the new atom will be adjusted as well depending on the location u. This is an important observation because it means that there is no need to define a phase parameter when creating the atoms in the dictionary, as was the case in this example, where all atoms were generated using zero phase. The phase information is essentially dictated by the location parameter and the sign of the expansion coefficient. Finally the second plot of the fifth row shows the result of subtracting the atom from the signal and what is left is the updated signal. This whole process demonstrates a single iteration of the GMP algorithm using an FFT pre-processing step. The process repeats on the updated signal until a stopping condition is reached.

In this case the main information obtained from the pre-processing step is the frequency of the component with the maximum magnitude. This information alone is enough to allow the creation of a group of atoms that will correlate well with that signal component, but there is much more processing that could be performed at this stage to either improve the frequency estimate or get more information about other components of the signal. For example a peak picking algorithm could be used to select the K most prominent components and allow the atom selection step to extract multiple atoms instead of only one, leading to a multiple atom extraction algorithm, an approach also suggested in the original paper [68].

Further processing could help determine harmonic relationships between signal components thus leading to a dictionary comprising harmonically related atoms. During the cross-correlation step the 'best' correlated and harmonically related atoms could be grouped together to form harmonic molecules. This could also be achieved without a peak picking algorithm by just letting the dictionary creation step deal with this process via pre-specified atom creation rules. For example assume that a frequency bin k was selected at step 2 of algorithm 6, then step 3 could create a number of atoms that are (almost) harmonically related by using $k_i \approx ik, i = 1, \ldots, K$. Additional processing could allow the fundamental frequency to be found thus getting extra information that could help with the dictionary creation. Another dictionary rule could create atoms that are closely spaced in frequency such as $k_i = k + i\lambda,\ i = -K, \ldots, K$ with $K \in \mathbf{Z}$, where the value of $\lambda \in \mathbf{R}^+$ dictates the frequency spacing between the generated atoms. The software developed as part of this research provides an interface that allows several such atom creation rules to be defined.

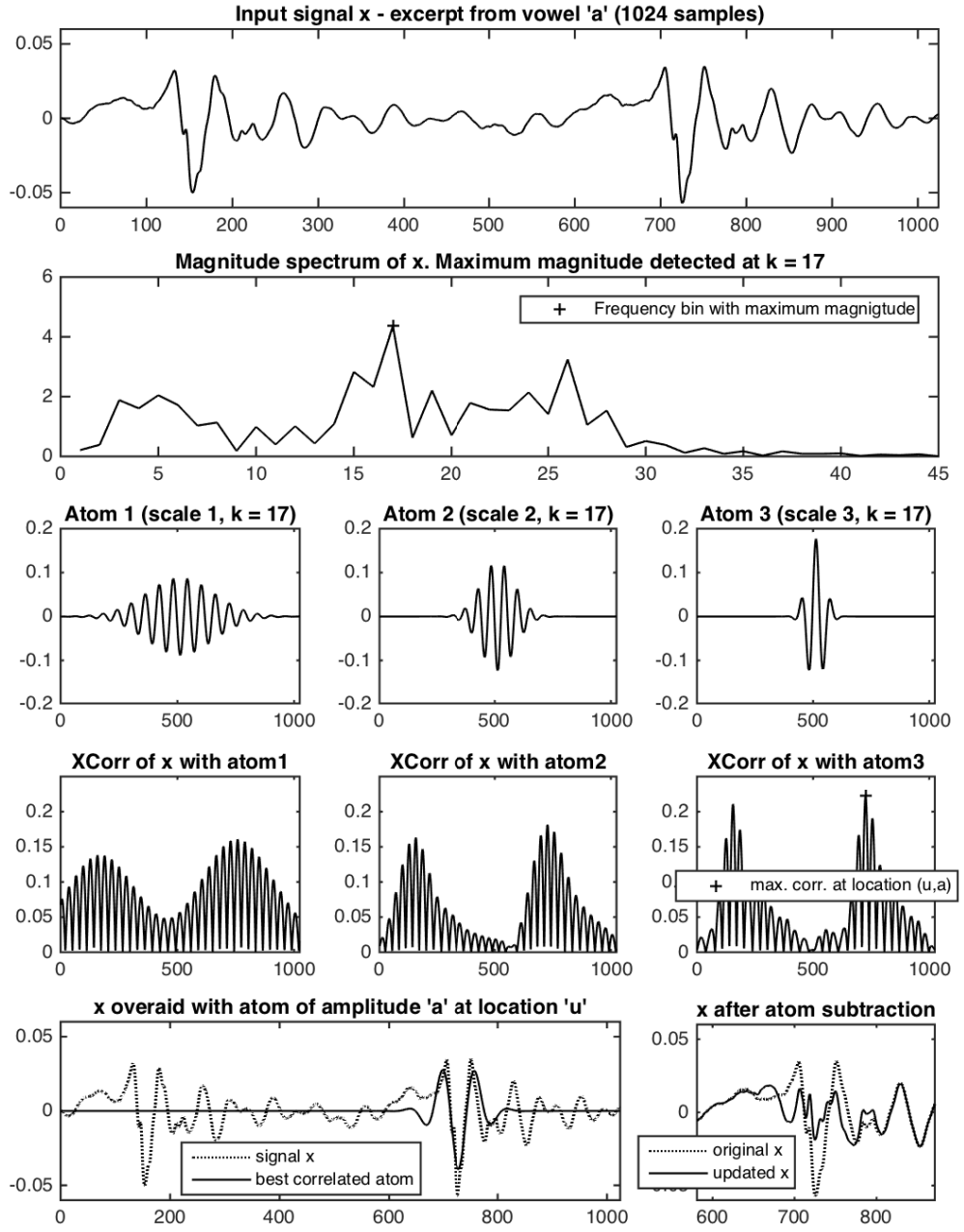


Figure 3.7: Demonstration of FFT pre-processing step. The first row depicts the input signal which is a 1024 sample long excerpt from a spoken vowel 'a'. The second row is the magnitude spectrum of the input signal with the maximum magnitude frequency bin noted by the plus symbol. Based on the extracted frequency a dictionary of three Gabor atoms with same frequency but different scales is created and these are depicted in the third row. The fourth row shows the correlations between each atom and the input signal. The maximum correlation occurs for the third atom. Finally the first plot of the fifth row shows the input signal overlaid with the best selected atom and the second plot the updated signal after atom subtraction.

3.4.2 STFT Processing

When dealing with long duration sequences the STFT can be used as a pre-processing step which leads to an LFMP approach to GMP. This pre-processing step is essentially the extension of the FFT pre-processing for long duration sequences. As already explained in section 3.3.2 this processing does not have to occur within the main body of the algorithm as was the case with the FFT. Computing the STFT of a long sequence at every iteration is computationally expensive so the solution is to compute the STFT of the signal once during initialisation, in order to create a 'guide map' that can be searched for maxima time-frequency points which first of all will guide the decomposition to regions of interest within the signal and secondly will dictate the creation of appropriate dictionaries. Then at each iteration and after the update steps, only the relevant region of the STFT 'map' will be updated. There are several issues with this approach that need to be addressed. First of all the search of the time-frequency point with maximum magnitude is in itself a computationally expensive operation. Searching the maximum of the STFT map at each iteration is wasteful so instead a search tree is created. A similar approach is employed in TMP and MPTK.

In short a search tree is a tree data structure (similar to the ones obtained by the discrete wavelet transform and wavelet packet decomposition) which represents a set of values and provides a fast way to locate specific values within that set. The field of computer science deals with the design of efficient search tree algorithms but this is beyond the scope of this dissertation. At this point optimisations are not of main concern. Having said that the introduction of a search tree in the GMP algorithm is in itself an optimisation but in this case it is required in order to achieve a realisable GMP implementation for long sequences. What is presented here is a simple search tree algorithm where the computation steps of the search tree are adapted from the TMP implementation [174].

In algorithm 7 the function 'CreateSearchTree' is responsible for initialising necessary variables and creating a two-level search tree by using function 'GetFrm' which returns the number of frames of the STFT of the signal and the start and end positions of each frame and finally function 'GetUpdRegion' calculates the start and end positions of the region in the residual that needs to be updated. A pictorial view of algorithm 7 is illustrated in figure 3.8.

In algorithm 7, step 13 computes a spectrogram of the residual $\mathbf{r}$ using a hanning window of length L, hop size H and FFT size of N_{FFT} and produces a complex output $\mathbf{S} \in \mathbf{C}^{K \times P}$ where K is the number of the frequency components and P the number of data frames. Step 14 of the algorithm produces the magnitude spectrum $\mathbf{M} \in \mathbf{R}^{K \times P}$ which is depicted in sub-plot $a)$ of figure 3.8. The first level of the search tree is computed in step 15 and consists of elements of maximum magnitude within each frame. The output of this step is $\mathbf{M}_{lv1} \in \mathbf{R}^{P}$ and is depicted in sub-plot $b)$. That step also keeps track of the corresponding frequency bin numbers. Step 16 of the algorithm computes the second level of the search tree which is depicted in sub-plot $d)$ of the figure. The output of that step is $\mathbf{M}_{lv2} \in \mathbf{R}^{P_2}$ with $P_2 \ll P$ and sub-plot $c)$ shows in detail what actually happens within that step; the first search level $\mathbf{M}_{lv1}$ is segmented into equally spaced P_2 segments and the maximum elements within each segment are grouped to produce the second search level $\mathbf{M}_{lv2}$. The process discussed so far can be performed once before the decomposition takes place.

Algorithm 7 STFT pre-processing

1: **function** CREATESEARCHTREE($\mathbf{r}$, L, H, N_{FFT}, N_a, u_m)
initialise variables
2: $N_r \coloneqq length(\mathbf{r})$ ▷ get length of residual in samples
3: $[P, \mathbf{P}_{idx}] \coloneqq getfrm(N_r, L, H)$ ▷ get number of frames P and indices $\mathbf{P}_{idx}$
4: $L_{o2} \coloneqq L/2$ ▷ half the window length
5: $U_r \coloneqq round(u_m * N_a)$ ▷ length of update region
6: $U_{ro2} \coloneqq U_r/2$ ▷ half the length of update region
7: $P_2 \coloneqq \lceil \sqrt{P} \rceil$ ▷ number of elements for 2^{nd} search level
8: $P_{pad} \coloneqq P_s^2 - P$ ▷ necessary padding length
allocate memory
9: $\mathbf{M}_{lv1} \coloneqq zeros(P, 1)$ ▷ allocate space for 1^{st} search level
10: $\mathbf{M}_{lv2} \coloneqq zeros(P_2, 1)$ ▷ allocate space for 2^{nd} search level
11: $\mathbf{M}_{pad} \coloneqq zeros(P_{pad}, 1)$ ▷ allocate space for extra padding
12: $\mathbf{K}_{lv1} \coloneqq zeros(P, 1)$ ▷ allocate space for frequency bins
compute spectrogram and search tree
13: $\mathbf{S} \coloneqq spectrogram(\mathbf{r}, hanning(L), H, N_{FFT})$ ▷ compute spectrogram of $\mathbf{r}$
14: $\mathbf{M} \coloneqq |\mathbf{S}|$ ▷ get magnitude spectrum
15: $[\mathbf{M}_{lv1}(:,1), \mathbf{K}_{lv1}(:,1)] \coloneqq max(\mathbf{M})$ ▷ 1^{st} level max. values and freq. bins
16: $\mathbf{M}_{lv2}(:,1) \coloneqq max(reshape([\mathbf{M}_{lv1}; \mathbf{M}_{pad}], P_2, P_2))^T$ ▷ 2^{nd} level max. values
17: **return** [$\mathbf{M}$, $\mathbf{M}_{lv1}$, $\mathbf{M}_{lv2}$, $\mathbf{K}_{lv1}$, $\mathbf{P}_{idx}$, P_2, L_{o2}, H, U_r, U_{ro2}]
18: **end function**

19: **function** GETUPDREGION($\mathbf{M}$, $\mathbf{M}_{lv1}$, $\mathbf{M}_{lv2}$, $\mathbf{K}_{lv1}$, $\mathbf{P}_{idx}$, P_2, L_{o2}, H, U_r, U_{ro2})
traverse the tree
20: $[m_2, lv2_{pos}] \coloneqq max(\mathbf{M}_{lv2})$ ▷ find max value and index in 2^{nd} search level
21: $\mathbf{rg} \coloneqq P_2 * (lv2_{pos} - 1) + (1 : P_2)$ ▷ get 1^{st} level corresponding region
22: $[m_1, m_{ofs}] \coloneqq max(\mathbf{M}_{lv1}(\mathbf{rg}))$ ▷ find max value and index in 1^{st} search level
23: $k_1 \coloneqq \mathbf{K}_{lv1}(\mathbf{rg}(m_{ofs}), 1)$ ▷ get freq. bin # of highest magnitude component
24: $lv1_{pos} \coloneqq (lv2_{pos} - 1) * P_2 + m_{ofs}$ ▷ position of max. value in the 1^{st} level
compute update region indices for time and time-frequency domains
25: $pos_s \coloneqq \lfloor (\mathbf{P}_{idx}(lv1_{pos}, 1) + L_{o2} + H + U_{ro2} - U_r)/H \rfloor$
26: $pos_e \coloneqq \lceil (\mathbf{P}_{idx}(lv1_{pos}, 1) + L_{o2} + H + U_{ro2})/H \rceil$
27: $r_s \coloneqq \mathbf{P}_{idx}(lv1_{pos}, 1) + L_{o2} - U_{ro2}$
28: $r_e \coloneqq r_s + U_r - 1$
29: **return** [r_s, r_e, pos_s, pos_e, $lv1_{pos}$, k_1, m_1]
30: **end function**

31: **function** GETFRM(N_r, L, H)
32: $P \coloneqq \lceil (N_r - L - H)/H \rceil$ ▷ get number of frames
33: $\mathbf{k} \coloneqq (0 : P - 1)$ ▷ create an index line.
34: $\mathbf{P}_{idx} \coloneqq zeros(P, 3)$ ▷ matrix for start, center and end pos. of each frm.
35: $\mathbf{P}_{idx}(:,1) \coloneqq \mathbf{k} * H + 1$ ▷ calculate start position of each frame
36: $\mathbf{P}_{idx}(:,2) \coloneqq \mathbf{P}_{idx}(:,1) + L - 1$ ▷ calculate end position of each frame
37: $\mathbf{P}_{idx}(:,3) \coloneqq \mathbf{P}_{idx}(:,1) + \lfloor L/2 \rfloor$ ▷ calculate middle position of each frame
38: **return** [P, $\mathbf{P}_{idx}$]
39: **end function**

Then at each iteration of the GMP algorithm the function 'GetUpdRegion' in algorithm 7 is used to compute the start and end indices of the region within the residual that corresponds to the maximum element in the search tree. In particular steps 20 to 24 traverse the search tree to find the frame location and frequency bin number of the component with the maximum magnitude

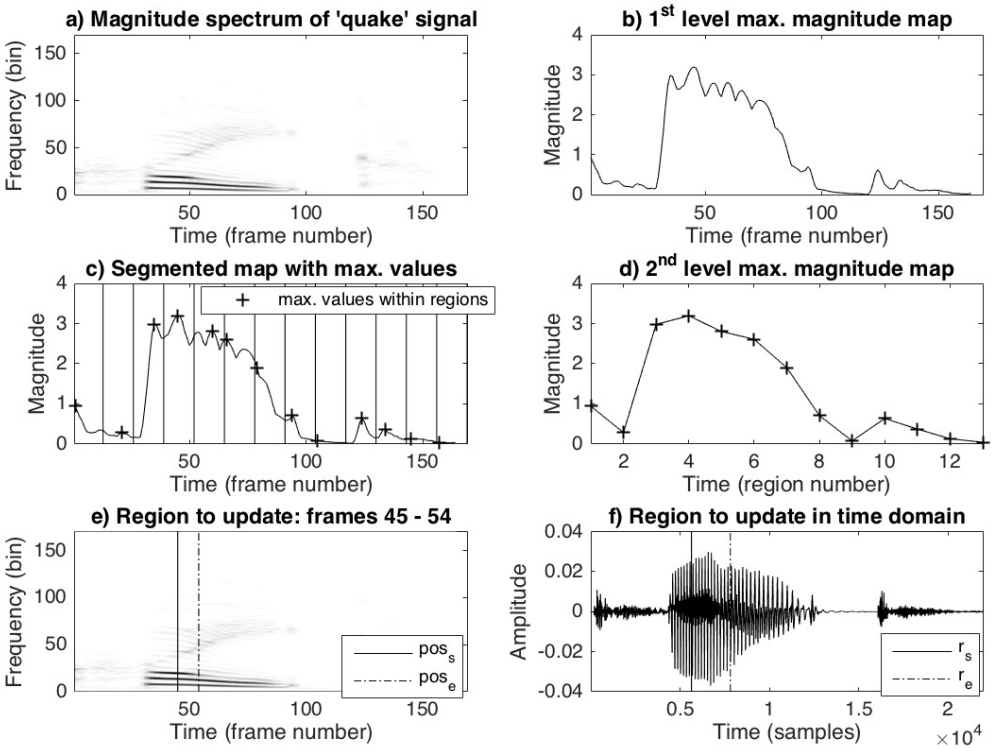


Figure 3.8: Demonstration of STFT pre-processing step.

and based on this information steps 25 to 28 compute the start and end positions of the region in the residual that need to be updated, with pos_s and pos_e indicating the update region in the time-frequency plane (sub-plot e)) and r_s and r_e indicating the update region in the time domain (sub-plot f)). Having obtained the coordinates $(k_1, lv1_{pos}, m_1)$ of the most important element in the time-frequency plane, the GMP algorithm continues in the way described in section 3.4.1.

Algorithm 7 presents only the most important functions of the STFT pre-processing step. A proper implementation must include extra logic to perform several other tasks like the update of the search tree after an atom extraction, boundary checks when calculating the update region positions and treatment of multi-channel signals. These sub-tasks are easy to implement since all the needed information in doing so are provided by algorithm 7.

3.4.3 QIFFT Processing

So far the main information that is extracted from the pre-processing steps discussed earlier, is the integer frequency bin number of the most prominent component in the spectrum of the residual, that is the component with the maximum magnitude. This frequency bin is then used to create a dictionary of atoms comprising that frequency and different scales which is then correlated with the residual. The frequency of the selected frequency bin though, does not correspond to the true frequency of the underlying sinusoid unless that sinusoid is periodic within the Fourier transformed data frame, so during GMP, in most cases the selected frequency bin is just a good enough estimate. A reasonable assumption to make is that given a better frequency estimate the resulting dictionaries will better correlate with the residual at each iteration thus leading to a better quality decomposition.

There are many ways to improve the frequency estimates obtained by the FFT or STFT including the phase vocoder, re-assigned spectrum techniques (which has many independent origins) or the chirp-z transform to name a few but for this work the quadratically interpolated FFT (QIFFT) was preferred for its relative simplicity, efficiency and accuracy. QIFFT was introduced in [107] as part of an analysis/synthesis system called 'PARSHL' and it has been extensively used in the field of spectral modelling for fast and accurate estimation of the modelled partials. using the QIFFT algorithm it is possible to get better estimates on many sinusoidal parameters including frequency and phase but amplitude modulation (AM) and frequency modulation (FM) as well. In the context of GMP, AM and FM estimation could be useful in determining chirp-rates for the creation of chirp-atoms and amplitude envelope modulation for determining appropriate window functions for the atoms. The developed software provided with this dissertation allows FM estimation using the QIFFT method and the tools for experimenting with chirp-like atom types are in place. Exploration into this area could form part of future research. The QIFFT algorithm for parameter estimation from peaks of a given magnitude spectrum can be performed as illustarted in algorithm 8 (adapted from [183]).

Algorithm 8 QIFFT pre-processing

1: **function** $\mathrm{QIFFT}(\mathbf{x},\, Z)$
The following processing takes place on a data frame as supplied by an STFT
2: $L \coloneqq length(\mathbf{x})$ ▷ get length of input data frame
3: $N_{FFT} \coloneqq 2^{nextpow2(Z*L)}$ ▷ N_{FFT} is a power of two larger than $Z * L$
4: $\mathbf{x} \coloneqq \mathbf{x} \odot window(L)$ ▷ modulate input by a window
5: $\mathbf{S} \coloneqq FFT(\mathbf{x}, N_{FFT})$ ▷ compute FFT of a zero-padded windowed data frame
6: $\mathbf{M} \coloneqq 20log_{10}(|\mathbf{S}|)$ ▷ get squared magnitude spectrum in dB
7: $\mathbf{PHI} \coloneqq \angle\mathbf{S}$ ▷ get phase spectrum if required
8: $[m_0, k_0] \coloneqq max(\mathbf{M})$ ▷ find value and location of maximum peak magnitude
▷ get magnitude values of the neighbouring bins
9: $k_{-1} \coloneqq k_0 - 1,\, m_{-1} \coloneqq \mathbf{M}(k_{-1})$ ▷ neighbour on the left of k_0
10: $k_1 \coloneqq k_0 + 1,\, m_1 \coloneqq \mathbf{M}(k_1)$ ▷ neighbour on the right of k_0
11: $p \coloneqq \frac{1}{2}\frac{m_{-1}-m_1}{m_{-1}-2m_0+m_1}$ ▷ quadratically interpolate using neighbouring samples
12: $\hat{k_0} \coloneqq k_0 + p$ ▷ get true peak location in frequency bins
13: $\hat{m_0} \coloneqq m_0 - \frac{1}{4}(m_{-1} - m_1)p$ ▷ get new magnitude estimate
14: Use step 13 on both real and imaginary parts of the complex spectrum to get a complex-valued peak estimate if required (both magnitude and phase).
15: **return** $[\hat{k_0},\, \hat{m_0}]$
16: **end function**

There are certain aspects of algorithm 8 that are not explicitly stated. First of all the choice of the window function in step 4 is important since it can influence the correct detection of peaks, among other things. The choice of the analysis window in STFT based applications is a huge subject and well beyond the scope of this dissertation but it is important to note that modulating an input signal with a window function in the time domain translates to convolution of their spectra in the frequency domain and as such the shape of the window function directly affects the output of the FFT operation in step 5. When dealing with windows we are mostly interested by two main characteristics of the window's spectrum; namely the main lobe width and the main-lobe to highest side-lobe ratio, or simply the main-lobe to side-lobe ratio. The best window would be one where its spectrum has the smallest main lobe width and highest main-lobe to side-lobe ratio but there is always a trade-off between these two characteristics and depending on the application usually one characteristic is preferred over the other.

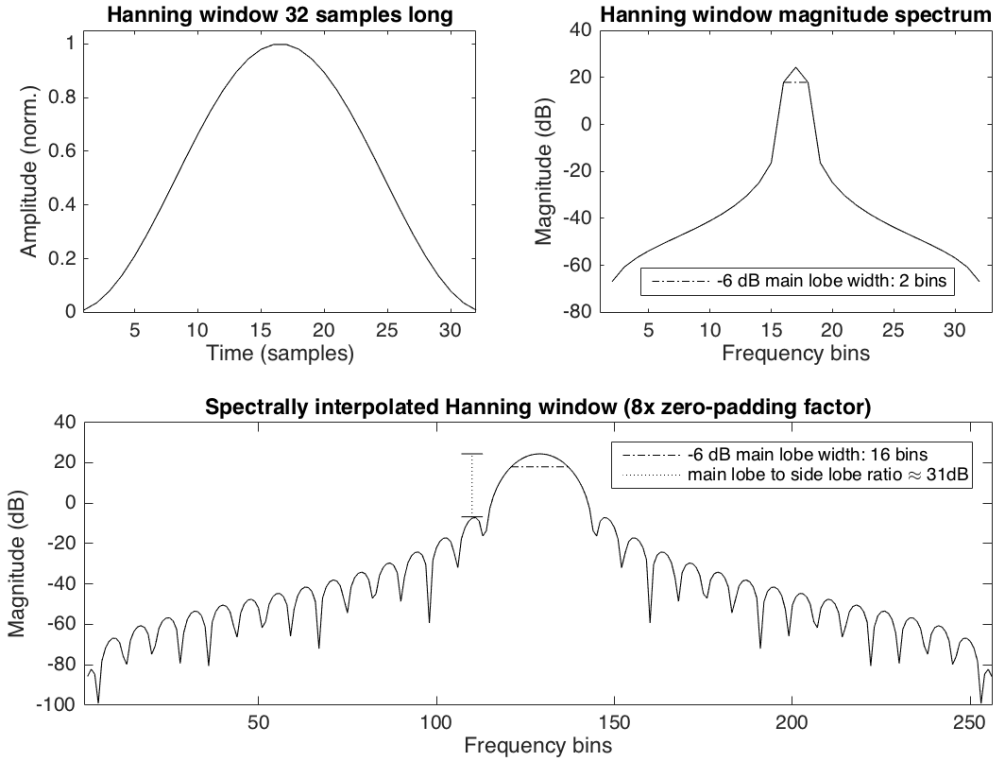


Figure 3.9: Demonstration of Hanning window. The top left sub-plot depicts a 32 samples long Hanning window in the time domain and the top right sub-plot its magnitude spectrum where the main lobe is visible and has a width of 2 frequency bins. The bottom sub-plot is an interpolated spectrum with an 8x zero-padding factor which brings the size of the FFT to $N_{FFT} = 8 \times 32 = 256$. Zero-padding in the time-domain leads to spectral interpolation, that is the spectrum is sampled more densely and the width of a main lobe of K bins in the original spectrum becomes $\frac{KN_{FFT}}{L}$ bins in the interpolated spectrum, where L is the length of the window. In this case the main lobe width becomes $2 \cdot 8 = 16$ samples long. Finally the main-lobe to side-lobe ratio is also visible and is about 31dB.

These window characteristics are illustrated in figure 3.9. The top left sub-plot depicts a 32 samples long Hanning window in the time domain with its spectrum shown in the top right sub-plot. In the spectrum the main-lobe can be seen and is 2 bins in width. Conventionally, the main-lobe width is defined as the width at -6 dB below the main lobe peak. The bottom sub-plot depicts an interpolated window spectrum which was achieved by zero-padding the window in the time domain. Zero-padding in the time-domain leads to interpolation in the frequency domain which means that the spectrum is more densely sampled and the zero-padding factor is calculated by:

$$Z = \frac{N_{FFT}}{L} \tag{3.17}$$

where N_{FFT} is the size of the FFT operation and L the size of the window or any input data sequence. In the Hanning window example the zero-padding factor is $8\times$ which makes the size of the FFT $N_{FFT} = 8 \cdot 32 = 256$ samples long. Since zero-padding in the time domain introduces more samples in the frequency domain the main lobe width of the window spectrum will be different and the following relationship holds [107]:

$$K_{zp} = K\frac{N_{FFT}}{L} \tag{3.18}$$

where K is the main-lobe width when the zero-padding factor is 1.

In this particular example the main-lobe width of the interpolated spectrum is $2\frac{256}{32} = 16$ bins. Finally in the bottom sub-plot the main-lobe to side-lobe difference is visible and for the Hanning window is approximately 31dB. It should be clear that an interpolated spectrum is much easier to read and for the purpose of quadratic interpolation the spectral peaks are much easier to find. These are the main reasons behind the first five steps of algorithm 8. For discussions on several commonly used windows the reader is referred to [184, 185] and for a detailed explanation on reasons behind choosing the analysis window and other parameters in the QIFFT technique to [107].

Continuing with the algorithm in step 6 the squared magnitude in dB is used instead of linear magnitude and this choice stems from empirical observations of the authors in [107] who claim that the frequency estimates are more accurate when dB magnitude is used. These claims were partially confirmed with informal testing of QIFFT within the GMP framework. Finally steps 8 to 11 perform the quadratic interpolation and steps 12 and 13 produce the new estimated peak location and corresponding magnitude. The equation in step 11 can be derived by the equation of a parabola of the following form [107]:

$$y(x) \stackrel{\text{def}}{=} a(x-p)^2 + b \tag{3.19}$$

The result of these steps is illustrated in figure 3.10.

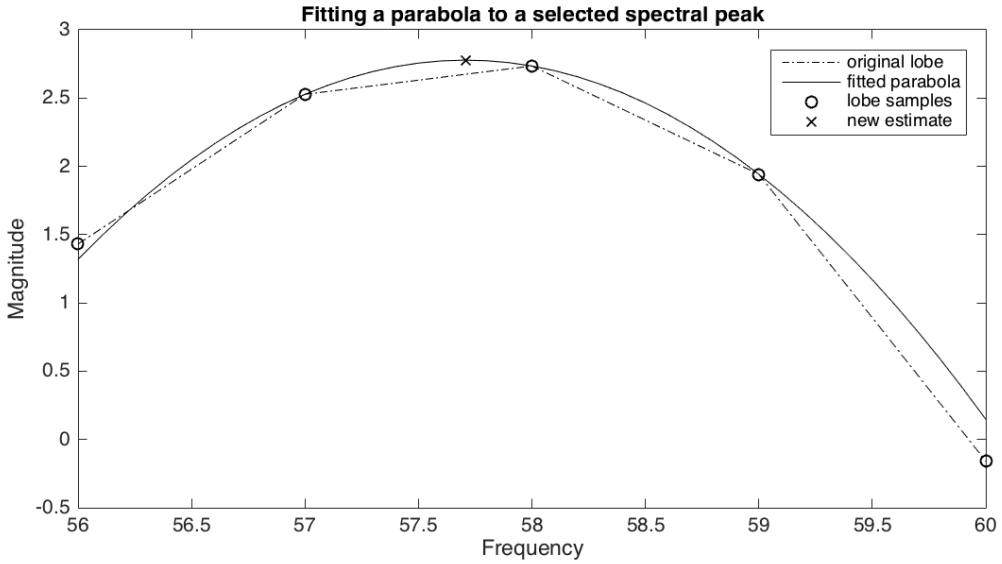


Figure 3.10: Fitting parabola to a selected spectral peak.

Dictionary Bias Demonstration

One good way to demonstrate the effect of the QIFFT pre-processing step in GMP decomposition is to consider the dictionary bias problem mentioned in section 2.3.6. To recall, the authors in [153] investigate the decomposition bias introduced by the use of a discretised dictionary and claim that this bias becomes statistically significant when a large number of decompositions is considered, as for example, in the analysis of EEG signals. A discretised dictionary is obtained by discretising its parameter space as for example suggested in [68] and [64], in contrast to a non-discretised dictionary where its parameter space is continuous.

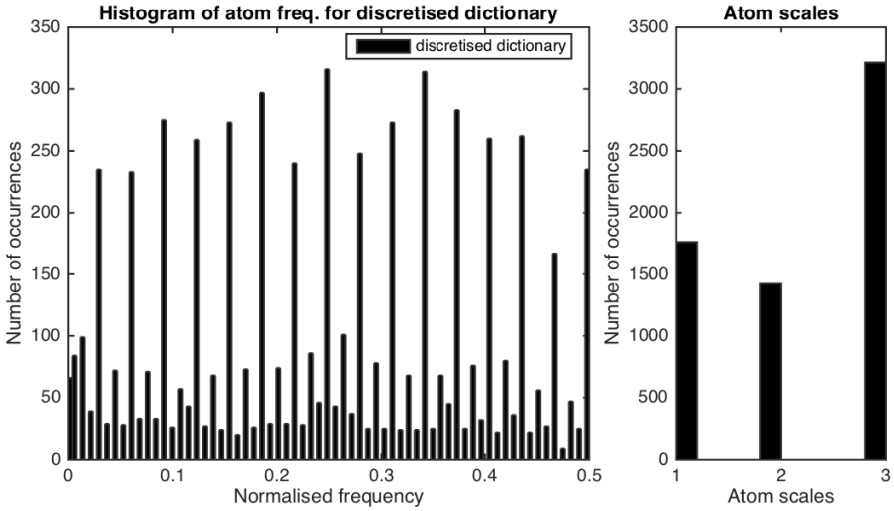


Figure 3.11: GMP dictionary bias - Histogram of selected atom frequencies when using a discretised dictionary.

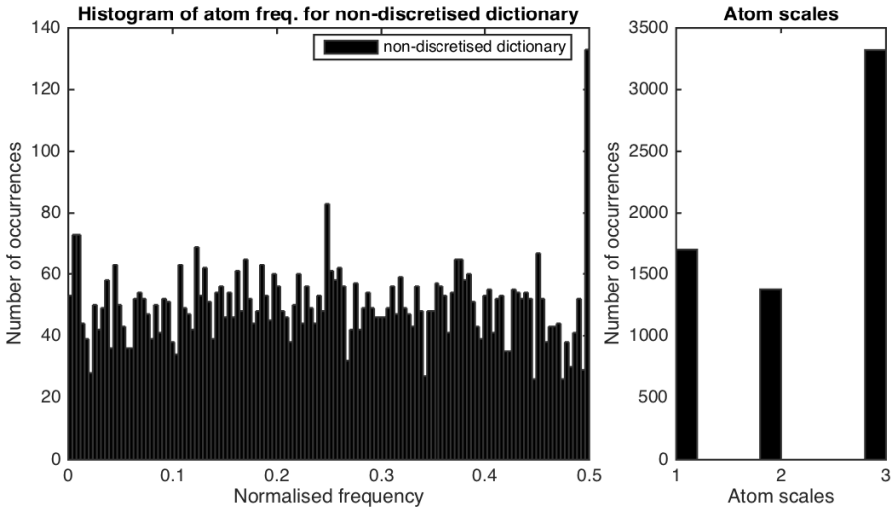


Figure 3.12: GMP dictionary bias - Histogram of selected atom frequencies when using a non-discretised dictionary.

To demonstrate the bias introduced by such structured dictionaries one hundred realisations of white Gaussian noise are decomposed using the common dyadic scale Gabor dictionary and the histogram of frequencies of the extracted atoms is displayed. The results of this experiment using a GMP decomposition with an FFT pre-processing step and a three dyadic scale Gabor dictionary are displayed on figure 3.11. The left sub-plot of that figure depicts the histogram of the frequencies of all extracted atoms and the right sub-plot depicts the histogram of the scales of the atoms. Note that the scale numbers are inversely proportional to atom length with scale one denoting the largest atom length. This figure clearly shows the dictionary bias; the frequencies of the extracted atoms are drawn from a discrete subset of all available frequencies. An interesting observation is that this frequency selectivity is directly related to the scales of the selected atoms which was expected and is a direct consequence of the structure of the dictionary. The same experiment was performed by using the QIFFT pre-processing step and the results are displayed on figure 3.12. This figure clearly shows that the dictionary bias has disappeared with the frequencies of the selected atoms being chosen from the complete frequency space regardless the scale of the atoms.

A similar effect can be demonstrated using the MPTK. As already mentioned the MPTK implementation makes use of the STFT to create Gabor 'blocks' which are equivalent to a single scale Gabor dictionary. Multiple such blocks can be used to facilitate a multi-scale decomposition. In MPTK a Gabor 'block' is defined within an .xml file as follows:

```
<block uses="GAUSS-WINDOW">
    <param name="type" value="gabor"/>
    <param name="windowLen" value="128"/>
    <param name="windowShift" value="16"/>
    <param name="fftSize" value="128"/>
</block>
```

These parameters reflect the parameters of an STFT where 'windowLen' is the length of the analysis window, 'windowShift' is the hop size and 'fftSize' is the size of the FFT to be performed. The same experiment was performed using three such Gabor 'blocks' (i.e. a three scale Gabor dictionary). For comparison purposes the parameters used in both the GMP and MPTK decompositions were as similar as possible. From a high level perspective the parameters are exactly the same, that is a three scale Gabor dictionary is used in all experiments. Implementation wise though, the effect of these parameters to the results of the decompositions is unknown since the implementations are different and although certain parameters might suggest the same functionality, the internal implementation of those is most probably different. Nonetheless the results obtained are similar in nature. Figure 3.13 displays the results of the white noise experiment using the MPTK implementation and a three scale Gabor dictionary. Again the dictionary bias is visible as expected, although in this case the distribution of the scales of the selected atoms is different compared to the GMP implementation. It seems that in this case, MPTK prefers small scale atoms whereas in GMP the scales are distributed in a more uniform fashion. The exact reason for this behaviour is unknown but is certainly related to the way dictionaries are implemented in each algorithm.

Letting the FFT size parameter of MPTK's Gabor 'block' definition be larger than the window length, translates to performing an STFT where each data frame is zero padded with a zero-padding factor of $fftSize/windowLen$ which leads to spectral interpolation as explained earlier. Translating the active frequency bins of the interpolated spectrum to their original scale leads to non-integer frequency bin values which affect the creation of the atoms. For example assume that a $N = 128$ length Gabor atom is created with a normalised frequency $f = k/N$ with $k = 11.7$. Now computing the magnitude spectrum of that atom with $N_{FFT_1} = 128$ will produce a single peak at frequency bin $k_1 = 12$. Computing the magnitude spectrum with $N_{FFT_2} = 1024$ (i.e. $8\times$ zero-padding factor) will produce a peak occurring at frequency bin $k_2 = 94$. Downscaling that result to the corresponding frequency bin of the original sized FFT will yield $\hat{k} = k_2 \frac{N_{FFT_1}}{N_{FFT_2}} = 94 \frac{128}{1024} = 11.75$ which is closer to the initial frequency bin and a non-integer value. The results produced by allowing the 'fftSize' parameter be larger than the 'windowLen' parameter are presented in figure 3.14 where again it can be seen that the initial dictionary bias has disappeared. An observation regarding the frequency histogram is that there is a dip in the distribution of frequencies close to both edges of the frequency axis and two big spikes at the extremes of either edge.

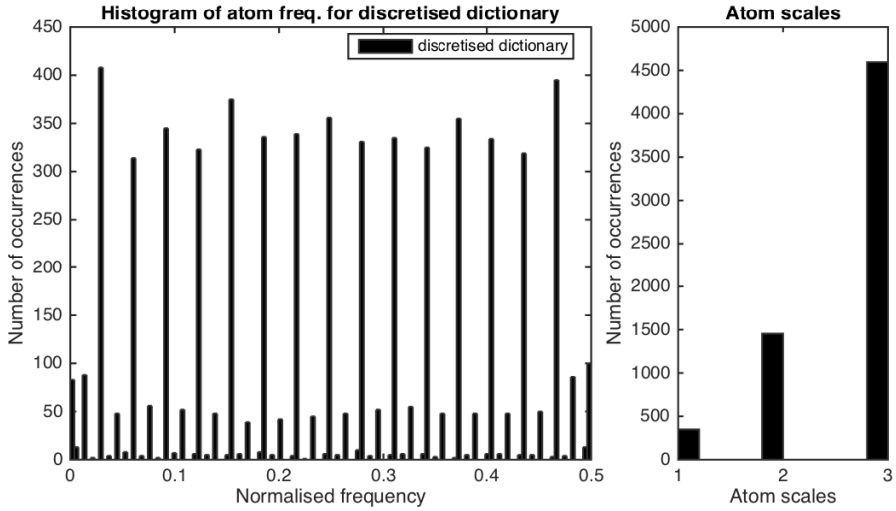


Figure 3.13: MPTK dictionary bias - Histogram of selected atom frequencies when using a discretised dictionary.

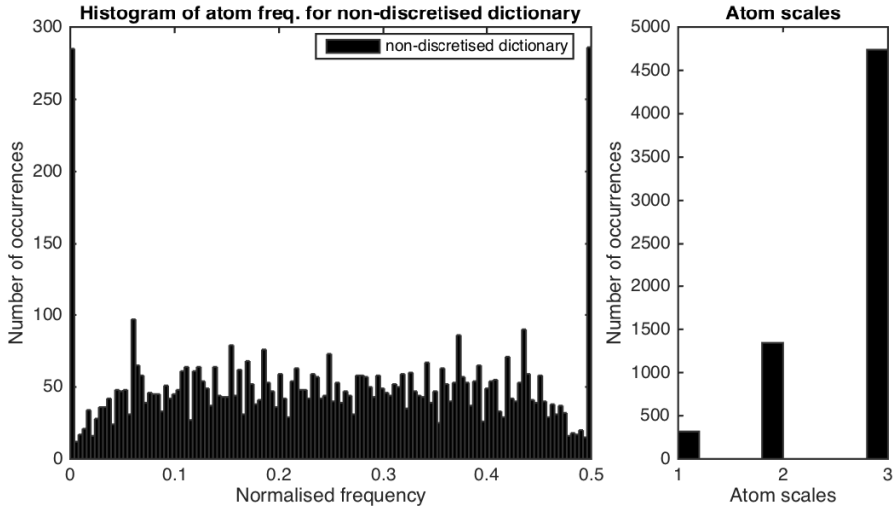


Figure 3.14: MPTK dictionary bias - Histogram of selected atom frequencies when using a non-discretised dictionary.

Again the reason for this behaviour is not known but it is assumed that it is probably due to the structure and also the type of the dictionary. Informal experimentation with GMP has shown that different atom types produce different frequency distributions in the white noise decomposition experiment and this has led the author to believe that the type of atoms used in the MPTK play a role in this behaviour. Unfortunately running those experiments with different atom types in MPTK was not possible because the current MPTK implementation was producing errors when windows other than the Gaussian one were used in the dictionary definition file.

3.4.4 Other Potentially Useful Processing

So far only FFT based algorithms have been suggested as the pre-processing step, but GMP is certainly not limited to those. The main idea is to use any kind of processing that is able to provide useful information about the signal being analysed and use the inferred parameters to construct dynamic dictionaries that will correlate well with the signal. In this context pretty much any of the techniques described in chapter 2 could be used as a pre-processing step.

The following list suggests some potentially useful algorithms, techniques and a priori information that could be adapted to work with GMP (in no particular order):

- Phase vocoder
- Spectrum reassignment
- Chirp Z-Transform
- Discrete wavelet transform
- Wavelet packet decomposition
- MDCT
- Spectral modelling
- Source-filter processing (e.g. Channel vocoder, LPC)
- Formant detection and tracking
- AM/FM rate estimation
- Low level feature extraction (e.g. sound energy, RMS amplitude, spectral tilt, spectral centroid, note onset/offset)
- High level feature extraction (e.g. multi-pitch, fundamental frequency, vibrato, harmonicity)
- Spatial cues (source azimuth estimation by using IID and ITD, elevation, distance)
- Perceptual modelling
- Residual and transient analysis
- Beat detection and tracking
- Mixing matrix estimation
- Source separation
- Musical score estimation/information
- Musical instrument identification/information
- Number of sources within a mixture

Some of the techniques mentioned are generic and include other items in the list as subtasks. Although this is not an exhaustive list it does show the plethora of information that can be obtained by pre-analysis of the signal. The software package provided with this dissertation only implements the three pre-processing algorithms explained in sections 3.4.1 to 3.4.3 but the main code infrastructure is in place and it should be relatively easy to adapt and use many of the techniques in this list.

3.5 Multi-Channel GMP

Another MP implementation issue that is not widely discussed in the literature is the treatment of multi-channel signals. The MP variants presented in section 2.4 suggest several approaches to this problem and indeed there are multiple ways to extend MP into a multi-channel setting, with each producing significantly different results. Which approach to use depends on the application at hand and also the mixing model of the analysed signal. As correctly stated in [186][4] this variety of choice is mostly due to:

- The structure of the multi-channel dictionary.
- The atom selection criterion.

Regarding the GMP algorithm and specifically the pre-processing steps, all discussions and examples discussed so far are based on single-channel signals but implementation wise, extending the algorithm for multi-channel processing is straightforward and can be achieved by just adding an extra dimension in all variables, vectors and matrices involved. The following sub-sections present and categorise a few possible solutions to the treatment of multi-channel signals.

3.5.1 Dis-Joint Multi-Channel Processing

Probably the most straightforward approach to multi-channel processing is to decompose each signal channel separately which leads to the category of dis-joint multi-channel processing techniques. In the context of source separation this approach was used in [166] where each channel of a stereo signal is decomposed separately and then based on the IID mixing assumption, atoms that occupy the same time-frequency space are used to estimate directions of arrival. A problem with this approach is that in practice the atoms between the channels rarely coincide in both time and frequency. Informal experimentation supports this statement and also this issue is implicitly acknowledged in [166] where the authors propose the use of trained dictionaries with the aim of increasing the correlation of atoms between the channels thus improving the quality of separated sources.

3.5.2 Joint Multi-Channel Processing

Another approach in the treatment of multi-channel signals is to process all channels simultaneously thus leading to the category of joint multi-channel processing techniques. Probably the first application of MP in stereo signals, at least in the field of source separation, is the stereo matching pursuit (SMP) in [162] where the author assumes that only the amplitude is varied across channels and uses a stereo dictionary to minimise the energy of the residual across both channels, or using another equivalent statement, to maximise the sum of squared dot products (i.e. energy) across both channels of the approximation (i.e. equation 2.161) which can be compactly expressed as:

$$\max_{d_\gamma \in \mathcal{D}} \sum_{j=1}^{J} |c_j|^2 \tag{3.20}$$

[4] http://www.scholarpedia.org/article/Matching_pursuit

where J is the total number of channels within a signal and c_j the expansion coefficients at each channel as given by equation 2.133.

Another sub-optimal solution proposed in [187] and suited to EEG signal analysis is the maximisation of the sum of products expressed as:

$$\max_{d_\gamma \in \mathcal{D}} \left| \sum_{j=1}^{J} c_j \right| \tag{3.21}$$

In GMP, the joint processing of channels occurs via the use of 'guide maps' extracted during the pre-processing steps. In particular, when any of the FFT-based pre-processing steps described in section 3.4 is used, the energy of the spectra of the channels are added together to create a single 'guide map' that will force the algorithm to operate at a particular support of the signal. How atoms are extracted from this location depends on the processing task that takes place. For a decomposition task, atoms are extracted so that they minimise the residual energy across both channels, at that particular location similarly to SMP. The difference with SMP is that in GMP the atom extraction location can be obtained by factors other than energy, so the 'greedy' criterion of standard matching pursuit does not always hold. Even when the atom's extraction location is found by obtaining the location of the maximum magnitude element from the guide map, this does not guarantee that the absorption of the residual energy after atom extraction will be maximal. For example when the task of separating sources in linearly mixed stereo signals is considered, the main approach in GMP is to first obtain the location of the maximum element in the guide map and then extract a couple of atoms that obey the IID mixing assumption and have the same frequency and zero phase difference, that is they occur at the same time location within the signal. Given this situation, it is clear that equation (3.20) is not necessarily obeyed even when additional energy constraints are imposed to the atom selection criterion.

3.5.3 Weakly Joint Multi-Channel Processing

Another possible solution, especially in the context of GMP, could be a weakly joint or constrained dis-joint multi-channel processing approach. For example each signal channel is decomposed separately but at each iteration the algorithm is constrained to operate at the same support of the signal. In this case, a time constraint is enforced but the processing is still termed as dis-joint since the extracted atoms can vary both in frequency and location (within the limits imposed by the time constraint). This approach has not been extensively tested and the benefits, if any, are unknown.

3.6 Atom Generation in GMP

The subject of time-frequency dictionaries in sparse decompositions was introduced in section 2.2.3 where two main families of dictionaries were described, namely analytic and trained dictionaries, and in section 2.3.2 where the family of Gabor atoms along with the most common atom properties and concepts of the parameter index γ and the parameter set Γ were introduced. GMP makes heavy use of analytic dictionaries and reasons for preferring analytic over trained dictionaries

were also given in section 2.2.3. This section describes several atom families and also several implementation aspects are discussed.

When dealing with real signals in order to get a decomposition with real expansion coefficients a dictionary of real time-frequency atoms must be used. Also when signals are discrete (or more precisely digital) then discrete time (and amplitude) atoms must be considered. The atoms of a dictionary must be square integrable which means they must have finite energy and typically they must have unit norm. The unit norm property is not a strict requirement and some MP implementations, such as the ones found in [148] and [112] for example, create dictionaries where the atoms have different weightings. When all atoms of a dictionary have unit norm it essentially means that during the inner-product operations of MP all atoms will have the same probability of being selected or, in other words, they are treated equally.

As already explained, GMP makes use of analytic atoms and the main reason is that such atoms can be expressed analytically and are described by a set of parameters which can be easily stored and modified. Studying the expressions of several different types of atoms found in the literature one can observe a pattern in the way they are created. Essentially all atoms are relatively simple waveforms multiplied by window functions, so any type of atom can be expressed as:

$$d_\gamma[n] \coloneqq K_\gamma x_{\gamma_x}[n] w_{\gamma_w}[n], \quad n = 0, \ldots, N_d \tag{3.22}$$

where $x_{\gamma_x}[n]$ is an elementary waveform characterised by the parameter index γ_x, $w_{\gamma_w}[n]$ is a window function characterised by γ_w, $d_\gamma[n]$ is the produced atom with $\gamma = \gamma_x \cup \gamma_w$, K_γ is a normalisation constant so that the atom has unit norm, i.e. $\|d_\gamma[n]\| = 1$ and N_d is the length of the atom in samples. Equation (3.22) is used to describe a single, static in time atom, without explicitly stating its translation in time as in equation 2.139 and facilitates a direct software implementation. As such the assignment operator '$\coloneqq$' is used to emphasize this direct implementation. Of course when considering families of atoms, the translation parameter (among others) is included in the parameter space but it was excluded from the above expression because it is implementation specific. Depending on the choice of x and w different atom types can be obtained the name of which could be determined from the properties of x or w or both. In the following sub-sections several common, and some not so common atom types are described and defined.

3.6.1 Sinusoidal Atoms

Many atom types are based on the well known sinusoidal waveforms, which are based on the sine and cosine functions. These waveforms are also called pure tones and are characterised by an amplitude, single frequency and phase. In particular a sine based waveform can be defined as:

$$x_{sin}(N_d, k, \phi) \stackrel{\text{def}}{=} x[n] \coloneqq \sin\left(\omega_k t[n] + \phi\right), \quad n = 0, \ldots, N_d - 1 \tag{3.23}$$

and a cosine based waveform can be defined as:

$$x_{cos}(N_d, k, \phi) \stackrel{\text{def}}{=} x[n] \coloneqq \cos\left(\omega_k t[n] + \phi\right), \quad n = 0, \ldots, N_d - 1 \tag{3.24}$$

where N_d is the length of the waveform in samples, ω_k is the normalised angular frequency, henceforth Fourier frequency or simply normalised frequency, $t[n]$ is a time vector and ϕ is the phase. The normalised frequency is expressed as:

$$\omega_k \coloneqq 2\pi \frac{k}{N_d}, \quad k = 0, \ldots, N_d/2 \tag{3.25}$$

where k is the frequency index and N_d is the length of the atom in samples. The time vector $t[n]$ can be computed in a couple of ways depending on the desired properties of the output. Let us define the following time operator:

$$t_q[n] \stackrel{\text{def}}{=} t[n] \coloneqq \begin{cases} n, & n = 0, \ldots, N_d - 1, \quad q = \{0, \text{'periodic'}\} \\ n\frac{N_d}{N_d - 1}, & n = 0, \ldots, N_d - 1, \quad q = \{1, \text{'symmetric'}\} \end{cases} \tag{3.26}$$

where for $q = 0$ the time vector will produce a periodic waveform and for $q = 1$ a symmetric waveform. In GMP, parameter q is included in the parameter space of all atoms but for the rest of this section and for purposes of clarity the notation will be dropped and the time vector will be denoted simply by $t[n]$. It should be noted though these time vectors produce slightly different atoms and the result of a decomposition is affected by this choice although not in a clear manner. This issue requires further study.

In order to obtain sinusoidal atoms the sinusoidal waveforms defined in equations (3.23) and (3.24) have to modulate a window function. The window function can be any window function of interest and depending on the chosen window function various types of atoms are produced. First let us define a generalised sinusoidal function operator:

$$x_v(N_d, k, \phi) \stackrel{\text{def}}{=} x_v[n] \coloneqq \begin{cases} x_{sin}(N_d, k, \phi), & v = \{0, \text{'sin'}\} \\ x_{cos}(N_d, k, \phi), & v = \{1, \text{'cos'}\} \end{cases} \tag{3.27}$$

Then the following definition can describe generalised sinusoidal atoms:

$$d_{sinusoid}(N_d, k, \phi, v) \stackrel{\text{def}}{=} d[n] \coloneqq x_v[n] w[n] \tag{3.28}$$

where w can be any window function.

3.6.2 Fourier Atoms

Fourier atoms are produced when a rectangular window function is modulated by sinusoidal atoms. The rectangular window function can be defined as:

$$w_{rect}(N_d) \stackrel{\text{def}}{=} w[n] \coloneqq 1, \quad n = 0, \ldots, N_d - 1 \tag{3.29}$$

When considering a Fourier dictionary, that is the family of sine and cosine waveforms, then the phase parameter in equations (3.23) and (3.24) is set to zero and the range of the frequency index is $k = 0, \ldots, N_d/2$ for all cosines and $k = 1, \ldots, N_d/2 - 1$ for all sines, thus the full dictionary consists of N_d waveforms and forms a basis [64].

Fourier atoms can be defined as:

$$d_{fourier}(N_d, k, \phi = 0, v) \stackrel{\text{def}}{=} d[n] \coloneqq x_v[n] w_{rect}[n] \tag{3.30}$$

3.6.3 Gabor Atoms

Gabor atoms are produced by modulating a Gaussian window function with sinusoidal atoms. The name is attributed to Dennis Gabor who was the first one to use them as explained in section 2.1.5. There are various ways to express a Gaussian window function but the following definition produces a normalised Gaussian window with an adjustable variance parameter [150]:

$$w_{gauss}(N_d, s) \stackrel{\text{def}}{=} w[n] \coloneqq \pi a_s^{-\frac{1}{4}} \exp\left(-\frac{t[n]^2}{2a_s}\right), \quad n = -N_d/2 + 1, \ldots, N_d/2 \tag{3.31}$$

$$a_s = \left(\frac{4}{\pi}\right) 2^{2(s_0 - s)}, \quad s = 0, \ldots, s_0 \tag{3.32}$$

where N_d is the maximum allowable length of the Gaussian window and its width can be altered by the variance a_s, where s denotes the scale with scale 0 producing the largest variance and s_0 the smallest variance and the choice of s_0 depends on the application. A Gabor atom with variable scale can then be defined as:

$$d_{gabor}(N_d, s, k, \phi, v) \stackrel{\text{def}}{=} d[n] \coloneqq x_v[n] w_{gauss}[n] \tag{3.33}$$

where all symbols have their usual meaning.

3.6.4 Gammatone Atoms

Gammatone atoms are produced by modulating a gamma distribution with sinusoidal atoms. In literature they are known as gammatone filters and have been used extensively in the study of auditory filter modelling. As such the objectives of gammatone filters and gammatone atoms can be considered different since when dealing with a signal decomposition we are not limited to using the standard gammatone filter models which are mainly tailored to replicate the human auditory filter response. The literature on gammatone filters is vast with many different designs being available.

A small introduction and historic review of auditory filter models can be found in [188] and discussions on gammatone filter design and applications in [189] and [190] for example. In GMP of particular interest is the shape of the gamma distribution which can be used to characterise the amplitude evolution of many sounds of interest. A gamma distribution function can be defined as:

$$w_{gamma}(N_d, a, p, \beta) \stackrel{\text{def}}{=} w[n] \coloneqq at[n]^{p-1} \exp\left(-\beta t[n]\right), \quad n = 0, \ldots, N_d - 1 \tag{3.34}$$

where a is an amplitude factor, p is the shape parameter and β the rate (or scale). When dealing with gammatone filters, the shape parameter corresponds to the order of the filter and the rate parameter is defined so as to express the bandwidth of the filter:

$$\beta = 2\pi b ERB(f_c) \tag{3.35}$$

$$ERB(f_c) = 24.7 + 0.108 f_c \tag{3.36}$$

where ERB is the equivalent rectangular bandwidth (ERB) of the auditory filter with f_c the center frequency of the filter and equation (3.36) was defined in [191] for moderate sound pressure levels. In this context β is the damping factor and parameter b defines the proportion of the ERB. Equations (3.35) and (3.36) are expressed in terms of frequency in Hz and when considering a GMP implementation this is a bit limiting in a sense that the sampling rate of the signal becomes an extra parameter in the parameter space of that atom. In order to avoid this issue the rate parameter β has to be expressed differently. One convenient way to express β is in terms of the scale parameter p:

$$\beta = 2\pi(log_2(p) + b), \quad p > 0, \quad b > 0 \tag{3.37}$$

with b influencing the damping of the distribution. Equation (3.37) was defined empirically. Finally a gammatone atom can be defined by:

$$d_{gammatone}(N_d, k, \phi, v, p, b) \stackrel{\text{def}}{=} d[n] \coloneqq x_v[n] w_{gamma}[n] \tag{3.38}$$

3.6.5 Chirp Atoms

Another interesting set of elementary waveforms that can be used to produce various atom types are the so called chirp signals or chirps for short. A chirp signal, also known as a sweep signal, is characterised by a frequency that evolves over time either in an increasing fashion, ('up-chirp' signal) or a decreasing fashion ('down-chirp' signal). This is in contrast to pure tones where their frequency remains constant over time. Chirp signals occur naturally or are synthesised and are ideal for capturing frequency sweeps within a signal, something that pure tones are not able to do. Depending on the way the instantaneous frequency changes within a chirp signal we can distinguish between three main types of chirps, namely linear, quadratic and logarithmic chirps [192]. Before defining those types let us define a trigonometric function operator:

$$\text{trigf}_v(h) \stackrel{\text{def}}{=} \begin{cases} \sin(h), & v = \{0, \text{'sin'}\} \\ \cos(h), & v = \{1, \text{'cos'}\} \end{cases} \tag{3.39}$$

Also let $f_0 = k_0/N_d$ and $f_1 = k_1/N_d$ be the normalised instantaneous frequencies at time sample 0 and N_d respectively.

A linear chirp can be defined as:

$$g_{lincrp}(N_d, f_0, f_1, \phi, v) \stackrel{\text{def}}{=} g[n] \coloneqq \text{trigf}_v \left(2\pi \left[f_0\, t[n] + \frac{\beta}{2}\, t^2[n] \right] + \phi \right) \tag{3.40}$$

$$n = 0, \ldots, N_d - 1$$

$$\beta = \frac{f_1 - f_0}{N_d} \tag{3.41}$$

A quadratic chirp can be defined as:

$$g_{quadcrp}(N_d, f_0, f_1, \phi, v) \stackrel{\text{def}}{=} g[n] \coloneqq \text{trigf}_v \left(2\pi \left[f_0\, t[n] + \frac{\beta}{3}\, t^3[n] \right] + \phi \right) \tag{3.42}$$

$$n = -\frac{N_d}{2} + 1, \ldots, \frac{N_d}{2}$$

$$\beta = \frac{f_1 - f_0}{(N_d/2)^2} \tag{3.43}$$

And a logarithmic chirp can be defined as:

$$g_{logcrp}(N_d, f_0, f_1, \phi, v) \stackrel{\text{def}}{=} g[n] \coloneqq \text{trigf}_v \left(2\pi f_0 \left[\frac{\beta^{t[n]}}{\log(\beta)} \right] + \phi \right) \tag{3.44}$$

$$n = 0, \ldots, N_d - 1$$

$$\beta = \left(\frac{f_1}{f_0} \right)^{\frac{1}{N_d}} \tag{3.45}$$

Let us define a generalised chirp function operator:

$$g_m(N_d, f_0, f_1, \phi, v) \stackrel{\text{def}}{=} g_m[n] \coloneqq \begin{cases} g_{lincrp}(N_d, f_0, f_1, \phi, v), & m = \{0, \text{'lin'}\} \\ g_{quadcrp}(N_d, f_0, f_1, \phi, v), & m = \{1, \text{'quad'}\} \\ g_{logcrp}(N_d, f_0, f_1, \phi, v), & m = \{2, \text{'log'}\} \end{cases} \tag{3.46}$$

And finally a generalised chirp atom can be defined as:

$$d_{chirp}(N_d, k_0, k_1, \phi, v, m) \stackrel{\text{def}}{=} d[n] \coloneqq g_m[n] w[n] \tag{3.47}$$

where $k_0 = f_0 N_d$, $k_1 = f_1 N_d$ and w can be any window function.

3.6.6 Gaussian Chirps

When using chirp waveforms to modulate a Gaussian window then the family of Gaussian chirps is produced and using already introduced notation the multi-scale Gaussian chirp atoms can be defined by:

$$d_{gaussianchirp}(N_d, s, k_0, k_1, \phi, v, m) \stackrel{\text{def}}{=} d[n] \coloneqq g_m[n] w_{gauss}[n] \tag{3.48}$$

3.6.7 Gammachirp atoms

In a similar fashion to Gaussian chirps, gammachirp atoms can be produced by using chirp waveforms to modulate a gamma distribution function instead and they can be defined by:

$$d_{gammachirp}(N_d, k_0, k_1, \phi, v, m) \stackrel{\text{def}}{=} d[n] \coloneqq g_m[n] w_{gamma}[n] \tag{3.49}$$

3.6.8 Frequency Modulated (FM) Atoms

An interesting family of atoms can be obtained when considering frequency modulated (FM) waveforms. The linear, quadratic and logarithmic chirp atoms of section 3.6.5 are a type of FM atoms with linear, quadratic and logarithmic frequency modulations respectively. These types of modulation are better suited than pure tones in capturing frequency sweeps within a signal. This can be seen in figure 3.15 where an up-chirp signal signal is decomposed using GMP. The left sub-plot of the figure depicts a spectrogram of the signal and the other two sub-plots depict the decomposition of that signal using Gabor atoms and Gaussian chirp atoms respectively. In the Gabor atom decomposition it can be seen that the algorithm has selected multiple small scale atoms to represent the frequency sweep pattern. This is standard and expected behaviour of MP algorithms and one could argue that this is not such a 'bad' representation since the basic pattern of the linearly increasing frequency has been captured. Also it would be possible to group these atoms together during post-processing using for example an agglomerative clustering technique (as the one proposed in [161]). But then again this is a simplified example and given a signal with more complex frequency modulations, mixed together, then the algorithm will most probably fail to capture the important signal structures. In the right sub-plot of figure 3.15 it can be seen that the Gaussian chirp atoms have done a better job in representing the up-chirp signal. Although several Gaussian chirps have been selected (in fact half of the atoms of the previous decomposition are used) they better capture the linearly increasing frequency pattern.

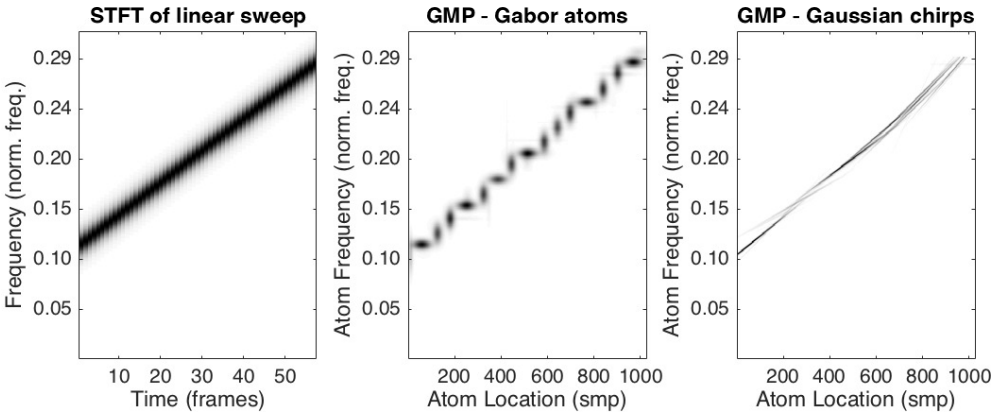


Figure 3.15: Decomposition of an up-chirp signal using GMP. The left sub-plot depicts a spectrogram of a linear chirp signal. The middle plot depicts the time-frequency energy distribution of a GMP decomposition of that signal using Gabor atoms. It can be clearly seen that the algorithm has chosen a lot of small scale atoms to capture the increasing frequency sweep. The right sub-plot depicts another GMP decomposition of the signal but this time using Gaussian chirp atoms and it can be seen that although several Gaussian chirps have been selected (in fact half of the atoms of the previous decomposition are used) these better capture the linearly increasing frequency pattern thus they are better suited for this task.

Although chirp atoms are better suited than pure tones to capturing frequency sweeps within a signal they do become somewhat limiting when trying to capture more complex signal patterns.

For example consider the vibrato effect which consists of a cyclic variation of pitch or in signal processing terms it is the effect produced by sinusoidal frequency modulation. The left sub-plot of figure 3.16 illustrates the vibrato effect, that is a signal with sinusoidal frequency modulation and the right sub-plot shows the enegry distribution of a GMP decomposition of that signal using Gaussian chirp atoms. It can be seen that the chirp atoms have captured the basic sinusoidal pattern but not precisely. Some atoms extend away from the range of the actual frequency modulation that occurs within the signal and the captured pattern seems more like a triangular frequency modulation rather than sinusoidal. Again this is expected behaviour and similar results have been produced for example in [158]. The problem is obvious; trying to capture vibrato patterns using linear or logarithmic chirp atoms is as limiting as trying to capture linear or logarithmic frequency sweeps using pure tone Gabor atoms.

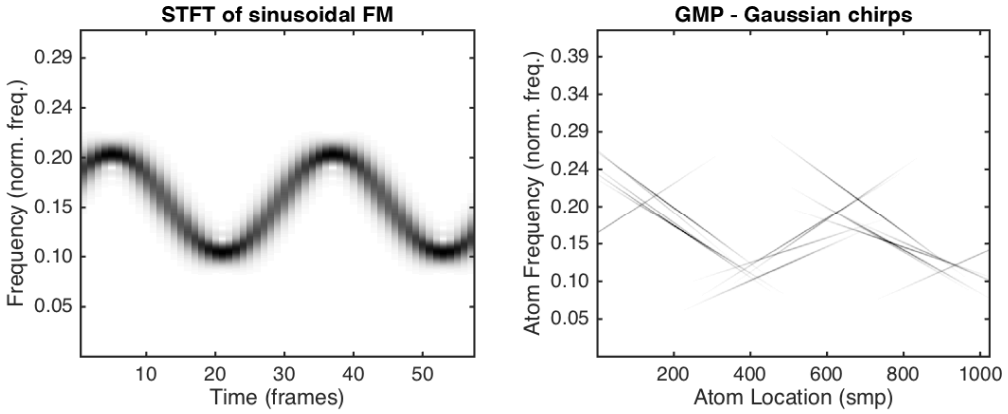


Figure 3.16

In the same way chirp atoms extend pure tone Gabor atoms, sinusoidal FM atoms extend chirp atoms. Although it is possible and useful to express a sinusoidal FM atom a more attractive approach is to define arbitrary frequency modulation (or arbitrarily frequency modulated) atoms, henceforth called AFM atoms. A signal with arbitrary frequency modulation can be defined as:

$$h_{afm}(N_d, f_i[n], v) \stackrel{\text{def}}{=} h_v[n] \coloneqq \text{trigf}_v\left(2\pi\,\text{cumsum}\left(f_i[n]\right)\right), \quad n = 0, \ldots, N_d - 1 \qquad (3.50)$$
$$0 \leq f_i[n] \leq 0.5,\ n = 0, \ldots, N_d - 1$$

where $f_i[n]$ is an instantaneous (normalised) frequency function which will be called the instantaneous frequency law (or f_i law for short) and cumsum is the cumulative sum, that is the accumulator system described by equations (2.26a) and (2.26b) and is redefined here for completeness:

$$\text{cumsum}\left(x[n]\right) \stackrel{\text{def}}{=} y[n] \coloneqq x[n] + y[n-1], \quad n = 0 \ldots N_d - 1 \qquad (3.51)$$

Equation (3.50) is very flexible and it can be used to produce any of the chirp signals described in section 3.6.5; in fact it can be used to produce pure tone signals as well. For pure tone signals the f_i law can be defined as:

$$f_{claw}(N_d, f_0) \stackrel{\text{def}}{=} f_i[n] \coloneqq f_0 \qquad (3.52)$$
$$\text{with } f_i[0] \coloneqq 0 \text{ and } n = 1, \ldots, N_d - 1$$

where the first element of f_i is set to zero so that the first element of the cumulated sum is zero which will produce the correct first sample for the cosine and sine functions.

Since the first element is forced to be zero the n index starts from one. For a linear chirp the instantaneous frequency law can be defined as:

$$f_{linlaw}(N_d, f_0, f_1) \stackrel{\text{def}}{=} f_i[n] \coloneqq \left(\frac{f_1 - f_0}{N_d}\right) t[n] + f_0 \quad \text{with } f_i[0] \coloneqq 0 \text{ and } n = 1, \ldots, N_d - 1 \tag{3.53}$$

In a similar fashion, quadratic and logarithmic instantaneous frequency laws can be defined. The sinusoidal f_i law can be expressed as:

$$f_{sinlaw}(N_d, \alpha, f_0, f_1, \phi, v) \stackrel{\text{def}}{=} f_i[n] = \frac{\alpha}{2}\left[\text{trigf}_v\left(2\pi f_0 t[n] + \phi\right) + 1\right] + f_1 \quad \text{with } f_i[0] \coloneqq 0 \text{ and } n = 1, \ldots, N_d - 1 \tag{3.54}$$

where α is the modulation amplitude and f_0, ϕ and f_1, the modulating frequency, phase and middle point of modulation of the instantaneous frequency law function respectively. The constants $1/2$ and 1 are used so that they limit the output of f_i between the allowable frequency limits $0 \leq f_i[n] \leq 0.5$.

It should be clear from the examples so far that equation (3.50) is very powerful indeed and can facilitate the creation of FM atoms. But all this flexibility comes with the expense of extra atom parameters. For the presented frequency laws only the sinusoidal law requires the extra amplitude parameter α. But using frequency laws that are produced by analytic expressions can become limiting in the variety of frequency modulations that can be created. Equation (3.50) is powerful because it allows the creation of any FM signal but for an arbitrary frequency law all elements of f_i must be stored along with other atom parameters and this will make the parameter space very large and inefficient to store. One possible solution could be similar in nature with the approach used in [173] where the authors estimate the parameter of a chirp atom by using linear regression on a set of points obtained by an STFT spectrogram. The same idea can be extended by using polynomial regression or other curve fitting techniques applied to sets of points extracted from an STFT spectrogram thus creating an instantaneous frequency law that describes a complex frequency modulation pattern. In this way only a few parameters are needed to express the f_i law and in particular the number of parameters will equal the order of the polynomial that is used in the regression step. The proposed approach is just an idea and other potentially useful solutions could be found. At the time of writing this dissertation, the GMP software package does not fully implement FM atoms therefore no further experimentation took place but the applicability of AFM atoms into the context of MP is definitely an issue worth researching.

Finally, to conclude, an AFM atom can be defined as:

$$d_{afm}(N_d, f_i[n], v) \stackrel{\text{def}}{=} d[n] \coloneqq h_v[n] w[n] \tag{3.55}$$

where w can be any window function.

3.6.9 Crackles

Crackles are transient pulmonary sounds that usually occur during the inspiratory phase. Their presence and their characteristics provide important information about the physiology and pathology of the lungs and the airways [11]. Although they are not directly related to music their time domain waveform resembles that of a gammatone atom or even a gammachirp since they have a slightly frequency modulated content and as such they could be useful in representing transient phenomena. Crackles are discussed in more detail in section 5.4. In this section only their definition is provided. A simulated crackle signal can be defined as (adapted from [193]):

$$z_{crk}(N_d, t_0, \beta, v) \stackrel{\text{def}}{=} z_v[n] \coloneqq \text{trigf}_v(4\pi\, t^{\alpha}[n]) \tag{3.56}$$

$$\text{with } \alpha \coloneqq \frac{\log(\beta)}{\log(t_0)} \tag{3.57}$$

$$\text{and normalised time } t[n] \coloneqq \frac{t[n]}{N_d}, \quad n = 0, \ldots, N_d - 1 \tag{3.58}$$

where z_{crk} is a two cycle sinusoid and parameters t_0 and β produce various configurations. When $\beta = 0.25$ (i.e. default value in [193]), then parameter t_0 dictates the initial deflection width of the wave (i.e. the duration between the beginning of the sinusoid and the first zero-crossing). The β parameter must be positive. When $\beta < t_0$ then the resulting sinusoid has a larger first cycle whereas when $\beta > t_0$ the first cycle is smaller than the second. When $\beta == t_0$ then a sinusoid of normalised frequency $\frac{2}{N_d}$ is created.

The window function can be defined as (adapted from [193]):

$$w_{crk}(N_d, b) \stackrel{\text{def}}{=} w[n] \coloneqq \frac{1}{2}\left[1 + \cos\left(2\pi\left(t^b[n] - \frac{1}{2}\right)\right)\right], \quad b > 0 \tag{3.59}$$

$$\text{with normalised time } t[n] \coloneqq \frac{t[n]}{N_d}, \quad n = 0, \ldots, N_d - 1$$

where b dictates the shape of the window.

Finally, a crackle atom can be defined as:

$$d_{crk}(N_d, t_0, \beta, v, b) \stackrel{\text{def}}{=} d[n] \coloneqq z_v[n] w_{crk}[n] \tag{3.60}$$

3.6.10 Atom Scales

So far all atom creation functions have been expressed in terms of the atom length N_d. When dealing with dictionaries of atoms it is desirable to have atoms of various lengths, that is different scales which lead to multi-scale decompositions. There are various ways to introduce scale into the structure of a dictionary. One way was presented in 3.6.3 where the Gaussian window function in equation (3.31) has a parameter that adjusts the variance of the window thus producing multi-scale atoms. But not all window functions posses this functionality. Another way to introduce scale into a dictionary is to use MPTK's approach where the STFT is used to produce a single scale Gabor dictionary. By using multiple STFTs with varying window sizes a multi-scale dictionary can be obtained.

Other transforms can be used in a similar manner (e.g MDCT). But, again, not all atom types have fast implicit implementations in terms of fast transforms. The approach used in GMP is more generic and can be applied to any type of atom. This approach involves defining a maximum allowable atom length N_{max} which denotes the first scale and all subsequent scales produce atom lengths that are smaller than N_{max}. The following equations compute the desirable left and right indices of each atom within the allowable maximum length:

$$N_{l_s} \coloneqq \left\lfloor N_{max} \left[\frac{1}{2} - \frac{1}{2^s} \right] \right\rfloor + 1 \tag{3.61a}$$

$$N_{r_s} \coloneqq \left\lceil N_{max} \left[\frac{1}{2} + \frac{1}{2^s} \right] \right\rceil \tag{3.61b}$$

$$s = 1, \ldots, s_{max}$$

where s is a scale number, s_{max} is the maximum scale which depends on the application and of course on N_{max} and N_{l_s} and N_{r_s} are the left and right indices of an atom of scale s within the allowable range N_{max}. In order to correctly compute the atoms the time vector of equation (3.26) has to be redefined in terms of s and the new indices:

$$t_{q_s}[n] \stackrel{\text{def}}{=} t[n] \coloneqq \begin{cases} n - N_{l_s}, & n = N_{l_s}, \ldots, N_{r_s}, \quad q = 0 \\ (n - N_{l_s}) \frac{N_d}{N_d - 1}, & n = N_{l_s}, \ldots, N_{r_s}, \quad q = 1 \end{cases} \tag{3.62a}$$

$$\text{with } N_d = N_{r_s} - N_{l_s} + 1 \tag{3.62b}$$

The results of applying the above equations to the equations of a Gabor atom are illustrated in figure 3.17. Note that the atoms are placed in the middle of the allowable maximum range since the time index $n = N_{l_s}, \ldots, N_{r_s}$.

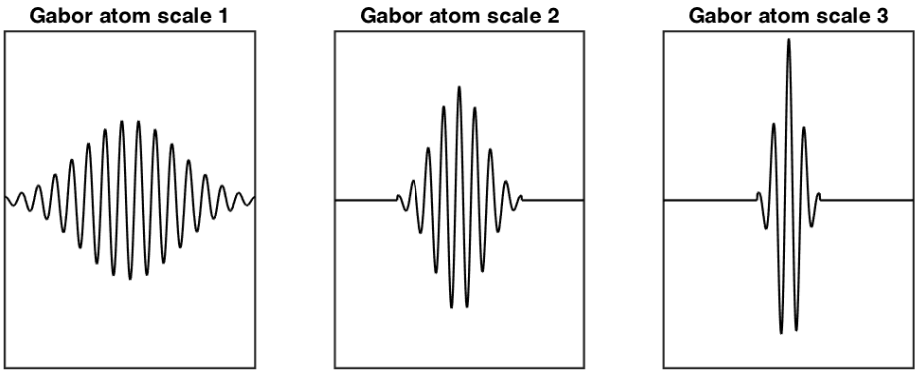


Figure 3.17: Three Gabor atoms of different scales.

Finally, all atom creation functions defined in previous sections can be expressed in terms of N_{max} (instead of N_d) and the scale parameter s to allow the creation of multi-scale atoms.

3.6.11 Inner Product Computation

One of the main bottlenecks in any MP algorithm and implementation for that matter, is the computation of cross-correlations (or groups of inner products) between the residual and the atoms in a dictionary. There are several approaches to this issue. The first one is suggested in

[68] where the authors propose the use of a modified update step based on equation (2.143). In this approach all cross-correlations between the residual and the atoms in a dictionary as well as the inner products between all the atoms in the dictionary are pre-computed and stored in memory thus avoiding any such computations within the main loop of the algorithm. Using this approach, the problem becomes memory bound rather than CPU bound and for high dimensional signals and large sized dictionaries this approach becomes impractical. Another approach proposed by MPTK is the use of fast signal transforms that allow the efficient computation of inner products between the residual and a dictionary. For example the output of the STFT is conceptually equivalent with the correlations of a signal and a set of single scaled Gabor atoms which MPTK terms a Gabor block. Any transform that can be interpreted as a set of correlations between a signal can be used, such as the wavelet transform and the MDCT-based transforms. Using multiple such transforms of various scales a multi-scale or multiple-basis analysis is possible[154]. The problem with this solution is that not all atom types can be interpreted in terms of fast signal transforms thus this approach is limiting to the type of atoms that can be used in a decomposition. When more 'exotic' atoms are used, such as chirp or harmonic atoms then other search strategies for obtained the best correlated atom must be employed. Another commonly used approach is to use a fast convolution algorithm to compute groups of inner products. Such algorithms are based on the convolution theorem explained in section 2.1.3 (equation (2.50)) which states that convolution of two sequences in the time domain becomes multiplication of their spectra in the frequency domain. The operation of transforming the sequences into the frequency domain and back becomes very efficient when the FFT is used. Since cross-correlation and convolution are related operations (as shown by equation (2.38)) a fast convolution algorithm can be used to compute the cross-correlation of sequences. This approach allows the computation of inner products between a residual and any type of atom but becomes impractical when a large sized dictionary is in use.

In GMP a fast convolution algorithm is used exclusively for all inner product computations. In particular an acyclic FFT convolution is used. To explain further, the output of a direct convolution (equation (2.34)) between two sequences of length N_x and N_h will be a third sequence of length $N = N_x + N_h - 1$. In order to account for this length when using an FFT convolution the input sequences must be zero padded so that their length is N before transforming them into the frequency domain. This process of zero-padding leads to acyclic convolution embedded within a cyclic convolution as provided by an FFT [179][5]. The fast convolution algorithm used in GMP is an adapted version of the one found in the TMP implementation [174][6]. The process is shown in the following algorithm:

Algorithm 9 assumes that the inputs are one dimensional sequences. In GMP $\mathbf{d}$ is a matrix where each column contains an atom and $\mathbf{x}$ is the residual sequence. When $\mathbf{d}$ is a matrix then the FFT of step 6 is applied to each column of that matrix and step 8 should be adapted so that the multiplication operation occurs for all atom spectra with the residual spectrum. This can be achieved by enclosing step 8 in a for-loop and $\mathbf{C}$ is now a matrix of complex spectra. In the same way step 9 computes the IFFT of each column of $\mathbf{C}$ and the produced output $\mathbf{c}$ is again a matrix with the same number of columns as $\mathbf{d}$. Finally cross-correlation of sequences $\mathbf{d}$ and $\mathbf{x}$ is achieved by passing the time reversed version of $\mathbf{d}$ into the function.

[5] http://ccrma.stanford.edu/~jos/sasp/Acyclic_FFT_Convolution.html

[6] http://uk.mathworks.com/matlabcentral/fileexchange/32426-matching-pursuit-for-1d-signals

Algorithm 9 FFT-based acyclic convolution

1: **function** $\mathrm{FFTConv}(\mathbf{d}, \mathbf{x})$
2: $N_d \coloneqq length(\mathbf{d})$ ▷ get length of sequence $\mathbf{d}$
3: $N_x \coloneqq length(\mathbf{x})$ ▷ get length of sequence $\mathbf{x}$
4: $N \coloneqq N_x + N_d - 1$ ▷ calculate minimum length of convolved sequence
5: $N_{FFT} \coloneqq 2^{nextpow2(N)}$ ▷ compute FFT size to accommodate length of convolved output. N_{FFT} should be a power of two to allow a Radix-2 FFT to be used.
6: $\mathbf{D} \coloneqq FFT(\mathbf{d}, N_{FFT})$ ▷ compute N_{FFT} sized FFT of sequence $\mathbf{d}$
7: $\mathbf{X} \coloneqq FFT(\mathbf{x}, N_{FFT})$ ▷ compute N_{FFT} sized FFT of sequence $\mathbf{x}$
8: $\mathbf{C} \coloneqq \mathbf{X} \odot \mathbf{D}$ ▷ perform point wise multiplication of spectra $\mathbf{X}$ and $\mathbf{D}$
9: $\mathbf{c} \coloneqq IFFT(\mathbf{C})$ ▷ compute IFFT of $\mathbf{C}$ to get the convolved sequence in the time domain
10: $\mathbf{c}(n) \coloneqq \mathbf{c}(n)$ for $n = 1, , \ldots, N$ ▷ trim output of convolution (in case it is larger than N)
11: $\mathbf{c}_{out}(i) \coloneqq \mathbf{c}(n)$ for $i = 1, \ldots N_x$ and $n = \left\lceil \frac{N_d+1}{2} \right\rceil, \ldots, \left\lceil \frac{N_d+1}{2} \right\rceil + Nx - 1$ ▷ get only relevant part of convolution output, that is exclude the transient response of $\mathbf{d}$
12: **return** $\mathbf{c}_{out}$
13: **end function**

The main events of algorithm 9 are illustrated in figure 3.18 where the top sub-plots depicts an input sequence $\mathbf{x}$ and a Gabor atom $\mathbf{d}$ which are convolved as shown in the algorithm to produce the convolved sequence $\mathbf{c}$ which is depicted in the left sub-plot of the second row of that figure. The dotted lines in this sub-plot define the relevant region of the convolved output which is depicted in the right sub-plot of the second row.

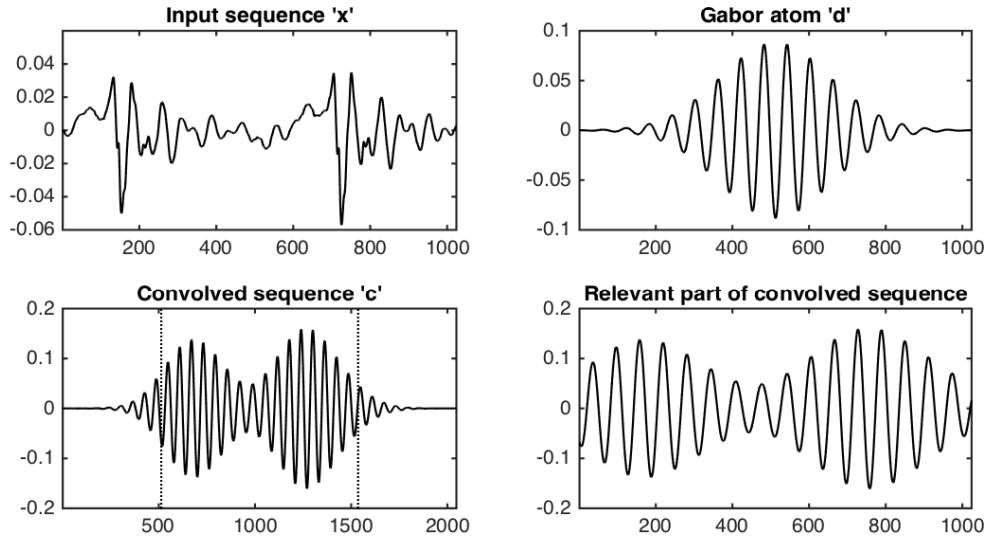


Figure 3.18: Demonstration of acyclic convolution using the FFT. The first row of sub-plots depict an input sequence 'x' and a Gabor atom 'd' before zero-padding. The length N of the convolution sequence is calculated by $N = N_x + N_d - 1$ where N_x and N_d are the length of the input sequence and the atoms respectively. The left sub-plot of the second row depicts the output sequence of the convolution between 'x' and 'd' which is performed as explained in algorithm 9. The zero padding is implicit and is performed by the actual FFT function. Note that the size of the output sequence is N. The dotted lines in this sub-plot show the relevant part of the convolution with parts on either side of the lines being the transient response of 'd'. Finally the right sub-plot of the second row depicts only the relevant part of the convolved sequence (the region within the dotted lines in the previous sub-plot).

3.7 Atom Selection Functions

The atom selection function is step 5 of the GMP algorithm (algorithm 4). This function can be thought of as a cost function that essentially selects the 'best' atom out of the dictionary. So far and for the rest of this dissertation, the word best is always quoted, since its meaning always depends on the context of a particular implementation. For example if the standard MP approach is used then the 'best' atom is the one that minimises the energy in the residual at each iteration. Of course the minimisation of the residual's energy is necessary for the algorithm to converge but the greedy approach of standard MP is not the only way to achieve this. For example in [148] a new similarity measure is introduced, other than the standard inner product, that emphasizes local fit over global fit at each iteration. As such the 'best' atom in this case would be different than the one selected if the usual inner product was used as the similarity measure. Another example is the work proposed in [123] and [112] where the atoms of the dictionary are weighted by perceptual masks, which implies that some atoms have higher probability of being selected than others. Again in this case, the 'best' atom would not necessarily correspond to the one obtained by the standard MP. In GMP, atoms are selected based on guide map which does not always guarantee that the selected atom is the one that minimises most of the energy in the residual. Finally in chapter 5 several atom selection functions will be described that are designed so that the 'best' atoms are the ones that follow certain sparsity and spatial constraints.

Before finishing this section, it should be noted that in practice, especially in the GMP implementation, step 5 of the algorithm does not return an actual atom but certain parameters of that atom such as its location along a particular support of the signal and its index in the mini dictionary which index is then used to get other parameters such as type, scale, frequency, phase and more.

Chapter 4

Guided Matching Pursuit Coder Toolbox (GMPC)

4.1 Why Yet Another Implementation?

Guided matching pursuit (GMP) is a variant of the matching pursuit algorithm (MP). A MATLAB® toolbox, termed "Guided Matching Pursuit Coder" (GMPC), has been designed and developed in order to aid research using this new MP variant. One could question the need for yet another toolbox for MP. There are actually plenty of reasons that such a toolbox was created and these are explained in this section.

One of the main ideas of the research presented in this dissertation was the use of adaptive decompositions as the front end to a source separation system. MP was a strong candidate, mainly because of its relative simplicity, intuitive interpretation of the resulting signal representation and flexibility of modification of that said representation. As such the journey begun to look for available MP software to help with the research of some new ideas. Unfortunately, it soon became apparent that MP software, for audio use, is not widely available. One would believe that because of the age of the MP algorithm (it was introduced in 1993) and the wealth of audio applications that it is being used for, one could find a good amount of software, already available to the public. But that is not the case. One reason could be the iterative nature of MP which makes it a very slow algorithm, compared to other transforms and decompositions such as the FFT, the MDCT, DWT or WPD. The real-time execution requirement of most audio systems asks for fast algorithms, which could be a reason why MP is not favoured as a solution to particular problems. For example consider phase-vocoder technology. Everything that can be done using a phase vocoder could be easily achieved using MP and one could argue that, in some cases, the results of certain audio modifications can be of better quality when using MP, mainly because it is an adaptive decomposition, compared to the phase vocoder's FFT which uses a fixed basis. Regardless of any possible benefits of using MP, the phase vocoder is still widely used with one of the main reasons being the speed of the algorithm, which makes it suitable for real-time execution.

Of course, it could be possible to implement a real-time version of MP but in order to do so, a specialised version of the algorithm is necessary. And this specialisation leads to another reason why MP software is not widely available. Researchers dealing with MP, usually try to adapt the basic MP algorithm to their needs, with one major need being fast execution. In many cases though the tricks used to achieve any such improvements, lead to over-specialised algorithms that are only good for the application at hand thus losing any generality of use. On a similar note, a lot of research focuses on the theoretical development of MP technologies, in which cases the researchers only require a short, specialised implementation of the algorithm that demonstrates the main idea. Researchers try to solve a theoretical problem and the solution is expressed mathematically in the relevant paper; there is no need for a coherent, re-usable and sustainable software implementation. Usually such implementations never find their way to the public domain. Another aspect of this issue is the nature of the signals that are tested. In some cases the signals are specifically synthesised for a particular solution; in other cases the signals are too small (only a few samples in duration) or lengthy signals are made smaller by down-sampling. Of course for the purposes of testing a newly proposed algorithm this is perfectly acceptable, but again, this practice of test data 'selectivity' leads to over-specialisation of algorithms and, in turn, their implementations.

MP's low demand due to slow execution speeds and over-specialisation of MP algorithms stemming from theoretical developments could be some possible reasons for the small amount of audio related MP software implementations available to the public. But even if these arguments are correct, the small amount of available software implementations is not enough of a reason to justify the need for a new toolbox. What about the implementations that are already out there? Couldn't we just find a suitable one, adapt it and perform our experiments? Unfortunately no. Trying to use existing software by adapting it, poses several problems and in most cases it is a difficult, error-prone and time consuming process.

Many of the MP implementations that are available are not suited for all purposes and applications; they have specific goals and are targeted to the application domain that is being studied at the time. This follows the 'over-specialisation' argument made earlier. Implementing MP for a particular purpose, usually means optimising the algorithm for a specific problem, thus making the adaptation of existing code to other problem domains difficult. Another issue to consider is that MP algorithms are implemented using different programming languages which again poses a problem when trying to adapt existing code. An interested developer might not be familiar with a specific programming language and in some cases it might be faster to implement a new version rather than trying to learn a new programming language, study existing code and adapt it. Following this, some programming languages are better for prototyping and experimenting with new ideas than others (MATLAB® vs C/C++ for example).

The most important reason for a new toolbox was the need for particular features necessary for implementing GMP and other generic features that cannot be found in any other existing implementation. In the following subsections the main goals of the toolbox are stated followed by brief descriptions of publicly available MP implementations and reasons as to why these were not suitable for the purposes of this research.

4.1.1 Toolbox Goals

The main goals of the GMPC toolbox are the following:

- Demonstrate the applicability of GMP in audio signal processing and especially sound source separation problems.
- Follow OOP design principles.
- Provide a single interface for comparing various MP implementations.
- Provide a single interface for evaluating source separation algorithms.
- Be easy to use.
- Promote reproducible research.

4.1.2 Audio Related MP Implementations

The following list provides all found, publicly available MP software and brief discussions as to why each software implementation was not suitable for the purposes of this work.

- Original MP software from Mallat and Zhang [68][1]. This is probably the first MP implementation in the form at least that was proposed by the original authors of the algorithm. This software is written in the C programming language. The software lacks many of the requirements that the GMPC toolbox tries to cover. First of all it is written in C which is not the best language for our purposes (it is difficult to write object oriented (OO) programs using the C programming language). This software also seems to lack the ability to process multi-channel signals and has no flexibility regarding the use of dictionaries and atom selection functions.
- The Matching Pursuit Toolkit (MPTK) by S. Krstulovic et.al [154][2]. This is probably the best MP open source software available. It has many of the features and shares similar goals with the GMPC toolbox but has some intricacies that do not make it suitable for the purposes of this research. The software is written in C++ which although is well suited for OOP, is not the best choice for such a toolbox. C++ is an excellent language but it is not good for prototyping software. First of all C++ is a compiled language in contrast to MATLAB®'s interpreted environment, thus making debugging in C++ much harder. Also it is a difficult language to master and much more prone to programming errors. Rewriting MPTK to MATLAB® would have been a very large task. Having said that MPTK has MATLAB® bindings but these are not suitable since they are essentially wrapper functions to C++ code, thus not much can be done to change the functionality of the toolbox at the MATLAB® level. Adding the required GMP functionality to MPTK would have been very difficult as well. Just trying to fit the idea of GMP into the existing MPTK C++ design seemed impossible without severely altering the code; it would have been the same

[1] `ftp://cs.nyu.edu/pub/wave/software`
[2] `http://mptk.irisa.fr/`

as writing a new software all together. Apart from programming hurdles, MPTK seems to lack certain features that are pivotal to the implementation of GMP. First of all there is no clear approach (if any) of how to allow the user to choose the way to process multi-channel signals. Also there is no clear way of how one can define new atom types and incorporate different atom selection functions. Finally the toolkit does not provide a way to compare various MP implementations. MPTK is an excellent piece of software but unfortunately lacks many of the specific features required by GMP. Despite this, MPTK is used throughout this research as a baseline software for comparison purposes.

- Matching Pursuit of 1D signals by Patrick Mineault [174][3]. This is a MATLAB® implementation of basic MP using some of the tricks used in MPTK. This is a short implementation which is very fast but unfortunately it is limited to single channel, short duration signals and dictionaries comprising few atoms. It is an excellent demonstration of a basic MP implementation and actually was the starting point for the GMPC toolbox.

- GabLab written by Kereliuk C. [136][4]. Another MATLAB® implementation aimed at comparing various sparse approximation algorithms. The toolbox provides a wide range of algorithms and has been written with audio analysis in mind [136] but the main goals of GabLab are completely different from the goals of GMPC and as such the design of this software cannot form a good base for the GMPC toolbox.

4.1.3 Other MP implementations

- MATLAB®'s MP implementation wmpalg [152][5]. This implementation offers three algorithms: basic MP, orthogonal MP and weak MP. These are 'text-book' implementations of the algorithms and although they provide a good starting point for understanding MP they were not suitable for the purposes of this work. The implementation can only process single channel, short duration signals and it seems that it is meant to be used as an educational tool rather than a full blown MP solution. It lacks many of the required features and goals of GMPC toolbox.

- MP using stochastic time-frequency dictionaries by Dobieslaw Ircha and Piotr J Durka[153][6]. The software is written in C and was mainly developed for the analysis and processing of EEG signals. This is an example of an over-specialised version of MP, targeted to a certain class of signals, using specific types of dictionaries and trying to overcome a specific MP problem, namely the statistical bias of a decomposition when the standard Gabor dictionary is used. Although this software has some nice ideas it lacks many of the required features for the proposed toolbox.

- SparseLab written by Donoho D. et al. [194][7]. It is a MATLAB® implementation that provides solvers for MP, Orthogonal MP and Basis Pursuit. The toolbox does promote some of the goals of the GMPC toolbox, such as re-producible research but lacks many features

[3] http://www.mathworks.com/matlabcentral/fileexchange/32426-matching-pursuit-for-1d-signals
[4] http://www.music.mcgill.ca/~corey/dafx2011/
[5] http://uk.mathworks.com/help/wavelet/ref/wmpalg.html
[6] http://eeg.pl/mp
[7] https://sparselab.stanford.edu/

required for working with audio signals. Also the main goal of SparseLab is to find sparse solutions to systems of linear equations, particularly under determined ones and provides many algorithms for doing so and as such it is yet another specialised implementation.

- Sparsify written by Blumensath T. [195][8]. This toolbox provides a set of functions that implement various algorithms for sparse signal approximations. Again this toolbox is of generic nature and it is not tailored for processing large audio signals. Similarly to all other MP implementations mentioned in this section, this toolbox lacks many of the desired features of GMPC.

To summarise, the main reasons for creating another MATLAB® toolbox are the following:

- Lack of publicly available MP software implementations for audio use.
- Different goals / problem domains of various MP implementations.
- Difficulty in adapting existing code in the context of the GMP framework.
- Lack of required features/functionality in existing implementations.

4.2 Specification

The new toolbox should provide classes and/or functions that:

- Perform GMP of one-dimensional signals, with emphasis given to audio signals.
- Be able to process single-channel and multi-channel signals of any duration.
- Allow the user to choose the way multi-channel signals are processed.
- Allow the user to use multiple pre-defined dictionaries and/or add dictionaries using a common and simple programming interface.
- Provide the user with a simple interface to experiment using various atom selection functions which govern the way a signal is decomposed.
- Provide the user with generic tools for the visualisation and/or modification of the resulting signal representations.
- Provide a common interface for the inclusion of other MP algorithms with the aim to compare and test various MP implementations.
- Be written in the MATLAB® programming environment using advanced MATLAB® features such as MATLAB® classes and object oriented programming (OOP) principles.
- Provide a consistent interface for performing evaluation of source separation algorithms.
- Provide a simple interface to construct experiments, thus promoting re-producible research.

[8] `http://www.personal.soton.ac.uk/tb1m08/sparsify/sparsify.html`

4.3 Software Testing

This section deals with software testing as part of the development cycle. This is the first part of testing from a series of tests that aim to prove the correct operation of the software. The section continues by defining the software testing goals, explaining and justifying the software testing strategy and finishes with some test examples and discussions on results.

4.3.1 Software Testing Goals

The testing and verification of software is an integral part of any software development cycle. Simply put, the main goal is to prove that the software functions correctly. In particular the software should always react correctly to user input, produce expected and meaningful results, meet the specifications and satisfy the requirements that were set in the initial stages of the development. We should note that testing software is a very complicated process and trying to test against all possible inputs, for all possible conditions while making sure all output is correct, is not always feasible, especially for complicated implementations such as this one. Therefore the purpose of this section is not to provide an exhaustive list of examples with their outputs but rather explain the software testing strategy that was used and justify why this strategy is sufficient to prove that the software works. The software is further tested by the use of a few representative examples that cover most common modes of operation and the output of those tests is inspected and validated.

4.3.2 Software Testing Strategy & Results

First of all the main purpose of the GMP algorithm is adaptive signal decompositions. This is the one and most critical task that the software performs. By altering the atom selection step, the algorithm can be allowed to perform other tasks such as source separation. It is clear that an erroneous output during the decomposition of a signal will hinder the output of any other subsequent task (source separation, mixing matrix estimation etc.). As such the main goal of the testing strategy is to verify that the decomposition aspect of the software performs correctly. In order to achieve this, a series of testing methods and software design decisions were employed. These are explained in the following subsections.

Testing of Critical Software Components

The GMPC toolbox comprises two interfaces: a set of functions that perform specific, single tasks such as inner product calculation between a signal and a dictionary, dictionary creation, FFT/STFT analysis etc., and a set of classes that call these functions to perform more complicated tasks, such as signal decomposition, source separation and mixing matrix estimation. Since the classes depend on the individual functions, the testing and verification of those functions is of paramount importance. These functions are actually the backbone of the system, the critical software components that form part of the GMP algorithm. If any of these components fail then the system will surely produce wrong results or it might even break.

Some critical components were tested by cross-checking the results with other software implementations that performed the same task and are assumed to be working correctly (e.g. MATLAB®'s built-in functions). In the case where no alternative method was available, the components were tested extensively by the author using a series of test inputs and inspecting the output. All test files can be found in the 'tests' folder of the provided toolbox.

The following software functions were tested and found to operate correctly:

- 'xcorr', 'fftconvs', 'fftconvl', 'fftconvf': These functions perform cross correlation between a signal and a dictionary of atoms. The algorithm uses the FFT to calculate groups of inner products (a sliding inner product). This is a common algorithm with many available software implementations. The computation of inner products is step 4 of the GMP algorithm (algorihtm 4) and is invoked at least once per iteration. For this test four signals (voice of a soprano, male speech, a guitar and a drum sound) were cross correlated with a Gabor atom individually, using MATLAB®'s 'conv' function, and GMPC's 'fftconvs' function. These functions perform the same operation (although they are implemented differently) and we expect the output to be identical. The results of this test are presented in figure 4.1 where the plots depict the difference between the outputs of the conv and the fftconvs functions which are almost zero (the amplitude of the y-axis is in the order of 10^{-14}). The small differences between the outputs of these functions are due to round-off errors produced by the different implementations.

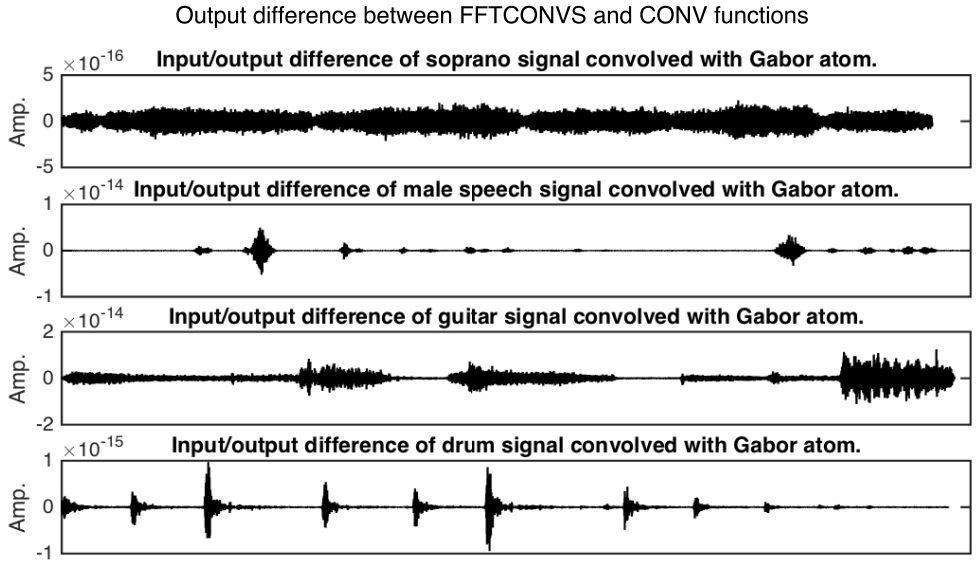


Figure 4.1: Output difference between fftconv and conv functions for various audio signal types.

- 'cztk': This function performs a chirp-Z-transform. It can be used in the pre-analysis step (step 3) of the GMP algorithm to calculate frequency and phase estimates of possible atoms. It is a wrapper function to MATLAB®'s 'czt' function so we can safely assume that it works correctly. A demonstration on a signal can be found in the 'tests' folder. For this test an excerpt of a piano signal was transformed using MATLAB®'s 'czt' function and GMPC's 'cztk' function and the magnitude spectra of the outputs were subtracted. The difference between the magnitude spectra of the two outputs is depicted in figure 4.2 where it can be clearly seen that it is zero, thus confirming correct operation of 'cztk'.

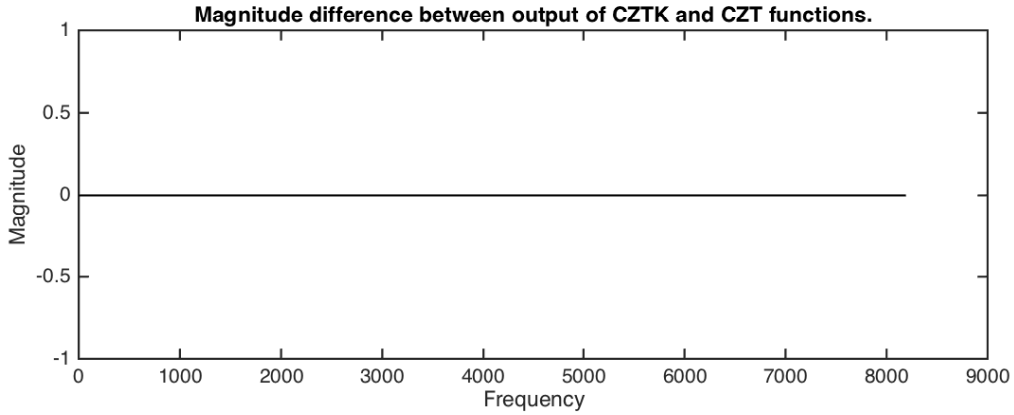


Figure 4.2: Magnitude difference between outputs of the cztk and czt functions.

- 'mixmatst': This function produces mixing matrix coefficients based on various panning laws. Parts of the implementation is based on source code found in [103]. The correct results are verified by figure 4.3 that shows the correct mixing matrix coefficients for every angle from -45 to 45 degrees for the constant power mixing law. The botton plot of the figure shows that the sum of the coefficients from the left and right channels add up to one which is the expected output.

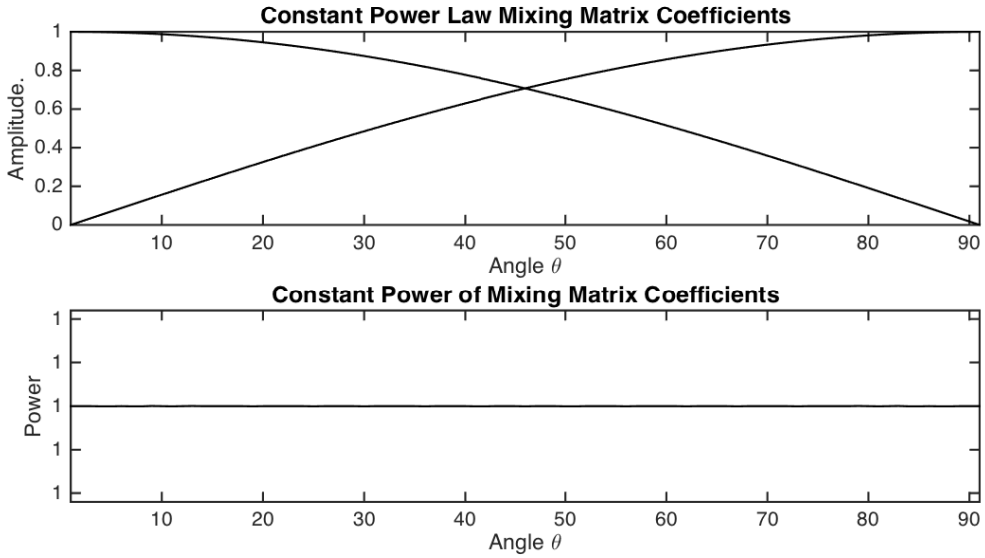


Figure 4.3: Testing of mixmatst function.

- 'atomgab', 'atomsin', 'atomcos', 'atomgmt' etc.: These are atom generation functions. Each of these functions is responsible for producing atoms of various types. The generation of various types of atoms is described in detail in section 3.6.

- 'wingaussn', 'wingamma', 'winrect': These functions are responsible for generating window functions used during atom creation. Their implementation is based on equations that are found in the literature. Correct operation was verified by cross-checking the output of these functions with their MATLAB® equivalents. Some examples are included in the 'tests' folder.

- 'qfit', 'qeval': These functions are part of the QIFFT algorithm [107] which can be used in the pre-processing step of the GMP algorithm. The 'qfit' function performs quadratic interpolation of three uniformly spaced samples and 'qeval' evaluates a point of a parabola given its coefficients. These are simple, straightforward implementations and similar versions can be found in the implementation of PARSHL [107]. For this test four sets of three ordinate values each were fitted to a parabola using the 'qfit' function and the coefficients of the parabola were obtained. The coefficients were then evaluated over a range using the 'qeval' function. The results of this process are verified by figure 4.4.

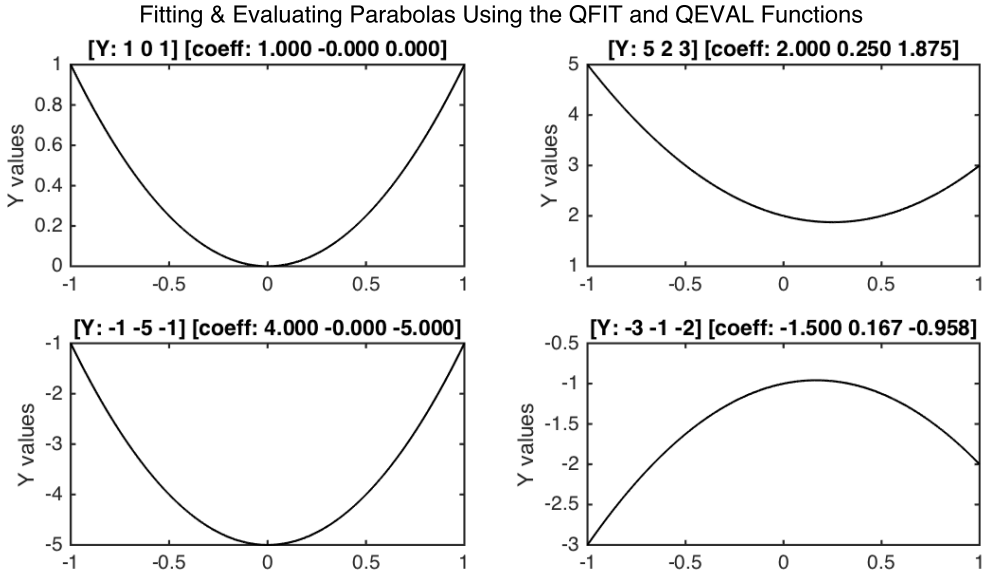


Figure 4.4: Fitting and evaluating parabolas using the qfit and qeval functions.

Testing and verification of these critical components was the first step of the testing strategy. By having these critical components tested and verified it can be ensured that any subsequent errors will be due to incorrect inter-connections of these components (i.e. by creating classes) rather than the components themselves.

Testing of Software Systems

The software components of the previous section form the building blocks for the class design. In OOP, classes define data (constants, variables, data structures and other classes) and methods (critical software components, utility functions) that act on that data. By connecting classes we can create larger systems that perform complicated tasks such as adaptive decomposition of a signal and source separation. The GMPC toolbox has several such classes, some of which are meant to be used as components in other classes. Two main classes are 'gmpenc' and 'gmpdec' which encode and decode a signal respectively, using GMP. Those classes depend on functionality of other classes. We can think of 'gmpenc', 'gmpdec' and all interconnected classes as the 'coding system' of the toolbox and as the first critical system in the toolbox on which other systems build upon. Therefore verification of this system is important.

One test was performed to prove that the system is an identity system. The system should also be a lossless system, in an energy sense, as long as the residual signal is included during the synthesis stage. To test the coding process, a few audio samples were decomposed using default settings and then synthesised back to produce the original. In particular all the audio samples were decomposed using an STFT pre-processing step with an FFT size of 8192 samples and $4\times$ overlap factor (which corresponds to the 'lfmp4x' setting of GMP) and a typical Gabor dictionary of 5 scales. The signals were decomposed into 1024 Gabor atoms each. The test files can be found in the 'tests' folder of the provided software. Figure 4.5 shows the *signal to noise ratio* (SNR) of the decompositions which was calculated for each test signal using $SNR = 10 \cdot log10(\frac{E_r}{E_n})$, where E_r and E_n are the energies of the reconstructed and difference (noise) signals respectively. It can be seen that the SNR for all signals is above 150 dB. Differences between the original and the reconstructed signals (for each test signal) are also shown in figure 4.6. We can see that the differences are negligible and most probably due to round-off errors which implies that the system behaves as an identity system (i.e. there is no loss of energy during the process). Of course, exhaustive testing for all possible parameter combinations with all possible audio signal types is not possible, but a good selection of audio samples has been chosen and the test cases cover at least the default settings of the algorithm and the type of audio signals covered in the specification.

Testing and verification also occurred throughout the development life cycle, similar to agile software development methods. The first algorithm that was developed was the simplest possible implementation of GMP with each subsequent version enhancing the basic functionality. Since the first implementation is based on the already tested and verified critical software components we can safely assume that it works correctly. Any new version of the software should produce the same output given the input data and parameters are the same. Following this logic a test was performed where at each version step the same signal was processed (encoded/decoded) and produced the same results.

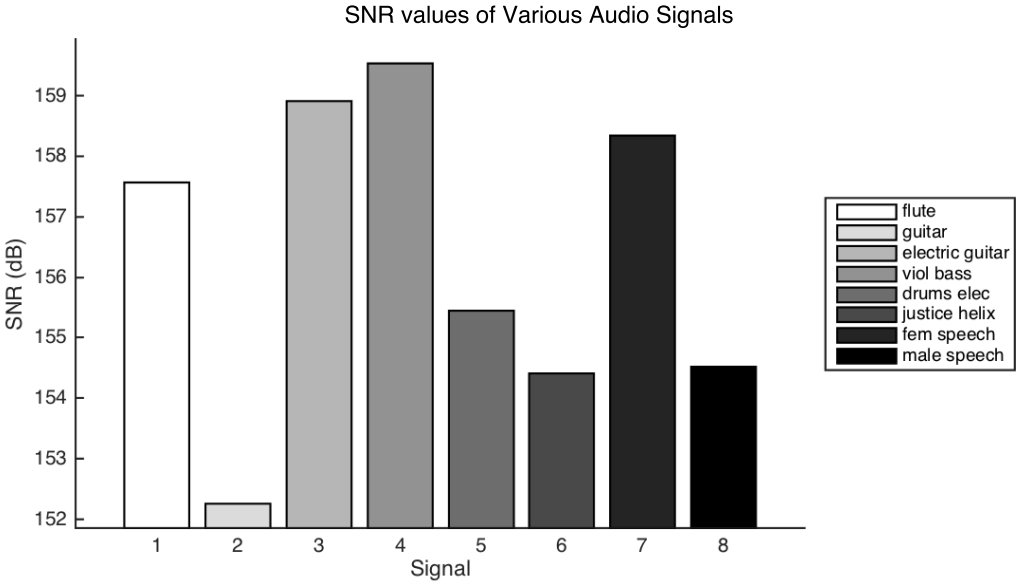


Figure 4.5: Identity system test - Signal to Noise Ratio (SNR) of various audio signals. The SNR is computed using: $10 \cdot log10(E_r/E_n)$, where E_r and E_n are the energies of the reconstructed and difference (noise) signals respectively.

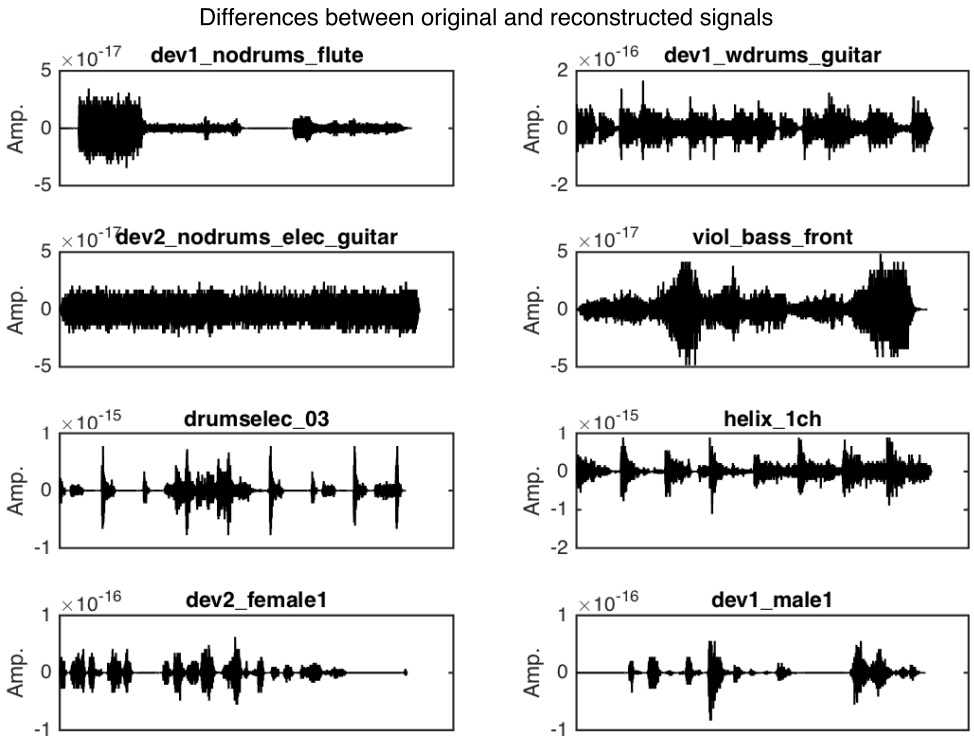


Figure 4.6: Identity system test

Design Decisions That Aid Testing & Debugging

Testing and debugging functionality was in mind from the early stages of the development process and influenced the class design considerably. In particular the class design has certain features that allow for robust input parameter checking/verification and easy debugging. These are the following:

Single, Coherent API The class design is loosely based on MATLAB®'s 'System Objects' [196][9]. Following the coding standards found in [196] ensures that the design follows a well established and tried design, which is a general solution to many audio algorithms. Also following these coding standards ensures a certain compatibility with the 'Systems Object' architecture and not inheriting from 'matlab.System' class allows the toolbox to be flexible. Finally, following a strict execution pattern for all classes (object construction, initialisation, setup, stepping through algorithm, exporting results, deletion) can aid with identifying and debugging errors quickly.

No Input Argument Constructor Requirement The classes in the toolbox fulfil the "no input argument constructor" requirement which as the name suggests, allows a class constructor to be called with no input arguments. If no input arguments are used, the newly created object is initialised to default values. This requirement is suggested in [197][10]. This step does not verify and/or validate the input arguments; it makes sure that all important variables and data structures in the class are initialised to a sensible value which is a good programming practice since it prevents any errors that arise from uninitialised variables.

[9] http://uk.mathworks.com/help/comm/gs/what-are-system-objects.html
[10] http://uk.mathworks.com/help/matlab/matlab_oop/class-constructor-methods.html

Robust Input Argument Checking All GMPC classes have a standardised mechanism to enumerate input arguments and assign them correctly to internal data. Employing such a mechanism helps prevent errors that originate from erroneous user input. If an unknown input argument is detected the mechanism ignores its value and leaves the default value of that parameter untouched. This is essentially implemented using a loop going through all input arguments and a switch statement within the loop processing only character inputs. All possible algorithm parameters are expressed in "property name, property value" pairs so the above described logic will successfully enumerate all property names, without missing any and ignoring unknown ones without flagging an error.

Input Parameter Verficiation Method Another testing feature is the 'validateParameters' method found in all GMPC classes. This method is called by the 'setup' method and implements logic that checks and verifies the correctness of the input data. Each class has specific implementations of such methods that depend on the problem the class is trying to solve. For example for an encoding task, the length of the atoms cannot be larger than the signal itself or for a source separation task a stereo signal is always needed. Such errors can be detected inside the 'validateParameters' method. If an error is detected it is either corrected automatically (if possible) or displayed to the user with an appropriate error message. The most important classes (including 'gmpenc' and 'gmpdec') have fully implemented validation methods whereas secondary classes have limited implementations. This is because of the time-consuming aspect of this coding task. It should be clear that a validation method is a useful design feature and greatly minimises any errors stemming from incorrect user input.

'Debug on Error' Feature All code found inside the setup, validation and step methods of each class is enclosed in 'try-catch' statements. As such when an error occurs the code execution will stop at the appropriate catch statement. The class design employs an error reporting mechanism which explains the most basic and obvious errors and some times suggests possible solutions. In case of an unknown error the "Debug On Error" feature allows the user to enter into debug mode when an error occurs. The debug entry point is at the catch statement the error occurred. This way the user can delve into the code and inspect the value of all variables at the time of the crash. This is a useful feature which enables easier debugging.

Debugging Log A final aid for debugging is the ability to log all program output. This can be useful when the software is running unsupervised and an error occurs. Future versions of the software will be able to also send an email message to the user when the error occurs (a class providing a basic form of this functionality already exists but is not incorporated into the main system).

4.3.3 Software Testing Discussion

In this section the software testing issue was discussed as required by a typical development cycle. The main aim of this testing was to prove correct operation of the software. Because exhaustive testing of all possible inputs and conditions is not feasible, a testing strategy was devised to try and prove the correctness of operation. This goal was achieved by the following:

- Testing and verification of critical software components. The correct operation of these building blocks ensures that systems that depend on these components will also perform correctly as long as the building blocks are inter-connected properly.

- Testing and verification of the basic coding system (i.e. encoding/decoding process). Correct operation of the coding system ensures that the building blocks have been inter-connected correctly and, to some extent, ensures correct operation for other dependant sub-systems.

- An iterative approach to testing the main algorithm. The simplest possible GMP implementation was first coded and tested for correct operation. Every new version of the software was tested with the same input to produce either identical or 'improved' (in terms of performance metrics) output to the previous version. That is an indication that the added features have not changed the behaviour and results of the software in an undesirable way.

- Following a tried, established, and clear design pattern for the classes ensures that no errors due to design problems occur. The class design at this point is fixed and any additional features can be implemented easily, following a stream-lined approach. Also following this pattern allows for easy inter-connection of classes to form larger systems which are expected to perform correctly since they are build on tested and verified foundations.

- Following MATLAB®'s suggested coding practices. Good coding practices combined with a solid class design ensures that the classes will behave as expected when included in bigger systems.

- Having a robust input argument checking mechanism. This ensures that any errors due to erroneous user input are eliminated.

- Enclosing critical sections of the code within 'try-catch' blocks. Using this practice an error can be captured and either corrected or displayed to the user. This practice combined with the "Debug On Error" and "Debugging Log" features helps with identifying errors early and makes debugging easier.

A testing strategy was proposed and justified with appropriate examples and figures. Although this is not exhaustive testing, the testing strategy, combined with the use of a robust class design and good coding practices, ensure to a great extent, that the core of the toolbox, i.e. the critical software components in the list of section 4.3.2 and the coding system classes ('gmpenc', 'gmpdec' and associated classes), perform as expected, at least for the cases they were tested for.

4.4 Performance Testing

This section presents performance testing of the coding software system. In contrast to the previous section that tested whether the core of the toolbox works correctly, this section focuses on the performance and behavioural aspects of the coding system which are evaluated by a set of metrics and by comparison to similar software. As already mentioned, the coding system consists of the gmpenc, gmpdec and associated classes and forms the basis for the source separation algorithms that are investigated in chapter 5. Therefore evaluating the performance of this system is important. This section has a similar structure to the previous one. Firstly, the set of performance metrics that are going to be used in the evaluation are presented, followed by the testing goals and methodology. Then appropriate examples are presented followed by results and discussion.

4.4.1 Performance Metrics

In order to evaluate the results of the coding system some common metrics that are found in the literature are employed. In particular the following are used:

- Energy Decay Curve (EDC)
- Signal to Residual Ratio (SRR)
- Relative Error in ℓ^2 norm (L2Err)
- Coefficient Cumulated Quality (CCQ)
- Execution Time (ET)

All these metrics were described in section 2.3.5.

4.4.2 Performance Testing Goals

In chapter 3, the GMP algorithm was described in full and several factors that can potentially affect system performance were pointed out. Section 3.3, describes three ways to process long duration signals. Section 3.4 suggests various algorithms for the pre-processing step and section 3.5 proposes three different ways to process multi-channel signals. Section 3.6 describes how to create analytic dictionaries with multiple atom types and section 3.7 briefly discusses atom selection functions. All these parameters affect the performance of the coding system in some way. It is clear, similarly to the software testing strategy, that a brute-force testing approach is not feasible. Instead we need to consider issues we want to examine and suggest testing methodologies for doing so. In particular the goals are to provide a numeric description of system performance and also compare the proposed system with another baseline system. Regarding system performance the following questions are addressed:

1. Does the algorithm behave as expected, when tested with various types of audio signals and using the default settings?

2. How does the pre-processing step affect performance?

3. Does the use of multi-dictionaries improve performance?

Performance is further tested and evaluated by comparing the proposed system with the MPTK software solution. This comparison is not meant to prove which software is better, but rather try to evaluate the performance and behaviour of the proposed system when compared to a baseline system, i.e. a system we assume works correctly. The main question to answer is the following:

4. How well does GMP compare to MPTK?

Each question will be answered through a series of tests with appropriate examples. It should be noted that these examples are not meant to provide an exhaustive testing coverage but rather a representative review of the issues that are being tested.

4.4.3 Performance Testing Methodology

Despite having multiple questions, only a single methodology is needed, that is generic enough for all tests. The main idea is to decompose a given audio signal using the GMP and MPTK algorithms with certain settings. The type of the audio signal to use as well as the algorithm settings depend on the test that is being performed. The decomposition results are then evaluated using the metrics of section 2.3.5. This process repeats until all samples have been processed and finally the evaluation metrics of all signals are compared and further discussed. How to interpret the metrics is also dictated by the question that is being investigated. The following generic testing methodology is proposed:

1. Choose a signal to evaluate. The type of signal depends on the test being performed.

2. Decompose the signal using GMP or MPTK with appropriate settings. The settings are dictated by the question being investigated.

3. Get performance metrics from the decomposition results.

4. Repeat until all audio samples have been processed. The number of audio samples depends on the test being performed.

5. Compare the generated metrics. How to interpret these results depends on the question that is being answered.

The rest of this section describes in more detail the tests that were performed. Each test tries to answer a question from section 4.4.2. In the following subsections each test is described and also the chosen sets of audio inputs and algorithm settings are explained and justified.

Testing the Effect of the Pre-Processing Step

The pre-processing step is one of the main features of the GMP algorithm. As already explained in section 3.4 this step performs some kind of analysis on the residual in order to extract important features which are then used for the creation of mini dictionaries. It should be clear that the quantity and quality of those features depends on the kind of pre-processing. The analysis techniques mentioned in section 3.4 were evaluated with this test. In particular the following configurations were evaluated:

1. '*lfmp2x*': This configuration uses the STFT as the pre-processing step. The STFT was performed using an FFT size of 8192 samples with $2\times$ overlap factor (i.e. a hop size of 4096 samples) and a 8192 samples long blackman window.

2. '*lfmp4x*': Same configuration as 'lfmp2x' but with a $4\times$ overlap factor (hop size $= 2048$ samples).

3. '*lfmp8x*': Same configuration as 'lfmp2x' but with an $8\times$ overlap factor (hop size $= 1024$ samples).

4. '*qislfmp2x*': This configuration uses a quadratically interpolated STFT (QISTFT) as the pre-processing step, using an FFT size of 8192 samples with $2\times$ overlap factor and a 8192 samples long hamming window.

5. '*qislfmp4x*': Same configuration as 'qislfmp2x' but with a $4\times$ overlap factor.

6. '*qislfmp8x*': Same configuration as 'qislfmp2x' but with a $8\times$ overlap factor.

7. '*qi1lfmp*': This configuration uses a quadratically interpolated STFT (QISTFT) as the pre-processing step with the settings suggested in [107]. In particular a hamming window with a fixed duration of 45 ms was used with $2\times$ overlap and an FFT of size 8192 samples. This particular configuration uses the first frequency estimate that is produced by the QIFFT algorithm.

8. '*qi2lfmp*': Same configuration as 'qi1lfmp' but a second, more refined frequency estimate is used.

9. '*qi3lfmp*': Same configuration as 'qi2lfmp' but a third, more refined frequency estimate is used.

All above configurations were used to decompose each of the signals mentioned in 4.4.4 using 2048 iterations (i.e. 2048 atoms for each decomposition) and a typical multi-scale Gabor dictionary comprising five scales. The results produced were evaluated using the metrics of section 2.3.5. Some of these results are presented in section 4.4.5. All the code necessary to re-produce this experiment can be found in the folders 'Test_PSTEP_Harmonic', 'Test_PSTEP_Mixtures', 'Test_PSTEP_Percussive', 'Test_PSTEP_Speech' and 'Test_PSTEP_Vocals' that reside in the 'tests' folder of the GMPC toolbox software package.

Single versus Multi Dictionaries

Another important aspect of GMP, and all sparse decomposition methods for that matter, is the choice of dictionary to use. A lot of research has been devoted to this issue and the testing possibilities are plenty. The main aim of this test is to find out if there is an obvious improvement to signal decomposition metrics when multi-dictionaries are used compared to dictionaries of a single type. Note that the purpose of this test is not to provide a comprehensive statistical outcome but rather explore the behaviour of GMP regarding this issue. In this test all signals mentioned in section 4.4.4 were decomposed using 2048 iterations and the 'lfmp4x' configuration discussed in the previous section. This choice was based on the results of the previous step which indicated that this configuration produces good performance in a reasonable amount of time. Each signal was decomposed against the following dictionaries:

1. *Gabor dictionary*: This dictionary contains Gaussian modulated sine and cosine atoms in five different scales, resulting in 10 atoms per iteration (for details see section 3.6.3).
2. *Hann windowed atoms*: This dictionary contains sine and cosine atoms modulated by a hann window in five different scales, resulting in 10 atoms per iteration (for details see section 3.6.1).
3. *Gammatone dictionary*: This dictionary contains sine and cosine atoms modulated by a gamma-distribution using five scales and with distribution parameters p and β both ranging from 1 until 4, resulting in 160 atoms per iteration (for details see section 3.6.4).
4. *Linear Gaussian chirps dictionary*: This dictionary contains linear chirp (cosine and sine) atoms modulated by a Gaussian window in four different scales, resulting in 8 atoms per iteration. The chirp rate of the selected atoms was calculated in each iteration (for details see section 3.6.6).
5. *Multi-dictionary*: This dictionary contains all aforementioned dictionaries.

Similarly to the previous test, the results produced were evaluated using the metrics described in section 2.3.5 and some of these results are presented and discussed in section 4.4.5. Also, all the code necessary to re-produce this experiment can be found in the folders 'Test_DICT_Harmonic', 'Test_DICT_Mixtures', 'Test_DICT_Percussive', 'Test_DICT_Speech' and 'Test_DICT_Vocals'.

Comparing GMP with MPTK

In this final test, the GMP and MPTK algorithms were compared using the same set of input signals and similar input parameters. Since the architectures of the algorithms are different, using the same input parameters is not possible but the parameters were chosen to be as similar as possible. In particular for the GMP decomposition the 'lfmp4x', 'qislfmp4x' and 'qislfmp8x' were used with an NFFT size of 8192 samples (and $4\times$, $4\times$ and $8\times$ overlap respectively). These configurations were chosen based on their performance, as obtained by previous testing (section 4.4.3). Also the Gabor dictionary mentioned in section 4.4.3 was used. For the MPTK algorithm, three dictionaries of five Gabor blocks each, were used. The block characteristics are presented below, in the following format: [window length, window shift, FFT size] (in samples). These parameters effectively affect the analysis step of the MPTK algorithm. For the MPTK the following dictionaries were used:

1. Gabor dictionary with the following blocks: [8192, 2048, 8192], [4096, 1024, 4096], [2048, 512, 2048], [1024, 256, 1024] and [512, 128, 512]. This dictionary is equivalent to five STFT representations with the aforementioned characteristics all with $4\times$ overlap factor. This configuration produces a similar analysis and amount of atoms as GMP's 'lfmp4x' configuration.

2. Gabor dictionary with the following blocks: [8192, 1024, 8192], [4096, 512, 8192], [2048, 256, 8192], [1024, 128, 8192], [512, 64, 8192]. This configuration produces STFT representations with $8\times$ overlap factor and over-sampled frequency spectra which is similar to GMP's 'qislfmp8x'.

3. Gabor dictionary with the following blocks: [8192, 512, 8192], [4096, 256, 8192], [2048, 128, 8192], [1024, 64, 8192], [512, 32, 8192]. This is similar to the previous configuration but with a $16\times$ overlap factor. Note that this is a much finer analysis than the one obtained by GMP's 'qislfmp8x'.

In a similar fashion to the previous tests, the signals of section 4.4.4 were decomposed into 2048 atoms and the results obtained were evaluated using the metrics defined in 4.4.5. Some of these results are discussed in section 4.4.5 Finally, all the code necessary to re-produce this experiment can be found in the folders 'Test_GMPvsMPTK_Harmonic', 'Test_GMPvsMPTK_Mixtures', 'Test_GMPvsMPTK_Percussive', 'Test_GMPvsMPTK_Speech' and 'Test_GMPvsMPTK_Vocals'.

4.4.4 Performance Testing Examples

The following tables provide details about all signals that were tested. In total forty nine signals were tested. In particular, table 4.1 provides details for seven percussive signals (e.g. bass-drum, snare, hihats, etc.), table 4.2 illustrates eight mixture signals that were used, that is signals with polyphonic content, table 4.3 provides details for twelve harmonic signals, that is signals comprising only one instrument which is harmonic in nature (e.g. guitar, piano, violin, bass etc.), table 4.4 shows details of six vocal signals that were tested (both male and female) and finally table 4.5 details sixteen speech signals including male and female speakers. It should be noted that all test signals are approximately four seconds long and all have the same sample rate of 44100 Hz which is the typical sample rate found on commercially distributed music.

Filename	Ch.	Fs	Description	License
dev1_wdrums_drums_4sec	1	44100	Bass-drum with snare	CC[11]
dev2_wdrums_drums_4sec	1	44100	Bass-drum with snare (with a bit of reverb)	CC
dev2_wdrums_hats_4sec	1	44100	Hi-hat sequence	CC
DrumsElec_01_4sec	1	44100	Bass-drum and snare	CC
DrumsElec_02_4sec	1	44100	Bass-drum, snare and hi-hat sequence	CC
DrumsElec_03_4sec	1	44100	Bass-drum, snare, hih-hats, and toms	CC
FC_4sec	1	44100	Four to the floor bass-drum and some bells	CC

Table 4.1: Percussive signals used in performance testing.

[11] Creative Commons Attribution-NonCommercial-ShareAlike 2.0

Filename	Ch.	Fs	Description	License
dev1_br_1ch_4sec	1	44100	Excerpt from acoustic mixture	CC
dev1_tamy_1ch_4sec	1	44100	Mixture of acoustic guitar and female vocals	CC
dev2_fort_1ch_4sec	1	44100	Mixture of six instruments (bass, drums, claps, samples, violins, vocals)	CC
dev2_nztour_1ch_4sec	1	44100	Mixture of five instruments (bass, drums, guitar, synth, vocals)	CC
Disco_1ch_4sec	1	44100	Mixture of three instruments (bass, drums, synth)	CC
DRM_1ch_4sec	1	44100	Mixture of three instruments (bass, drums, synth)	CC
GTR_WithBass_1ch_4sec	1	44100	Mixture of four instruments (bass, drums, pad, guitar)	CC
Helix_1ch_4sec	1	44100	Mixture of four instruments (bass, drums, synth, vocals)	CC

Table 4.2: Mixture signals used in performance testing.

Filename	Ch.	Fs	Description	License
dev1_nodrums_guitar_4sec	1	44100	Acoustic guitar chords	CC
dev1_nodrums_flute_4sec	1	44100	Flute melody	CC
dev2_nodrums_guitar_4sec	1	44100	Acoustic guitar chords (fast strumming)	CC
dev2_nodrums_bass_4sec	1	44100	Bass guitar sequence	CC
ins_clarinet_4sec	1	44100	Clarinet melody (anechoic recording)	CC
ins_violin_4sec	1	44100	Violin melody (anechoic recording)	CC
Viol_Treble_Front_4sec	1	44100	Viol melody, treble (direct sound recording)	CC
Viol_Bass_Front_4sec	1	44100	Viol melody, bass (direct sound recording)	CC
PC_GtrMain_4sec	1	44100	Acoustic guitar, slow melody (noisy recording)	CC
PC_Bass_4sec	1	44100	Acoustic guitar melody (low tuning)	CC
Piano_LeftCh_Prog_4sec	1	44100	Piano chords progression (recorded in big space)	CC
Piano_LeftCh_4sec	1	44100	Piano solo (single note events, recorded in big space)	CC

Table 4.3: Harmonic signals used in performance testing.

Filename	Ch.	Fs	Description	License
dev1_br_vox_1ch_4sec	1	44100	Male vocals	CC
dev1_tamy_vox_1ch_4sec	1	44100	Female vocals	CC
dev2_ad_vox_1ch_4sec	1	44100	Male vocals	CC
dev2_fort_vox_1ch_4sec	1	44100	Male vocals	CC
dev2_nztour_vox_1ch_4sec	1	44100	Female vocals	CC
ins_soprano_4sec	1	44100	Female soprano	CC

Table 4.4: Vocal signals used in performance testing.

Filename	Ch.	Fs	Description	License
dev1_female1_4sec	1	44100	Female speaker	CC
dev1_female2_4sec	1	44100	Female speaker	CC
dev1_female3_4sec	1	44100	Female speaker	CC
dev1_female4_4sec	1	44100	Female speaker	CC
dev2_female1_4sec	1	44100	Female speaker	CC
dev2_female2_4sec	1	44100	Female speaker	CC
dev2_female3_4sec	1	44100	Female speaker	CC
dev2_female4_4sec	1	44100	Female speaker	CC
dev1_male1_4sec	1	44100	Male speaker	CC
dev1_male2_4sec	1	44100	Male speaker	CC
dev1_male3_4sec	1	44100	Male speaker	CC
dev1_male4_4sec	1	44100	Male speaker	CC
dev2_male1_4sec	1	44100	Male speaker	CC
dev2_male2_4sec	1	44100	Male speaker	CC
dev2_male3_4sec	1	44100	Male speaker	CC
dev2_male4_4sec	1	44100	Male speaker	CC

Table 4.5: Speech signals used in performance testing.

4.4.5 Performance Testing Results & Discussion

This section presents some results that were obtained by the testing procedures explained in section 4.4.3 and briefly discusses their importance in terms of performance evaluation of the produced software. It should be noted that although the decomposition outputs for each test were evaluated using all metrics described in section 2.3.5 it was decided to present only the SRR results. The reason for this is that most evaluation metrics (i.e. EDC, SRR and L2err) provide similar information about the decomposition of a signal but in a different way. For example the EDC is a monotonically decreasing curve showing the reduction of energy of the residual at each iteration step whereas the SRR is a monotonically increasing curve showing the ratio between the energies of the input signal and the residual. Both of these metrics though, provide information about the behaviour of the decomposition and generally speaking the faster these metrics converge, the better the decomposition quality since fast convergence implies that only a few significant atoms have been used in the decomposition. Of course the metrics have to converge to desired values, for example the EDC demonstrates good behaviour if the metric converges to zero whereas the SRR must converge to infinity. One advantage of using the SRR is that it is presented in a logarithmic scale (dB units) and thus differences between different SRR curves are much easier to distinguish. Another advantage is that the SRR metric is invariant to the initial energy of the signal; a 30 dB final SRR value for a particular decomposition would be the same regardless of the initial energy of that signal (if the signal is multiplied for example by a gain factor). Having said that, the SRR curves for each decomposition example are not presented here as they do not illustrate the algorithm's overall behaviour and performance. Instead, box-plots of the final SRR values of each decomposition are presented.

A box-plot (as used in descriptive statistics) is a convenient way to visualise groups of numerical data by partitioning a set of observations into subsets that convey different meaning. Figure 4.7 depicts a typical box-plot. The segmentation of a box-plot is based on quantile values which are the cut points that divide a given data set into equal sized groups.

Typical aspects of a box-plot are the box that indicates the minimum and maximum values of a certain percentage of the data in the set, a band inside the box which indicates the median value of the data set, the whiskers which indicate the minimum and maximum value of the whole data set (although the whiskers could represent other values as well) and outliers which are usually indicated by points extending outside the limits indicated by the whiskers. The results presented here were generated using MATLAB®'s 'boxplot' function.

Depending on the settings used, a box-plot can convey different information. For all figures presented in the following subsections the quantiles of the box-plot were set so that the box region represents 80% of the data, the box band (the line inside the box) indicates the median value of the data, the small rectangular point within each box represents the mean value of the data, the whiskers represent the minimum and maximum values found in the data set and any outliers are represented by the plus symbol. Both the mean and median terms are used to describe the central tendency of a distribution, but in typical box-plot configurations the mean is usually ignored since it can be largely influenced by the presence of outliers. However, the mean is included here since when compared with the median value it can provide further information about the skewness of the distribution. In particular if the mean is the same as the median this indicates that the data is symmetrical (i.e. evenly distributed) around the mean whereas if the mean and median are different implies a skewness in the distribution (when the mean is greater than the median then the distribution is skewed to the left (positively skewed) and the other way around).

Before continuing with the explanation of the results, note that the data presented here are the final SRR values of each decomposition, that is ,each box in the box-plots represents the forty nine (the number of signals tested) final SRR values for each test case (i.e. for each different decomposition configuration as described in the previous sections).

Pre-Processing Step Test Results

This test aims to examine, the effects of the pre-processing step to the decomposition of various audio signals. Figure 4.7 depicts the results for each different pre-processing step configuration in a box-plot format. The first observation that can be made is that the variance (not in a strict statistical sense) of the data is quite large. The minimum and maximum values indicated by the whiskers are approximately 9 dB and 42 dB respectively, across all configurations. Also the majority of the data (80%) range approximately between 12 dB to 28 dB across all configurations. This large spread can be explained by the use of a Gabor dictionary for decomposing all signals. Although the Gabor dictionary has very useful properties, it is not able to represent all types of audio signals equally well. For example, Gabor atoms are expected to correlate well with oscillatory patterns in signals but not transients which are better represented by wavelet-type atoms. Some of the signals tested here are percussive instruments so the Gabor dictionary is expected to "under-perform" in these cases. Other signals that might not be well represented by this dictionary are the mixture signals, especially those with percussive content. It should be noted though that the approximate maximum SRR achieved is 42 dB which is very impressive, considering the length of the input signals ($\approx 4 \cdot 44100 = 176400$ samples) and the number of atoms used for the decomposition (i.e. 2048 iterations). From this figure, we cannot get any more clear information on the performance of the different configurations.

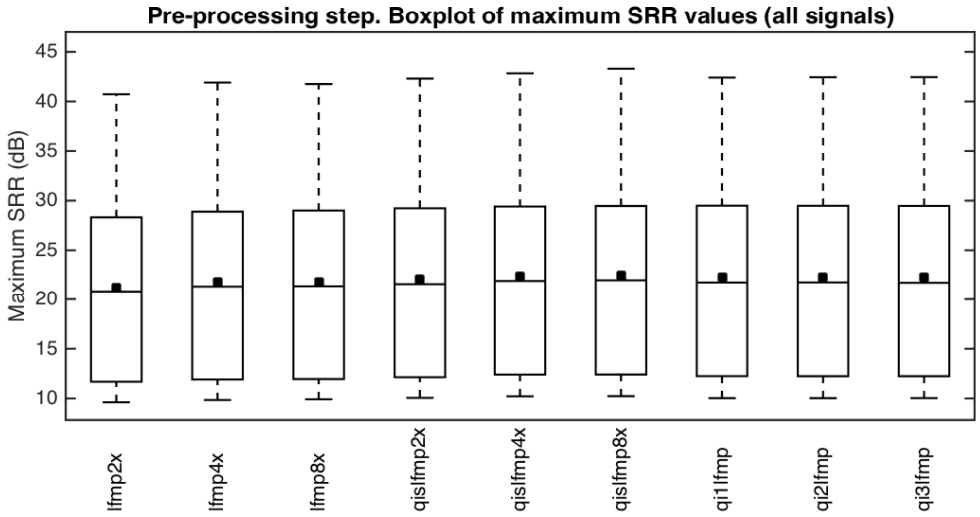


Figure 4.7: SRR values of all test cases.

Figure 4.8 depicts the data of figure 4.7 but zoomed around the median and mean values of the box-plots. Also in this figure the mean values are connected with a line in order to make any variations more visible. Inspecting this figure we can firstly see that the mean and median values of the data are similar which implies that the data are almost evenly distributed around the mean and also the mean being larger than the median implies a positively skewed distribution. Most importantly, it can be seen that both the mean and median SRR values increase with each configuration. In particular the difference between the 'lfmp4x' configuration (which leads to the least accurate analysis) and the 'qislfmp8x' (most accurate analysis) is almost 1.3 dB. Although this is a small difference, it is consistent across all signals (given the even distribution of the data) which implies that the pre-processing step does indeed have an effect to the decomposition of a signal and in particular the better the analysis performed by the pre-processing step the better the final SRR achieved. These results are promising and very important since they verify one of the main design ideas behind GMP which is that of guided selection of atoms at each iteration via the inclusion of a pre-processing step. It should be noted that all pre-processing configurations tested here, improve only on the frequency estimates of the atoms whereas the scale is fixed. In fact, informal experimentation has shown that when the window lengths for the STFT analysis are chosen so that they correspond to the temporal structure of the signal (for example based on tempo information) the results of a decomposition improve. It is thus expected that incorporating this information into the guidance step will improve performance even further. This claim has yet to be verified.

The last three QISTFT -based configurations, 'qi1lfmp', 'qi2lfmpx' and 'qi3lfmp' do produce better results compared to the STFT-based configurations (i.e. 'lfmp2x', 'lfmp4x' and 'lfmp8x') but it seems that more accurate frequency estimation steps do not increase performance. However, it should be noted that the analysis parameters used for these configurations are better suited for speech signals as suggested in [107].

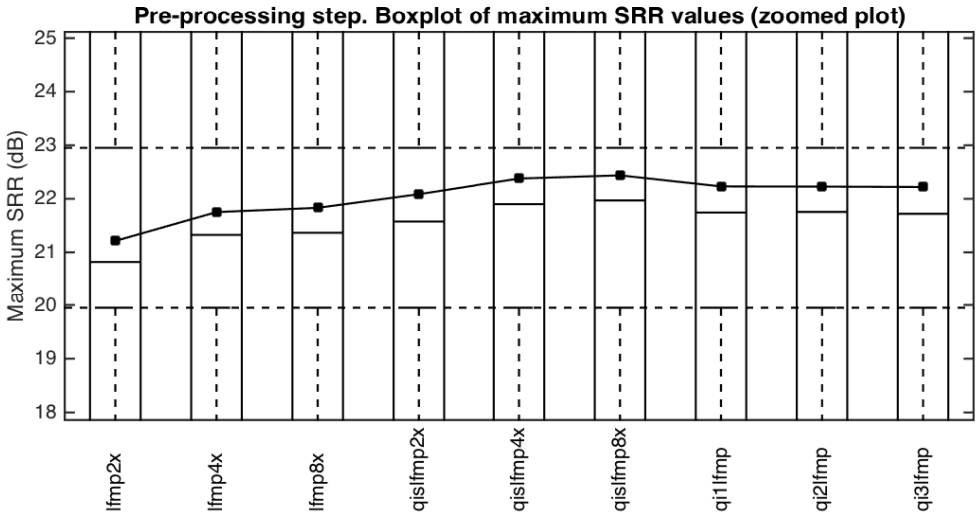


Figure 4.8: SRR values of all test cases.

Figure 4.9 depicts a box-plot containing the average execution times (in seconds) of all signals for each configuration. This figure clearly shows that execution time increases with an increasing analysis accuracy (and thus computational complexity) of the pre-processing step. This result was expected. The last three QISTFT-based configurations have very similar execution times and this indicates that the frequency estimation algorithms, which are executed within the main algorithm of the pre-processing step, do not incur extra computation time.

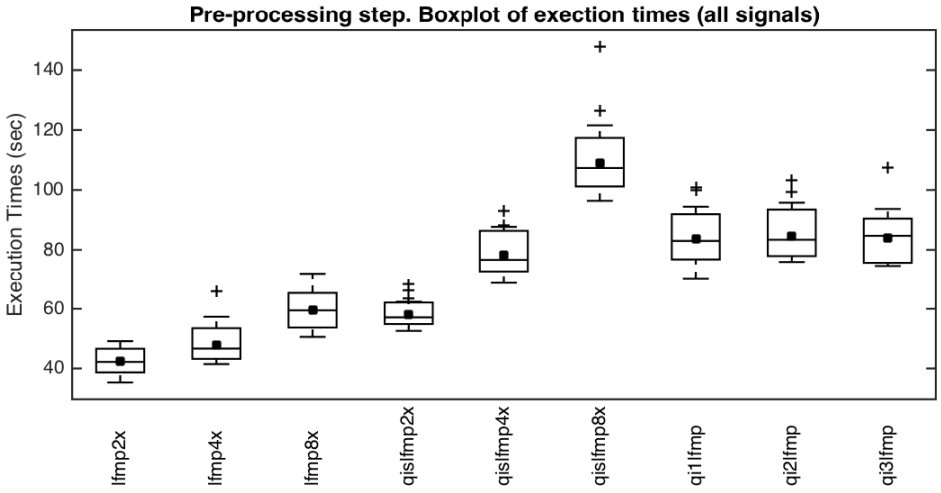


Figure 4.9: Average execution times for all test cases.

Dictionary Testing Results

The aim of this text was to examine the effect of various dictionaries on the decomposition performance of audio signals. Figure 4.10 presents the results of audio signal decompositions against several different dictionaries. By inspecting the figure we can see that the Gabor atoms, hanned windowed atoms, chirp atoms and the multi-dictionary have similar distributions whereas the gamma-tone dictionary produces a distribution with a smaller variance. The mean and the median values are also similar for every case which again implies an almost even distribution of the data around the mean.

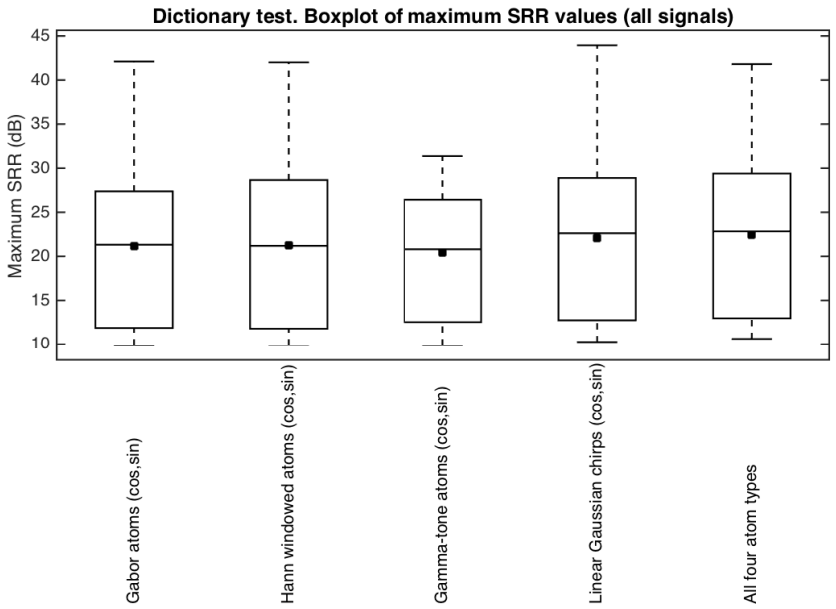


Figure 4.10: SRR values of all test cases.

Figure 4.11 is a zoomed version of figure 4.10 and provides a clearer view on how the median and mean values of the data are affected. In this figure it can be seen that the Gabor atoms and hann windowed atoms produce similar mean and median values which implies that these dictionaries perform similarly. This is not suprising, considering that the only different between these two dictionaries is the amplitude modulation of the underlying sinusoidal atoms. The gamma-tone dictionary seems to perform less well compared to the other dictionaries and the dictionary of Gaussian chirps performs better than all other single-type dictionaries. The multi-type dictionary seems to have overall better performance. Although the differences in the mean and the median values are small, given the almost even distribution of the data around the mean, it can be said that these differences are consistent across all audio signal types. The better performance of the Gaussian chirp dictionary and the multi-type dictionary was expected since the chirp dictionary contains Gabor atoms so it should perform better than the Gabor dictionary and the dictionary of hann windowed atoms and following the same logic the multi-type dictionary should perform better than all other single-type dictionaries. These results are consistent with the main idea of using over-complete dictionaries where a signal is expected to be decomposed better given a large range of bases.

Figure 4.12 shows the average execution times for each dictionary configuration. In this figure, it can be seen that the Gabor and hann windowed atoms take the least amount of time to execute, followed by the dictionary of Gaussian chirps, then the gamma-tone dictionary and finally the multi-type dictionary. These results were expected, when considering the amount of atoms per iteration that were used for each dictionary (10 atoms for Gabor and hann windowed dictionaries, 8 atoms for Gaussian chirp dictionary, 160 atoms for gamma-tone dictionary and 188 atoms for the multi-type dictionary). The Gaussian chirp dictionary although it uses less atoms per iteration compared to the Gabor and hann windowed dictionaries is does require more time to execute. This happens because Gaussian chirps need to be estimated at every iteration thus incurring extra computational cost which leads to increased execution times.

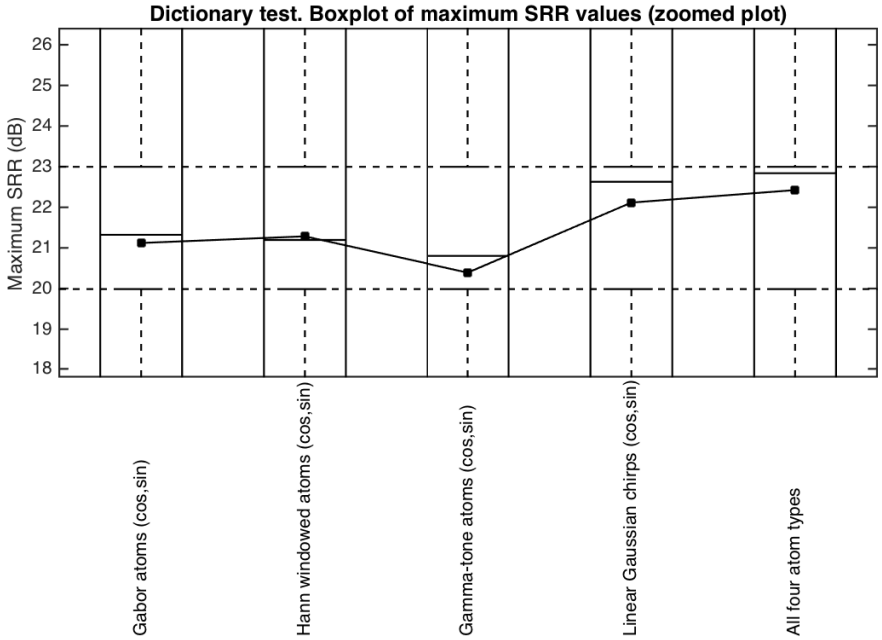


Figure 4.11: SRR values of all test cases.

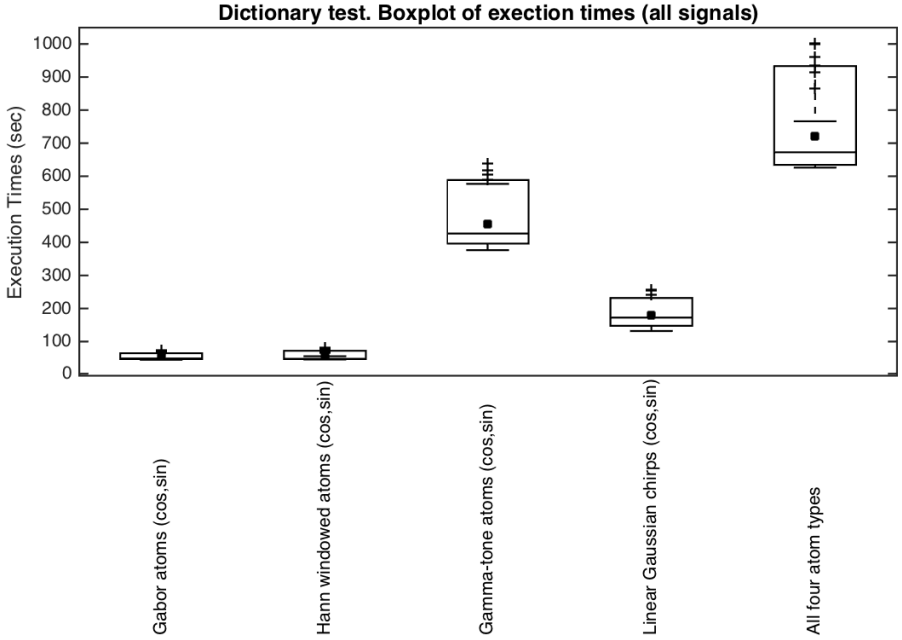


Figure 4.12: Average execution times for all test cases.

The results obtained form this test verify the claim that using an over-complete dictionary that is a union of several bases leads to better decomposition performance. However this increase in performance comes with an increase in computational cost and execution time so the question arises of how to implement fast algorithms that can take advantage of such over-complete dictionaries.

GMP versus MPTK Comparison Results

The main goal of this test was to compare the GMP algorithm against a state of the art MP implementation in order to verify that the results obtained by GMP are meaningful and that GMP provides a good alternative to already available algorithms. Figure 4.13 illustrates the results obtained by the decomposition of the signals mentioned in section 4.4.4 using the configurations described in section 4.4.3. By observing the figure it can be seen that the results produced by all configurations have similar distributions, with the majority of the data lying in the range between 11 dB and 28 dB (i.e. data within each box region). This result alone is encouraging since it shows that GMP does indeed provide similar results (on average) to well established MP implementations.

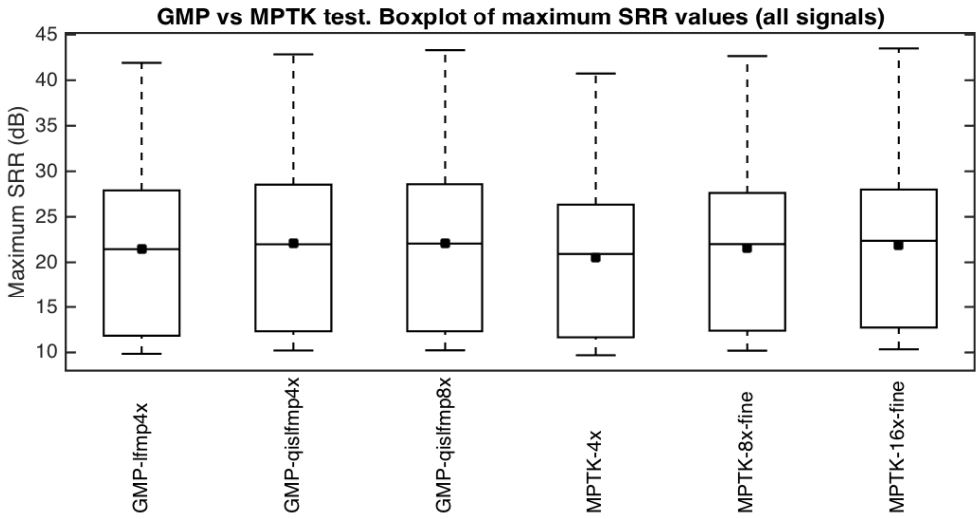


Figure 4.13: SRR values of all test cases.

Figure 4.14 shows a zoomed version of the data around the mean and median values. This figure shows that, on average, GMP performs at least as good as MPTK. In particular, by comparing similar configurations, it can be seen that GMP's 'lfmp4x' performs better than MPTK's 'MPTK-4x' configuration whereas GMP's 'qislfmp4x' and 'qislfmp8x' configurations perform similarly to MPTK's 'MPTK-8x-fine' and 'MPTK-16x-fine'. An interesting pint to make is that performance is improved on average by increasing the accuracy of the analysis in both algorithms. This behavior was expected since a more accurate analysis leads to larger dictionaries and therefore better decompositions.

Figure 4.15 shows the average execution times (in seconds) for each configuration. From this figure it is obvious that MPTK is faster than GMP. In particular an approximately 30 second difference can be seen for each configuration pair (i.e. 'lfmp4x' vs 'MPTK-4x', 'qislfmp4x' vs 'MPTK-8x-fine' and 'qislfmp8x' vs 'MPTK-16-fine'). This result was expected because MPTK is implemented in C++ which is a compiled language and is expected to produce faster code than MATLAB® which is an interpreted language. MPTK being faster in these experiments does not mean that GMP is slow but it does show that the current MATLAB® implementation of GMP is slow. Implementing GMP in C++ would definitely improve its execution times. In fact, the author has produced unofficial, experimental MATLAB® versions of GMP exist that are capable of

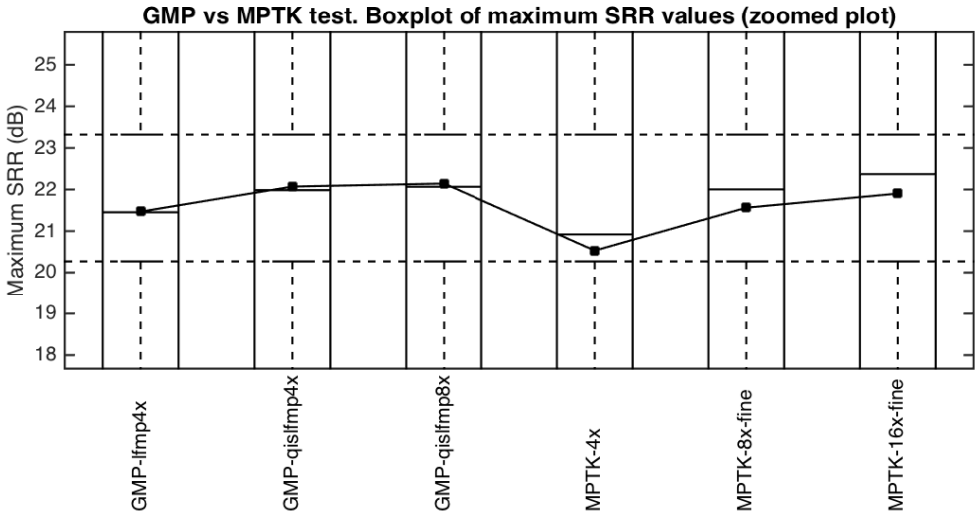


Figure 4.14: SRR values of all test cases.

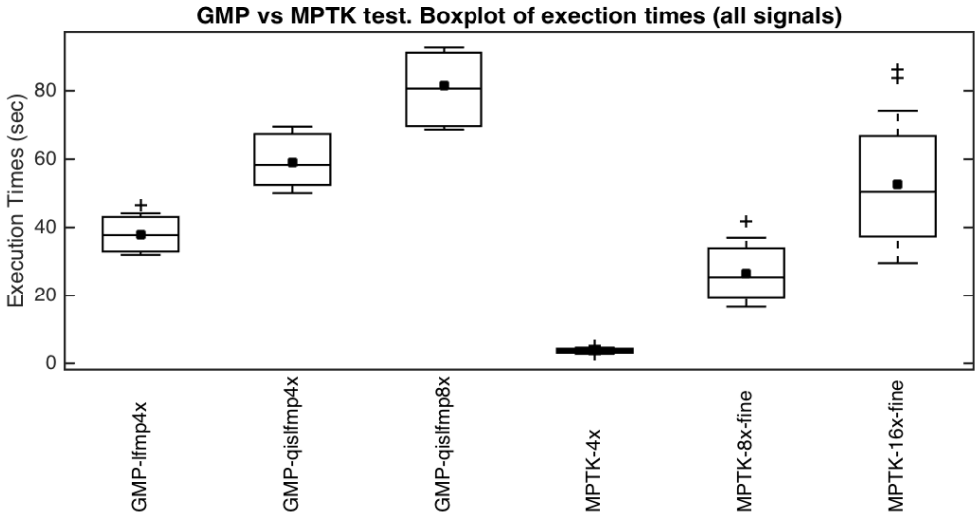


Figure 4.15: Average execution times for all test cases.

real-time audio processing which indicates that an appropriately optimised GMP implementation can serve as an attractive and useful alternative to currently available MP implementations.

Chapter 5

Application of GMP in Sound Source Separation

In chapter 3, GMP was introduced and several aspects of the algorithm were described in detail. GMP is a generic algorithm based on matching pursuit and can be used in many audio signal processing tasks. The aim of this chapter is to demonstrate the applicability of GMP in sound source separation problems with emphasis given on linearly mixed stereophonic mixtures as produced either by intensity stereo microphone techniques, or in a studio setting using a mixing desk or dedicated software with panning control.

Most BSS techniques are based on the signal properties of independence and sparsity. In particular ICA-based techniques rely on the assumption that the source signals within the mixture are statistically independent and identically distributed and source separation is achieved by the minimisation of a certain dependency measure between the estimated sources [1]. This assumption is sometimes too strict for musical signals which generally are highly dependent, especially over short time durations [1]. Also ICA-based techniques are tailored for determined and over-determined source separation cases (same number or more mixture signals than sources respectively), but in music the problem is almost always under-determined (fewer mixture signals than sources).

For under-determined source separation cases, the SCA-based family of techniques is better suited. The main assumption here is that the source signals can be sparsely represented in a suitable domain which implies that the sources are expected to have disjoint (or almost disjoint) supports in that domain. In this case source separation is achieved by identifying and estimating the mixing directions of the sources in the new domain, a process known as mixing matrix estimation and then by clustering the coefficients of the representation based on the estimated directions. Since the sources are assumed to have disjoint supports in the domain, the energy of each coefficient is usually attributed to the activation of a single source or in other words, it is assumed that each time-frequency atom pair (for left and right channels) in the new domain representation belongs to a single source which leads to the so called "binary masking" techniques. Application of the inverse transform, or the synthesis step of the decomposition onto the clustered coefficients yields the estimated separated sources in the time domain. This process was described in section 1.4.3. Of course in order to take advantage of this approach, a stereo mixture is essential but this is

not a 'harsh' requirement since most commercially produced music is in stereo format. Having said that, a sparse representation usually implies an adaptive representation, which means that the selected atoms are well adapted to the signal's inner structures and by taking advantage of this 'adaptivity' it is possible to use SCA-based algorithms when a single channel mixture signal is considered, as for example in [168] and [169].

Considering the steps of a SCA-based system, it should be clear that the first step, that is the front-end to the system, plays a crucial role in the subsequent steps of mixing matrix estimation and clustering of the expansion coefficients. Traditionally the STFT representation and sometimes the MDCT are employed as the front-end mainly due to their computational speed but as already explained and demonstrated throughout chapters 2 and 3 atomic decompositions provide adaptive and sparser representations than the STFT. Therefore it makes sense to use such a decomposition as the front-end. Although there are several choices for such an algorithm, MP is one of the most attractive options since it is relatively fast, provides 'good' sparse representations but most importantly is a very flexible algorithm. Assuming a known (or already estimated) mixing matrix then a typical STFT-based SCA system will first need to transform the signal into the time-frequency domain and then in separate steps perform clustering of the coefficients and inverse transform these to obtain the estimated sources. Using MP all these steps are reduced to one, where the decomposition of the signal, clustering and reconstruction of the estimated sources occur together at each iteration of the algorithm. This approach provides a way to make decisions about the separation of the sources at the decomposition level, before obtaining a full set of coefficients as with the STFT which can lead to some very interesting source separation approaches, especially when prior information about the signal is available (e.g mixing matrix, score information, number and type of sources). Of course it is possible to follow the typical approach of decomposing the mixture first before clustering the coefficients as is done for example in [166].

The rest of the chapter continues with a discussion of mixing matrix estimation followed by several approaches to source separation using GMP. Although the main source separation experiments performed in this work assume a known mixing matrix, its estimation is vital for the correct operation of a SCA-based system and it would be beneficial to know how one could actually get a mixing matrix estimate and also how GMP could be employed for such a task.

5.1 Mixing Matrix Estimation

The second step of a SCA-based source separation system is to estimate the mixing matrix $\mathbf{A}$ using the scatter plot of the expansion coefficients $(c_{x_1}(q), c_{x_2}(q))$, $q = 1, \ldots, Q$ obtained by the sparse decomposition of the input mixture. Most common approaches are geometrical in nature and use various clustering algorithms on the scatter plot of the coefficients. These techniques can be used in practice regardless of the number of sensors but do require at least two sensors, which is the case for stereophonic music. Plot $a)$ of figure 5.1 depicts the scatter plot of the Fourier coefficients of a stereophonic mixture comprising three sources, linearly mixed together. Observing the scatter plot it can be clearly seen that the data points lie mainly around three distinguishable lines which correspond to the directions of the columns of the mixing matrix.

Another way to represent the same information is to convert the Cartesian coordinates of each data point into polar coordinates using the following equations:

$$\rho(q) = \sqrt{|c_{x_1}(q)|^2 + |c_{x_2}(q)|^2} \tag{5.1}$$

$$\theta(q) = \arctan\left(\frac{c_{x_2}(q)}{c_{x_1}(q)}\right) \tag{5.2}$$

$$\text{for } q = 1, \ldots, Q$$

where $\rho(q)$ is the distance of a point from the origin (i.e. point $(0, 0)$), $\theta(q)$ is the angle between the x-axis and the line connecting the origin and the point $(c_{x_1}(q), c_{x_2}(q))$ with $0 \leq \theta(q) < \pi$ and M is the number of coefficients. The scatter plot of $\theta(q)$ and $\rho(q)$ is depicted on plot $b)$ of figure 5.1. The simplest approach would be to produce a histogram of all $\theta(q)$ values and estimate the directions of the mixing matrix by using thresholding. The histogram for this particular example is depicted in figure 5.2. As can be seen from the figure only one peak is clearly visible and a method based on simple thresholding would fail to detect all three directions. This happens because most of the coefficients have low amplitude (visible on plot $b)$ of figure 5.1) and for these coefficients equation (5.2) does not produce directions which are representative of the column directions of the mixing matrix, thus having a flattening effect on the histogram [1].

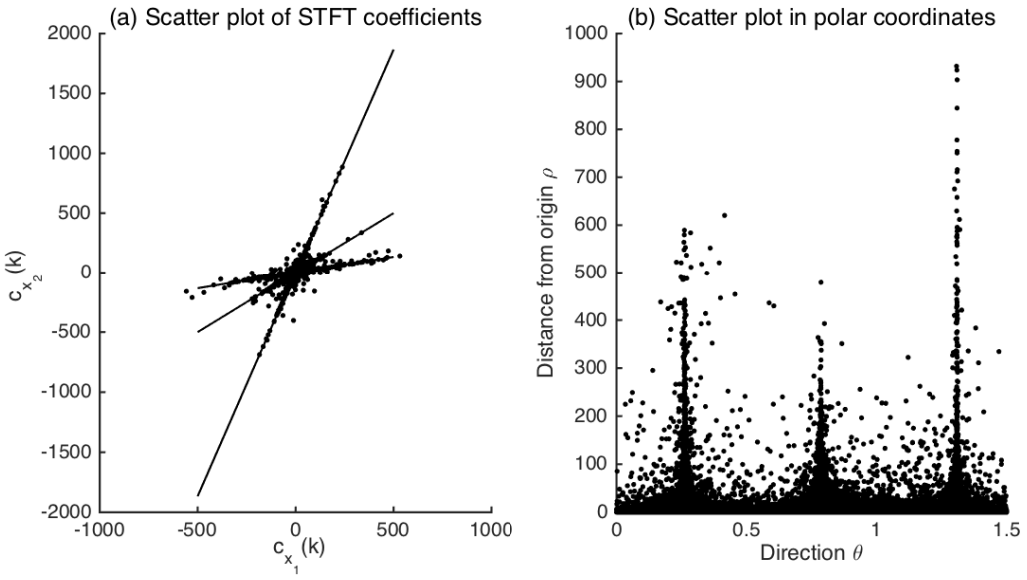


Figure 5.1: Scatter plots of Cartesian and polar coordinates of STFT coefficients. The analysed signal is a stereophonic mixture (J=2) comprising three sources (P=3). The scatter plots are the global STFT coefficients of the two mixture channels. (a) shows the Cartesian coordinates and (b) the polar coordinates.

One way to overcome this problem is to use a weighted histogram such as in [53]. The main idea is to calculate the histogram based only on those coefficients that have significant amplitudes. The authors in [53] make use of a "potential function" which gives different weights to each data point in the scatter plot depending on their distance from the origin. This produces a weighted histogram where the peaks of the directions are more visible and can be more easily detected by an automated system. This way of estimating the mixing matrix is based on the global scatter plots of the decomposition coefficients and is used in algorithms like DUET [58]. Using clustering algorithms on the global scatter plots is not always applicable, and can be difficult especially in situations where the source signals are not sufficiently disjoint in the transformed domain, which might be the case when instruments play in time and in harmony or even when the sources are

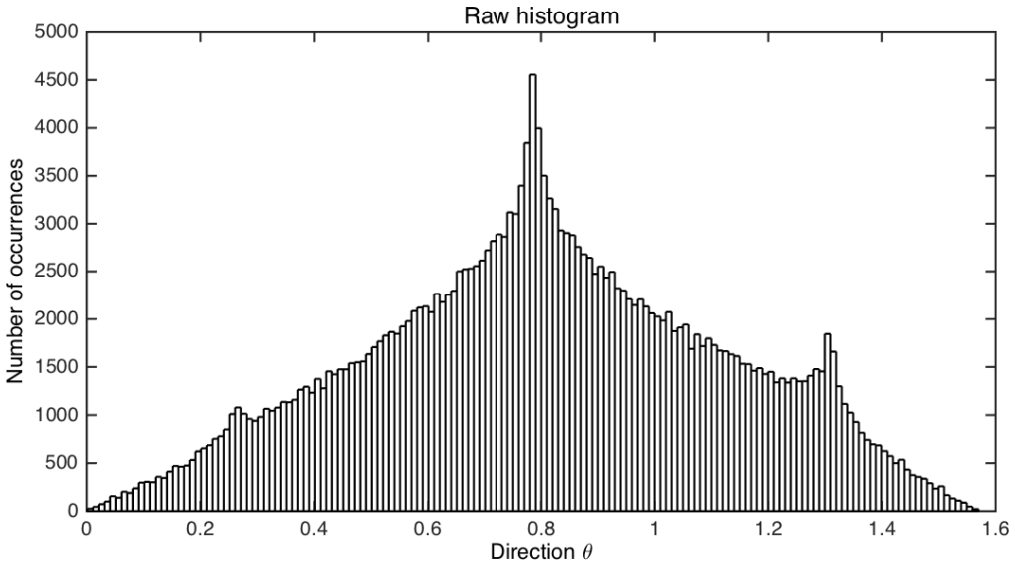


Figure 5.2: Mixing matrix estimation using raw histogram. This is the histogram of the direction coordinate of the polar scatter plot. It shows the frequency of occurrences of each angle $\theta(m)$. Only a single peak is clearly visible.

mixed with diverse intensities [1]. In such cases it is difficult to establish which atoms in the decomposition belong to which sources.

Other approaches to mixing matrix estimation make use of local scatter plots in the time-frequency domain. Such approaches are based on the assumption that at a particular time-frequency region only one source will be active (or will contribute significantly more than other sources), thus when calculating the scatter plot for this region the corresponding atoms will all lie on one line which corresponds to the column direction of the mixing matrix for that source. If such time-frequency regions can be detected then estimation of the mixing matrix is possible. This is depicted in figure 5.3 where the plots in the top row show the spectrograms of the left and right channels of the stereophonic mixture that was used in the previous figures and the scatter plots in the bottom row are produced by the regions seen in the spectrogram plots (black annotated boxes). From the scatter plots we can see that in region A only once source is active and estimation of the direction is possible whereas in region B multiple sources are active simultaneously and directions cannot be inferred. This is the main idea behind blind source separation systems like TIFROM [198, 20, 21] and DEMIX [76, 199, 200].

Although the above examples make use of the STFT representations of the mixtures, the described techniques can be easily applied to the expansion coefficients of a GMP decomposition, or any MP decomposition for that matter. Of course there are many others approaches to mixing matrix estimation but cataloguing all available techniques is beyond the scope of this work. The main purpose of this section is to demonstrate the ability to estimate a mixing matrix. Mixing matrix estimation is usually treated as a separate problem to source separation and given the wealth of already available techniques one can assume that this information is relatively easy to obtain.

5.2 Source Separation Using A Known Mixing Matrix

When dealing with linear instantaneous stereo mixtures, as described in section 1.3, then the intensity differences between the two channels can be used for basic parameter estimation of the

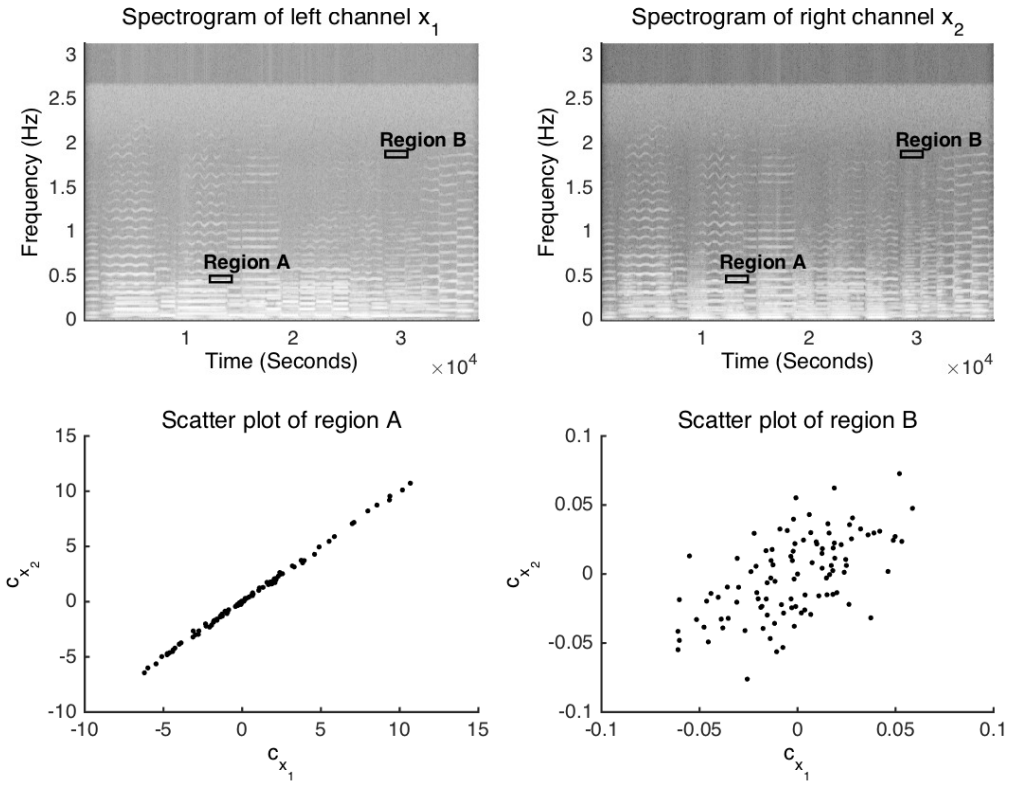


Figure 5.3: Mixing matrix estimation using local scatter plots. The top row depicts the spectrograms of a stereophonic mixture comprising three sources (left and right channels). The bottom row depicts the scatter plots calculated from the region that are highlighted in the spectrogram plots (black annotated boxes). The scatter plot for region A clearly shows one dominant source, and all the data points lie along the line which corresponds to the directions of one column of the mixing matrix. The scatter plot of region B gives us no information on mixing matrix directions since many sources are active in this region.

directional of arrival (DOA) of each time-frequency atom pair (i.e. left/right channel atoms). Assuming that $\mathbf{C_x}$ is the matrix holding the expansion coefficients of a GMP decomposition then the estimated directions of arrival can be calculated by:

$$\mathbf{\Theta} = \arctan\left(\frac{|\mathbf{C}_{\mathbf{x}_2}|}{|\mathbf{C}_{\mathbf{x}_1}|}\right) \tag{5.3}$$

where $\mathbf{C}_{\mathbf{x}_1}$ and $\mathbf{C}_{\mathbf{x}_2}$ are the expansion coefficients of the left and right mixture channels respectively and $\mathbf{\Theta}$, $0 \leq \mathbf{\Theta} \leq \frac{\pi}{2}$ contains the estimated directions of arrival. This equation follows directly from the instantaneous mixing model described by equation (1.9a).

Another useful parameter, used in many source separation systems, is the phase offset between the time-frequency atoms of the mixture channels. When the front-end of the system is the STFT representation of the mixture channels (i.e. $\mathbf{C}$ contains the complex coefficients of the STFT output), then the phase offset can be obtained by:

$$\varphi = \angle\left(\frac{\mathbf{C}_{\mathbf{x}_1}}{\mathbf{C}_{\mathbf{x}_2}}\right) \tag{5.4}$$

When the front-end is the MP algorithm then in practice, matrix $\mathbf{C}$ usually contains the magnitudes of the expansion coefficients but these are associated with further parameters relating to the extracted atoms such as the scale, location, frequency, and phase of the atoms. When the

phase of the atoms is available then the phase offset parameter can be estimated by:

$$\boldsymbol{\varphi} = \mathbf{C}_{\mathbf{x}_1}^{(\phi_1, f_1)} - \mathbf{C}_{\mathbf{x}_2}^{(\phi_2, f_2)} \tag{5.5}$$

where $\mathbf{C}_{\mathbf{x}_1}^{(\phi_1, f_1)}$ and $\mathbf{C}_{\mathbf{x}_2}^{(\phi_2, f_2)}$ denote the phase (ϕ) and (normalised) frequency (f) parameters associated with each time-frequency atom and $f_1 == f_2$ in order for the phase offset to be meaningful. In GMP the phase parameter is implied by the extracted atom location, which will be denoted by u so the phase offset can be obtained by:

$$\mathbf{u}_{offset} = \mathbf{C}_{\mathbf{x}_1}^{(u_1, f_1)} - \mathbf{C}_{\mathbf{x}_2}^{(u_2, f_2)} \tag{5.6}$$

where u_1 and u_2 are the locations of the extracted time-frequency atoms from the left and right mixture signals respectively and $\mathbf{u}_{offset}$ is a vector containing the location differences which can be converted to phase by:

$$\begin{aligned} \boldsymbol{\varphi} &= \mathrm{princarg}(2\,\pi\,f\,n_0) \\ n_0 &= \frac{N_d}{2} - u \end{aligned} \tag{5.7}$$

where N_d, u and f are the atom's length in samples, extracted location and normalised frequency respectively, n_0 the time reference in samples and the function '$\mathrm{princarg}$' returns the principle argument of the calculated phase. Note that the time reference n_0 is expressed in terms of half the atom's length and is implementation specific. When using atom parameters obtained by MP implementations other than GMP then equations (5.6) and (5.7) will most probably need to be adapted accordingly.

Usually the phase offset parameter is used for estimating the time delay introduced by the mixing model, but it can be beneficial even when the sources are mixed using simple linear panning and no delay between the channels. An interesting observation regarding the phase offset parameter is that a zero phase offset between a pair of time-frequency atoms is a strong indicator that only a single source is active whereas a non-zero phase offset indicates that more than one sources is active. This observation is studied in detail in [17]. Another measure that can offer similar information is the inter-channel phase coherence proposed in [201] but will not be pursued any further in this work.

When using the STFT as the front-end, this interesting fact about the phase offset parameter can be exploited so that the energy of atoms with non-zero phase offset is shared among multiple sources rather than a single source, thus correcting the main problem introduced by binary masking. In this case the atoms produced by the STFT have a 'fixed' phase in the sense that it is computed as part of the FFT of each data frame and so the phase offset parameter is estimated. When using MP as the front-end the phase offset between the atoms can be enforced rather than estimated. To explain further consider the stereo dictionary used in stereo matching pursuit (SMP) proposed in [162]:

$$\mathcal{D}_{stereo} \coloneqq \{\cos(\theta)d(t),\, \sin(\theta)d(t-\tau)\} \tag{5.8}$$

where $d(t) \in \mathcal{D}$ is a mono atom, $0 \leq \theta \leq \frac{\pi}{2}$ the panning parameter and $\tau \in \mathbb{R}$ is a delay parameter with $|\tau| \leq \tau_{max}$ where τ_{max} is the maximum delay between the channels. By setting the delay

parameter τ to zero then a decomposition of a signal using $\mathcal{D}_{stereo}$ will use atoms with zero phase offset. In this case the phase offset parameter is enforced. In GMP the dictionary creation and atom selection steps are different from the standard MP (as used in SMP) but a similar effect can be achieved by extracting the left and right channel atoms from the same location, that is enforcing $u_1 == u_2$ (and of course $f_1 == f_2$) in which case equation (5.6) will produce a zero location difference between the atoms, which by using equation (5.7) is translated to a zero phase offset.

In the following subsections four different algorithms are proposed that try to take advantage of the instantaneous linear mixing model, a known mixing matrix and the zero phase offset observation described earlier. In the context of the GMP framework these algorithms are essentially different atom selection functions that can be used as drop-in replacements of the standard atom selection step 5 of algorithm 4. As such they all operate on the coefficient matrix $\mathbf{C}_i$ which is produced by the cross-correlation between the residual and the atoms of the mini dictionary $\mathbf{D}_i$, where i denotes the iteration step. For the following algorithms the coefficient matrix $\mathbf{C}_i$ is an $N \times Q \times J$ matrix where N is the maximum length of the atoms, Q is the number of atoms in $\mathbf{D}_i$ for the i^{th} iteration step and J is the number of channels, which for a stereophonic mixture is $J = 2$. Also $\boldsymbol{\theta} \in \mathbf{R}^P$ is a vector of known directions with P indicating the number of sources in the mixture.

5.2.1 Algorithm SBASS 1

This first atom selection criterion is a simple implementation that takes advantage of the zero phase offset indicator. In this algorithm the atoms are selected so that the difference between the estimated atom direction and the known mixing matrix direction is as small as possible. The main idea is that if a pair of coefficients is close to a known direction then there is a higher chance that the corresponding atoms belong to the source indicated by that direction. Also since a zero phase offset is enforced (i.e. by extracting the atoms from the same location within the signal) then it can be assumed that the selected atom pair belongs to a single source and therefore all energy is assigned to a single source.

First of all the estimated directions $\boldsymbol{\Theta}$ are calculated using equation (5.3). Note that $\boldsymbol{\Theta} \in \mathbf{R}^{N \times M}$. Then a distance matrix is calculated by:

$$\mathbf{V}_{n,q,p} = |\boldsymbol{\Theta}_{n,q} - \boldsymbol{\theta}_p| \tag{5.9}$$
$$\forall\, n \in \{1, \ldots, N\}, \quad \forall\, q \in \{1, \ldots, Q\}, \quad \forall\, p \in \{1, \ldots, P\}$$

where $\mathbf{V} \in \mathbf{R}^{N \times M \times P}$ is a matrix holding the distances between each estimated direction in $\boldsymbol{\Theta}$ and the known directions $\boldsymbol{\theta}$. The indices of the minimum value in $\mathbf{V}$ indicate the extraction location in the signal (in samples), the atom to choose from the dictionary (which implies the extra atom parameters such as type, scale, frequency etc.) along with the expansion coefficients (obtained from $\mathbf{C}$) and finally the source number the atom belongs to.

A potential problem with this particular atom selection function is that the GMP algorithm might fail to converge in an acceptable number of iterations and that is because the energy of the selected atoms is not taken into account.

5.2.2 Algorithm SBASS 2

This selection criterion is an extension of the previous one where the energy of the selected atoms is taken into consideration. In particular, at each iteration, the combined squared magnitude of the coefficients is calculated by:

$$\mathbf{E}_c = \sum_{j=1}^{J=2} |\mathbf{C}_{n,q,j}|^2 \tag{5.10}$$
$$\forall\, n \in \{1, \ldots, N\} \text{ and } \forall\, q \in \{1, \ldots, Q\}$$

Then an energy weighted distance measure is calculated as:

$$\mathbf{V}_{ew} = \mathbf{E}_c \oslash \mathbf{V}_p \tag{5.11}$$
$$\forall\, p \in \{1, \ldots, P\}$$

where $\mathbf{V}_p$ is the p^{th} sub-matrix of $\mathbf{V}$ and $\mathbf{V}_{ew} \in \mathbf{R}^{N \times Q \times P}$. In this case the indices of the maximum element in $\mathbf{V}_{ew}$ indicate the atoms extraction location within the signal, the index of the atoms in the dictionary $\mathbf{D}_i$ (which implies the extra atom parameters) and the source number the atoms belongs to. The expansion coefficients are obtained by the elements of the coefficient matrix $\mathbf{C}$ with the indices extracted by the previous step. In this atom selection function, the selected pair of expansion coefficients is close to a known direction and the energy weighted distance measure ensures that the selected coefficient pair is the 'best' in an energy sense.

5.2.3 Algorithm SBASS 3

This atom selection function takes advantage of the way the values of the columns in $\mathbf{\Theta}$ are distributed. The algorithm starts by choosing the 'best' histogram for obtaining the expansion coefficients. The choice is based on the shape of the histograms obtained from $\mathbf{\Theta}$. In particular histograms with values concentrated on one direction only are favoured. The idea is that if all or most of the estimated directions of a particular column of $\mathbf{\Theta}$ are clustered around one direction only then this is a strong indication that the corresponding atoms belong to the source indicated by that direction. The 'best' histogram h is selected as follows:

$$h = \max \left(\frac{\sum_{n=1}^{N} \mathbf{M}_{n,q}}{\min \left(\frac{\sum_{n=1}^{N} |\mathbf{\Theta}_{n,q} - \boldsymbol{\theta}_p|}{N} \right)} \right) \tag{5.12}$$
$$\forall q \in \{1..Q\}, \quad \forall p \in \{1..P\}$$

The selected histogram indicates the atom index from which the extra atom parameters can be obtained. After selecting the histogram to operate upon then equations (5.9), (5.10) and (5.11) are used with fixed $q = h$ to obtain $\mathbf{V}_{ew}$ which now becomes a vector of length N. The index corresponding to the maximum value of $\mathbf{V}_{ew}$ indicates the atom extraction location (in samples). Finally using the found location and h the expansion coefficients can be obtained from $\mathbf{C}$. Note that equation (5.12) was obtained empirically.

5.2.4 Algorithm SBASS 4

This atom selection function is similar to the one used in the demixing pursuit algorithm (described in section 2.4.2) but adapted to work in GMP. In particular, let $\mathbf{A} \in \mathbf{R}^{J \times P}$ be the mixing matrix and $\mathbf{a}_p \in \mathbf{R}^J$ be the p^{th} column of $\mathbf{A}$. Also let $\mathbf{M} \in \mathbf{R}^{N \times Q \times J}$ be the magnitudes of the expansion coefficients in matrix $\mathbf{C}$:

$$\mathbf{M} = |\mathbf{C}| \tag{5.13}$$

and let $\mathbf{M}_q \in \mathbf{R}^{N \times J}$ be the q^{th} sub-matrix of $\mathbf{M}$. Then at each iteration the following dot product is calculated:

$$\mathbf{L} = \mathbf{M}_q \mathbf{a}_p \tag{5.14}$$

$$\forall q \in \{1, \ldots, Q\}, \quad \forall p \in \{1, \ldots, P\} \tag{5.15}$$

where $\mathbf{L} \in \mathbf{R}^{N \times PQ}$ contains the dot products between each coefficient pair in the coefficient matrix and each column in the mixing matrix. The indices of the maximum element in $\mathbf{L}$ can be used to obtain the parameters of the atoms to be extracted.

5.3 Source Separation Using A Gaussian Mixture Model (GMM)

In the previous section all atom selection functions that were described, attributed the energy of each left/right pair of expansion coefficients to a single source only. This was due to the assumption that the sources have disjoint supports in the new domain and as such it is expected that each time frequency point is activated by a single source only, a property also known as W-disjoint orthogonality [58] (WDO). This assumption leads to the so called "binary masks" which can be thought of as 'spatial' windows applied to the histogram of the estimated DOA as calculated by equation (5.3). When a single source is mixed in stereo and the mixing direction is known then the estimated DOA of all frequency components will be the same and will correspond to the single mixing matrix direction. When dealing with musical mixtures of multiple sources though, then the WDO assumption is often violated since musicians usually play at the same tempo and the sound of each instrument (source) is harmonically related, therefore there is significant overlap between the sources in the time-frequency domain. In this case, the application of binary-masks will lead to artifacts in the reconstructed separated sources. Also the presence of noise and estimation errors add to the complexity of the separation problem and affect the quality of the separated sources.

A solution to the above problem is the use of non-binary, that is continuous weighting masks, where the energy of a coefficient pair can be attributed to more than one source. One way to introduce such masks is to use a mixture model to describe the histogram of the estimated DOA. In short, a mixture model consists of random variables assumed to be distributed in a mixture of P components that follow the same distribution. A common assumption in sound source separation is that the estimated DOA of a source follows a Gaussian distribution with a mean of θ (i.e. centred around θ) and some variance σ. In case of multiple K sources then a mixture model of order-K can be introduced and when the underlying distributions are assumed to be Gaussian

then this model becomes a Gaussian mixture model of order-K. Such a model can be described mathematically as [103]:

$$P_K(\theta \mid \Gamma) = \sum_{k=1}^{K} w_k \phi_k(\theta \mid \mu_k, \sigma_k) \tag{5.16}$$

$$\text{with } w_k \geq 0 \text{ and } \sum_{k=1}^{K} w_k = 1$$

where Γ is a set of parameter vectors $\gamma_k = (w_k, \mu_k, \sigma_k), \quad k = 1 \ldots K$ that completely describes the mixture model. Each parameter vector contains the weight w_k, mean μ_k and variance σ_k of the k^{th} Gaussian component in the mixture. A Gaussian component can be described mathematically as [103]:

$$\phi_k(\theta \mid \mu_k, \sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(\frac{(\theta - \mu_k)^2}{2\sigma_k^2}\right) \tag{5.17}$$

The main goal of the GMM approach to source separation is to try and fit the histogram of the estimated DOA to the model described by equations (5.16) and (5.17), that is try to estimate the set of parameters Γ that best describe the observed data. The estimated parameter vectors γ_k essentially describe the 'spatial' windows which can then be used to cluster the expansion coefficient pairs indicated by the histogram of the estimated DOA.

The first step in a GMM source separation approach is to calculate the estimated DOA from the coefficients of a transform or a decomposition using equation (5.2). When GMP is used as the front-end, then a full set of coefficients is required before estimating the DOA which is in contrast to the algorithms described in section 5.2 where the clustering and separation of the sources occur at each iteration of the algorithm. In this case, in order for the estimation of DOA to be possible the expansion coefficients must correspond to a pair of atoms taken from a stereo dictionary which means that the extracted atoms must occur at the same time support of the signal. In GMP this can be achieved by employing a weakly-joint or joint multi-channel processing strategy as described in section 3.5.

The next step is to obtain the histogram of the estimated DOA, which will be denoted as $h(\theta)$. Figure 5.4 depicts such a histogram obtained from a linearly mixed stereo signal, comprising four sources equidistantly spaced in the front quadrant with $\boldsymbol{\theta} \approx (0.19, 0.58, 0.98, 1.37)$ radians. In this histogram the number of visible peaks indicate the number of sources in the mixture and the abscissa of each peak indicates the spatial location of each source. Another histogram of a mixture of three sources is also depicted in figure 5.2 but in this case the peaks are not so visible. In cases where the peaks cannot be distinguished, a smoothed version of the histogram is used and this can be achieved by any of the methods described in section 5.1. It is interesting to observe the histogram of the four sources example as produced by the STFT coefficients. This is depicted in figure 5.5 where the peaks are not easily distinguishable. The GMP-based histogram is much clearer and this is due to the sparser representation obtained by the GMP decomposition. That is the GMP produces only a few coefficients of significant magnitude whereas the STFT produces coefficients, many of which have negligible energy, which contribute to the flattening effect of the histogram of the estimated DOA. This is yet another benefit of using GMP (or any MP decomposition for that matter) as the front-end to a SCA-based source separation system.

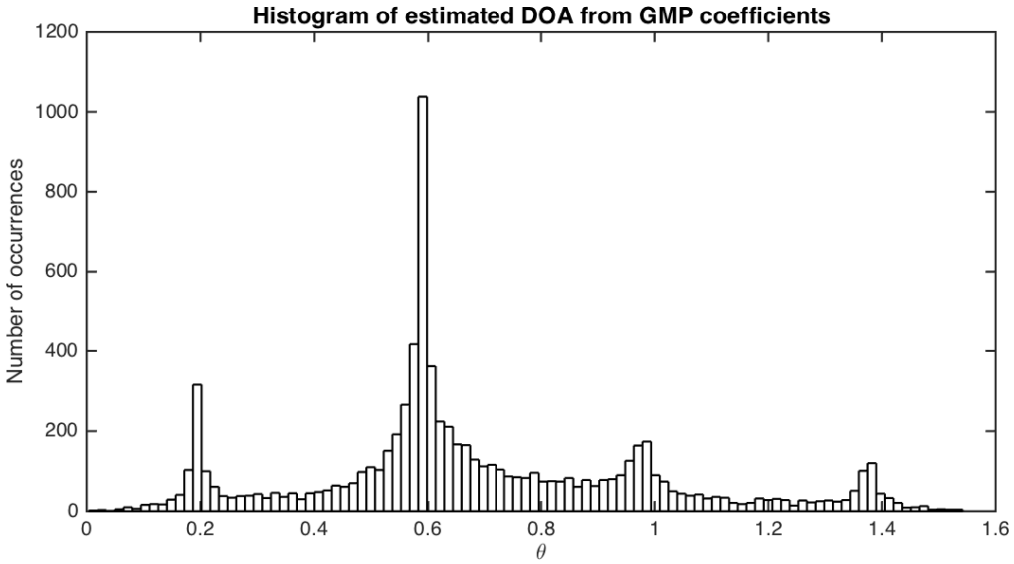


Figure 5.4: Histogram of DOA estimates from GMP coefficients.

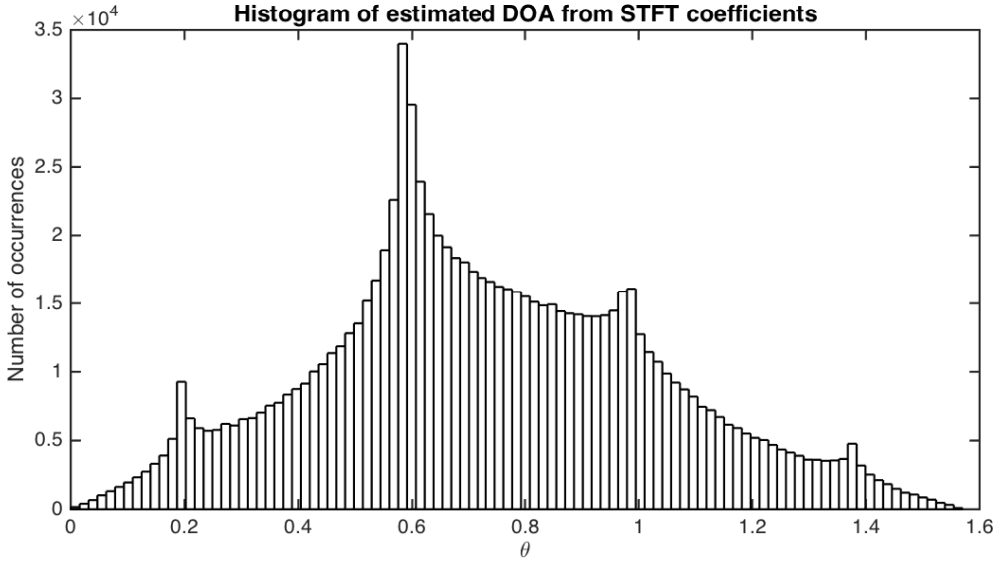


Figure 5.5: Histogram of DOA estimates from STFT coefficients.

Having obtained the histogram of the estimated DOA, the next step is to fit a GMM to this data. A popular approach to this problem is the expectation-maximisation (EM) algorithm [202]. EM has been extensively used in many problem domains and literature on the subject is widely available. A simple explanation with a basic implementation is given for example in [103]. EM is an iterative algorithm. The mixture model is first initialised with a parameter set Γ where the order of the model and the means of the components can be initialised with first estimates obtained from the histogram function. Then at each iteration, EM finds the optimal parameter set Γ' that locally increases the log-likelihood of the mixture model, that is the log-likelihood difference between the mixture models with parameters Γ and Γ' is maximised [103]. The algorithm continues until the log-likelihood difference has exceeded a certain pre-defined threshold.

The results of the EM algorithm applied to the data of the four source mixture example are depicted in figure 5.6. Observing the figure it can be seen that four components have been identified, along with their corresponding parameters (μ_k, σ_k, w_k). These components are essentially

spatial Gaussian filters [103] and will form the basis for the spatial masks. Figure 5.7 shows the histogram data overlaid with the calculated GMM components/filters. The EM algorithm used for this example is only initialised with the number of expected components but the results are accurate and observing figure 5.6 it can be seen that the components have been placed at the right (almost) spatial locations (i.e. estimated $\boldsymbol{\theta} \approx 0.26, 0.58, 0.89, 1.36)$ radians) with appropriate variances (spreads) apart maybe from the 1^{st} and 3^{rd} components that have slightly wrong means and a large variance for the 3^{rd} component (it is spread out a lot in the histogram). Such problems could be solved by allowing a decomposition with different settings and more expansion coefficients (i.e more iterations in the GMP algorithm) with the aim to produce sharper peaks in the histogram, by using a smoothed histogram as described in section 5.1 or by providing better initial estimates of the mixture model to the EM algorithm.

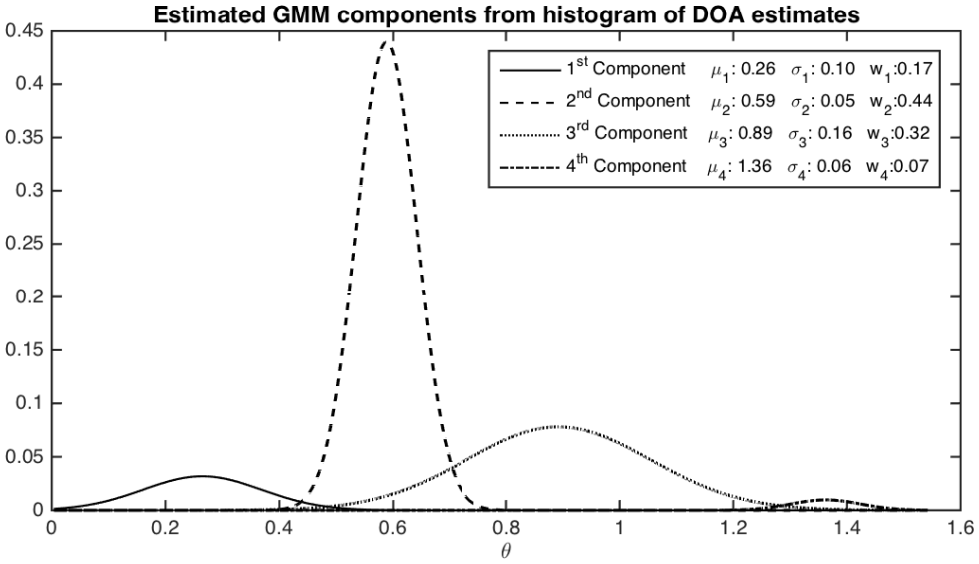


Figure 5.6: Estimated GMM components.

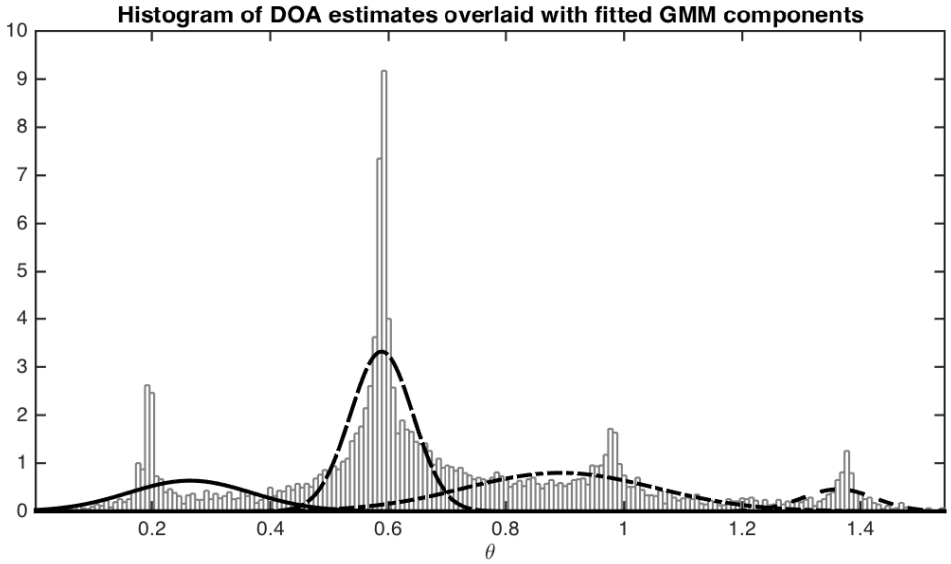


Figure 5.7: DOA histogram overlaid with GMM components.

The final step is to create the continuous time-frequency masks based on the results of the EM algorithm. Such a mask can be obtained by [103]:

$$\mathbf{M} = \begin{cases} P_K(k \mid \theta, \Gamma), & 10log_{10}|\phi_k(\theta \mid \mu_k, \sigma_k)| > L_{dB} \\ 0, & \text{otherwise} \end{cases} \tag{5.18}$$

where L_{dB} is a threshold value that accounts for the tail of the Gaussian distribution that stretches out to infinity [103], that is any coefficient pair below that threshold is not attributed to the source.

5.4 Separation & Categorisation of Crackles In Pulmonary Sounds

An interesting biomedical field where sound source separation techniques could prove very useful is the field of computerised lung sound analysis (CLSA). Research in this field of study focuses on the physical description of known pathologic sounds and techniques for their automatic analysis, identification and classification with the final goal of accurate medical diagnosis. Although this is not a musically related topic, the techniques used in CLSA have very much in common with techniques used in traditional musical signal analysis and the author has found a novel way to tackle a specific problem of CLSA using GMP, in particular the separation and categorisation of crackles from pulmonary (i.e. related to the lungs) sounds. This section is included as part of this thesis since it demonstrates one more potential use of GMP and further exemplifies its applicability to source separation problems.

The information contained within respiratory sounds can provide valuable clues regarding the physiologies and pathologies of lungs and airways obstruction [11], thus the identification and classification of such sounds is of paramount importance for correct medical diagnosis and treatment. This job is usually performed by a clinician via the use of a stethoscope for chest auscultation but there is research (reviewed in [11]) that suggests that detection, categorisation and validation of such sounds should not be based on auscultation alone. There are many arguments to justify the use of an automated system. First of all, conventional stethoscope auscultation is subjective [11] and an accurate diagnosis greatly depends on the experience and listening ability of the clinician. Certain adventitious (i.e. accidentally formed or not native) respiratory sounds contain subtleties that provide important information about the underlying pathology but are very hard to distinguish and easy to miss by simple auscultation. Another problem is that results obtained by such tests are usually not recorded thus they cannot be shared amongst physicians or stored for further analysis. Automated auscultation systems can help overcome many of these issues by providing accurate medical diagnosis and by allowing a systematic monitoring of patients, even at remote locations, and easy data exchange via the Internet [11]. Also such systems could prove useful in clinical research [10] and as learning aid for medical students [203]. Of course this is not an exhaustive list of arguments but the need for automated auscultation systems should be clear.

Generally speaking, pulmonary auscultation and sound analysis is concerned with a variety of sound types such as adventitious sounds, lung sounds, breath and normal breath sounds, the differences of which are explained in [204]. Also there is a variety of known 'markers' that are used for the categorisation of such sounds such as crackles, rhonchi, wheezes, squawks, stridors, cough and snoring sounds [11], all of which have certain spectral and temporal characteristics and

indicate a certain type of pathology. In the literature there are two main categories of abnormal sounds [205]:

1. Continuous or stationary sounds like wheezes and rhonchi.

2. Discontinuous or non-stationary sounds like crackles which are further categorised into coarse and fine crackles.

Also the two most studied noises are wheezes and crackles [11]. Figure 5.8 depicts a recording of a lung sound containing crackles. The first sub-plot clearly shows large bursts of energy which correspond to crackles but also smaller energy bursts some of which are difficult to distinguish and buried in noise. The right sub-plot is a zoomed version of the region between the dotted lines in the left sub-plot and shows the waveform of a typical crackle.

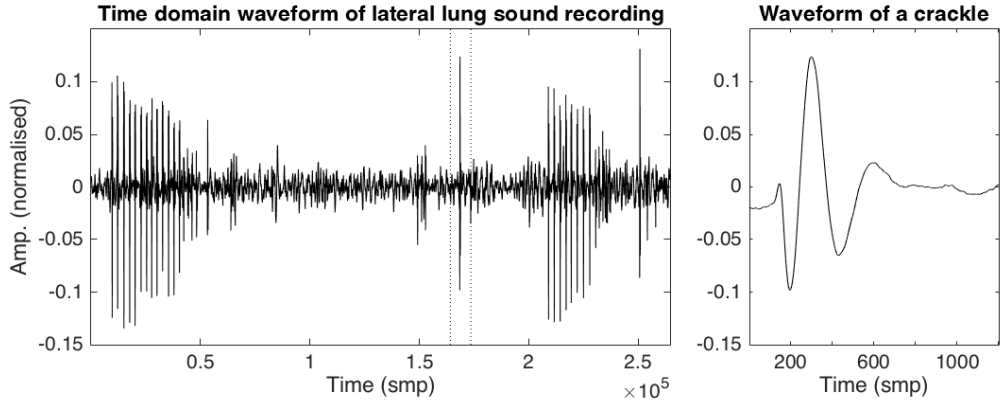


Figure 5.8: Time domain waveform of recorded lung sound captured from lateral position.

Crackles are usually described by several characteristics pertaining to their time domain representation. The characteristics of these sounds have been extensively described in the literature as for example in [206, 207, 208, 209] and their connection to main pathologies in [210]. Crackle sounds are generally characterised by an initial deflection width (IDW) which is the duration between the beginning of the crackle and the first deflection (zero-crossing), the two-cycle duration (2CD) which is the duration that covers the first two cycles of the crackle and the total duration width (TDW) which is the total duration of the crackle. Crackles are typically less than 20 ms in duration and their frequency content ranges from approximately 100 to 200 Hz. The crackle presented in the right sub-plot of figure 5.8 is about 800 samples long and the sampling frequency of the signal is 44100 Hz which makes the duration of this particular crackle approximately 18 ms. Also a quick inspection of the FT of that crackle shows that most of its energy is concentrated in the region between approximately 90 to 400 Hz where the extra frequency content could be attributed to external noise (the edges of the crackle in figure 5.8 are indeed a bit noisy). Finally crackles are further categorised, based on the duration of their temporal characteristics, into [11]

1. fine crackles: IDW$\approx 0.5ms$ or $0.9ms$, 2CD$\approx 3.3ms$ or $6ms$, TDW$\approx 4ms$ or $6ms$

2. coarse crackles: IDW$\approx 1ms$ or $1.25ms$, 2CD$\approx 5.1ms$ or $9.5ms$, TDW$\approx 6.7ms$

First of all it should be noted that crackles can be modelled with the crackle atoms described in section 3.6.9 but they also look very similar to the gammatone atoms described in section 3.6.4. Also by considering the time domain characteristics, especially the fact that the IDW is

smaller than the second cycle in the 2CD then it seems that crackles exhibit a certain amount of frequency modulation, which in certain cases is also noticeable in a spectrogram representation as produced by the STFT. Taking into account this frequency modulation then better modelling could be achieved by the use of gammachirp atoms (described in section 3.6.7). Surprisingly this fact does not seem to have been mentioned in the literature.

Before performing any sound analysis a required step in any auscultation system is the capturing of the sounds of interest. A typical sequence of steps performed at this stage is the following [11]:

1. Sound capture by placing microphones on desired areas on the subject. The effects of various microphone types can be found in [211]. Electret microphones are usually preferred.
2. Signal amplification.
3. Filtering and sampling.
4. Cardiac (i.e. heart related) sound cancelling/reduction.
5. Sound recording.

Recording of lung sounds is a tricky process and the resulting recordings typically suffer from external background noise but also noise stemming from within the body such as cardiac sounds which can hinder the analysis of the sounds of interest. Therefore reduction or cancellation of these artifacts is important for an accurate analysis and diagnosis. Looking in the literature most proposed automated systems and commercially available systems make use of high quality hardware components and try to eliminate the noise at the recording stage, for example by using high quality microphones and hardware filters at the sound capturing stage. As a result many solutions cost several hundreds or thousands of pounds sterling (a quick Goolge search can verify this). Also before analysis there are several techniques that are used solely for the task of noise reduction with the most promising ones being adaptive filtering and wavelet packet decomposition [11]. Having obtained a 'noiseless' signal the analysis of lung sounds takes place where, again, different algorithms and techniques are used for the identification and classification tasks.

The authors in [11] and [10] provide extensive reviews of state of the art automated auscultation systems and all systems they review use one or more of the following techniques and algorithms: temporal analysis, auto-regressive models, FFT, STFT, PSD, linear prediction of coefficients, genetic algorithms, neural networks, fuzzy filters, empirical mode decomposition and wavelet analysis. Reading the two aforementioned papers there are a few interesting observations to be made. First of all none of the reviewed systems use matching pursuit. Observing the spectra of various types of lung sounds, such as crackles and wheezes, it can be clearly seen that a sparse adaptive decomposition is most suited. The FFT, STFT, PSD or linear prediction coefficients are not suitable for representing such signals. It seems that the only attempt to adaptively represent lung sounds is via the use of wavelet analysis with wavelet packet decomposition. Also, according to [11] the most successful systems are based on wavelet analysis where some systems use wavelets for both denoising and classification of lung sounds which is an extra indication that sparse, adaptive decompositions are necessary. The good quality of wavelet based systems makes sense since wavelets are an excellent choice for representing sounds with discontinuities, but as

explained in section 2.1.7 wavelet techniques have certain limitations such as a fixed tiling of the time-frequency plane and also limited adaptivity of wavelet packets since this decomposition does not change over time. Matching pursuit algorithms are generic and can decompose a signal against multiple bases including wavelets, wavelet and cosine packets and the Fourier basis, to name a few, so why not implement a system where the front-end is an MP-based algorithm? It seems that a trend has been established in this field and all proposed systems follow it.

The second observation to be made is that the tasks of denoising, identification and classification are treated as separate problems and different algorithms and techniques are used for each solution. This is not a wrong approach to solve the problems posed by CLSA but again an MP-based algorithm could be used to perform all these tasks during the decomposition of a signal. For example, assume that a signal contains discontinuous sounds such as crackles with external background noise and apply an MP decomposition using a dictionary of atoms that are crafted in such a way so that they correlate well with these sounds. Then at each iteration the MP-based algorithm will select the best correlated atoms which essentially represent the crackles in the signal. After a certain number of steps and when the energy reduction has reached its slow asymptotic stage (which implies that none of the atoms in the dictionary correlate well with the residual anymore), the algorithm can stop. The resulting approximation signal will contain the important signal structures (i.e. the crackles) and the residual all the remaining background noise. Also the classification of the crackles is inferred by the parameters of their corresponding atoms (such as type, frequency, scale etc.). Therefore the steps of denoising, identification and classification tasks have occurred at the same time, during the decomposition of the signal. This particular example is demonstrated later on in this section.

A final observation is that none of the reviewed systems (at least the ones reviewed in [11] and [10] and the ones reviewed independently and informally by the author) treat CLSA as a sound source separation problem. In CLSA, several sound sources such as lung sounds, heart sounds, irrelevant internal noises and background noises are recorded using single or multiple microphones and then a single source of interest is isolated for further analysis. The recording essentially produces a mixture signal comprising several sources and the main task is that of source isolation prior to further analysis. This process exactly describes a source separation problem. In fact the problem could be formulated as an over-determined, determined or under-determined source separation problem depending on the number of sensors and the number of sources under investigation. Treating CLSA as a source separation problem could allow the use of some advanced separation algorithms such as those based on ICA, non-negative matrix factorisation (NMF) or spectral modelling. Even harmonic/percussive separation algorithms such as the one proposed in [212] could be put into good use by separating pitched sounds like wheezes from transient in nature (percussive) sounds like crackles. Please note that although this subject is not the main focus of this thesis, at the time of writing this work the author is not aware of any examples of source separation applied to CLSA.

In order to demonstrate the application of GMP on CLSA let us consider the example signal depicted in figure 5.8. The main tasks are separation of the crackles from the noise and classification of crackles. As explained earlier both tasks can be achieved by the GMP algorithm. The main idea is to use carefully designed atoms that will correlate well with the signal at hand. Using the time

and frequency characteristics of crackle sounds it is possible to either create static dictionaries containing a selection of crackles, gammatone and gammachirp atoms or set appropriate rules for dynamic dictionary creation to be used in GMP, as described in section 3.2. The results of decomposing the signal of figure 5.8 using GMP are shown in figure 5.9 where the top sub-plot depicts the original signal, the middle sub-plot the approximation and the bottom sub-plot the residual signal. Inspecting the approximation signal it can be clearly seen that the main features of the original signal have been successfully captured. Listening to the approximation signal verifies that all noise has been rejected and only the important crackles and the heart sound are audible. Of course a requirement is to remove cardiac sounds as well but in this simple example only a small amount of atoms were used. Cardiac sounds have different temporal and spectral characteristics which could be employed in the design of appropriate atoms capable of capturing heart sounds only. An important parameter to be considered is that of the number of iterations to be used in the decomposition. This parameter cannot be fixed and the stopping condition of the algorithm must be based on energy. The idea here is that since the dictionary comprises few selected atoms that are expected to correlate well with the crackles of the original signal, then after some iterations the algorithm should stop converging at the initial rate. By taking advantage of this behaviour the following energy threshold rule was defined:

$$E_{avg}[i] = \frac{1}{N_e} \sum_{k=i-N_e}^{i} \lfloor E_{r_k} \rfloor, \quad i = 1, \ldots, I \tag{5.19}$$

$$\text{stopping condition: } E_{avg}[i] == E_{avg}[i - N_e] \tag{5.20}$$

where i is the current iteration, I is the maximum allowable number of iterations, E_{r_k} (inside the summation) the residual energy at the current and past N_e iterations and $E_{avg}[i]$ is a vector holding the average energy values of the past N_e iterations. The algorithm stops when the value of the average energy at the current iteration is the same as the value of the average energy N_e iterations before. Using this approach the actual number of iterations performed are dictated by the value of N_e where small values will stop the decomposition quickly whereas large values will result in more iterations. For the example presented here, N_e was set to 10 and this produced a decomposition of 101 atoms. Also, regarding the sound separation aspect of this example, it should be noted that the proposed approach is similar to the idea of content adaptive dictionaries proposed in [168] and [169] but the main difference is that no dictionary training takes place since all atoms are expressed analytically.

Figure 5.10 depicts several atoms that were selected during this decomposition. Atoms of several scales have been chosen and this could be exploited for the classification task. Since the main temporal characteristics of fine and coarse crackles are known then the scale parameter could be a strong indication of the type of crackle that the atom tries to represent. An automated system could flag small scaled atoms as fine crackles and large scaled atoms as course crackles. It should be noted that in this particular example there is not a direct relationship between the crackles of the signal and the extracted atoms, meaning that in some cases several atoms have been selected to represent a single crackle. This problem occurs mainly because, for this example, the created dictionaries do not contain a wide selection of atoms. A first solution would be to allow more accurate dictionary creation rules during the GMP pre-processing step. Another solution would be to perform a post-processing step where atoms that significantly overlap in the time domain

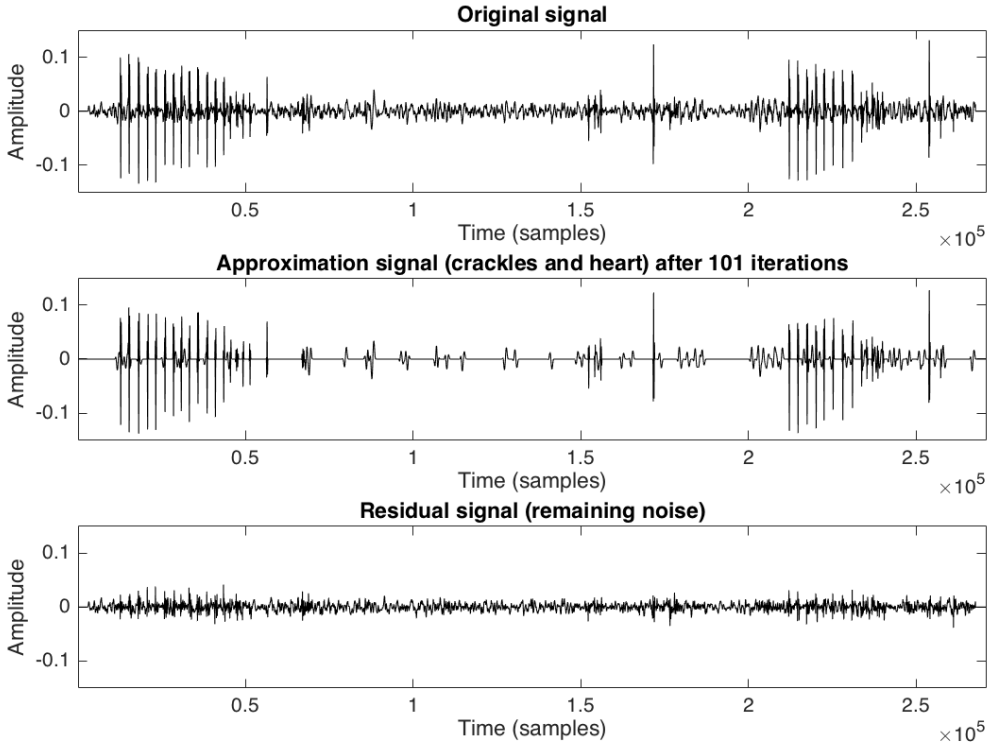


Figure 5.9: Original, approximation and residual signals after GMP decomposition.

can be grouped together to form a molecule that represents a single crackle. Note that this example is presented here as proof of concept and not as a polished solution. Nonetheless this example clearly demonstrates the ability of GMP to successfully separate crackles from lung sound recordings and its potential for crackle classification.

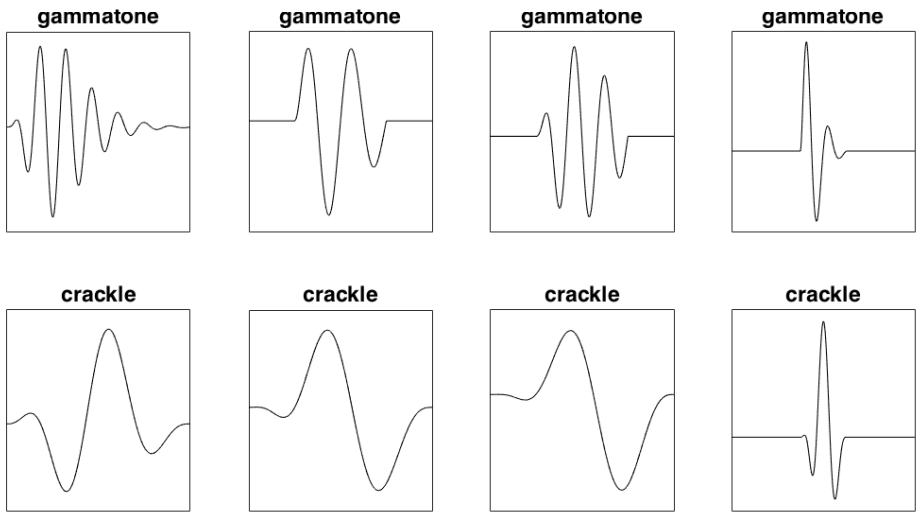


Figure 5.10: Atoms contained in the dictionary of the decomposition.

5.5 Summary

This chapter presented various approaches to sound source separation using GMP. In section 5.1 the problem of mixing matrix estimation was introduced and several possible solutions, already found in the literature, were discussed as well as potential extensions of those using GMP. Section 5.2 focuses on the problem of source separation of instantaneous linearly mixed stereophonic recordings and four different atom selection functions are described in detail. These functions correspond to step 5 of the GMP algorithm (algorithm 4) and demonstrate the ability to guide (or control) the source separation process by taking advantage of prior information about the mixture signals stemming from the assumptions of the intensity stereo mixing model the sparsity of the sources and the zero phase-offset between the selected atom pairs. In section 5.3 a typical Gaussian mixture model approach to source separation is described and adapted to work with the output of a GMP decomposition. Through the use of an appropriate example it is shown that when GMP (or any MP-based algorithm) is used as the front-end to a GMM system then the histogram of the estimated DOA is much clearer than the one produced by a typical STFT front-end. This occurs because GMP (and MP in general) produces sparser representations than the STFT, with only a few significant coefficients thus leading to histogram functions were peaks corresponding to mixing matrix directions are easier to distinguish and process. Finally in section 5.4 a novel approach to the problem of separation and categorisation of crackles from pulmonary sounds is described. This problem, found in the field of CLSA, has many proposed solutions with the most effective being based on wavelet analysis and wavelet packets decomposition, applied separately to the identification and classification (categorisation) tasks. Via the use of an appropriate example, a novel solution based on GMP is described and demonstrated and observations regarding several aspects of available state of the art systems are briefly discussed.

Chapter 6

Testing & Results

In chapter 5 several applications of GMP in sound source separation were presented and discussed. Of particular interest are the algorithms presented in section 5.2 where the atom selection function of GMP was modified to perform source separation of instantaneous linear stereo mixtures given a known mixing matrix. In this chapter the performance of these algorithms is evaluated using a set of commonly used performance measures. Also the same performance measures are used to evaluate the performance of some state of the art source separation algorithms thus allowing the comparison of those methods with the GMP.

6.1 Source Separation Performance Evaluation

When dealing with sound source separation, or any kind of source separation for that matter, an important step is the performance evaluation of the proposed algorithm. The goal of evaluating performance is twofold. First of all evaluating performance allows to assess the quality of an algorithm by providing numerical descriptions of all different distortions and artifacts that are introduced during the separation process and secondly it allows the comparison of different source separation algorithms regardless the specifics of their implementation or technique they are based on. Although there are plenty of performance measures (or criteria) found in the literature there are a couple of software packages, MATLAB® toolboxes in particular, that have become the standard within the source separation community for evaluating the performance of source separation algorithms (at least for sound source separation). These are the BSS evaluation toolbox and the PEASS toolbox, with each providing a different set of performance measures. These two packages and the performance metrics they provide are briefly discussed in the next subsections.

6.1.1 Global Evaluation Measures (BSS Eval Toolbox)

The BSS evaluation toolbox allows the performance evaluation of source separation algorithms when the original sources and possibly the noise perturbing the mixture are available for comparison [213][1]. Each estimated source $\hat{s}_p$ is compared with its original version to produce a set of

[1] http://bass-db.gforge.inria.fr/bss_eval/

performance metrics. In particular the computation of these metrics is a two-step process. The first step involves the decomposition of the estimated source into the following sum [214]:

$$\hat{s}_p = s_{target} + e_{interf} + e_{noise} + e_{artif} \tag{6.1}$$

where s_{target} is a modified version of the original source (modified by distortion) and the signals e_{interf}, e_{noise} and e_{artif} are error terms stemming from interferences relating to unwanted sources, noise from sensors and artifcats from other causes (e.g. musical noise) respectively. The second step computes energy ratios that evaluate the relative amount of each error term on the duration of the signal[214]. Four global performance metrics are defined:

Source to Distortion Ratio

$$SDR \stackrel{\text{def}}{=} 10\log_{10}\frac{\|s_{target}\|^2}{\|e_{interf} + e_{noise} + e_{artif}\|^2} \tag{6.2}$$

Source to Interferences Ratio

$$SIR \stackrel{\text{def}}{=} 10\log_{10}\frac{\|s_{target}\|^2}{\|e_{interf}\|^2} \tag{6.3}$$

Source to Artifacts Ratio

$$SAR \stackrel{\text{def}}{=} 10\log_{10}\frac{\|s_{target} + e_{interf} + e_{noise}\|^2}{\|e_{artif}\|^2} \tag{6.4}$$

Source to Noise Ratio

$$SNR \stackrel{\text{def}}{=} 10\log_{10}\frac{\|s_{target} + e_{interf}\|^2}{\|e_{noise}\|^2} \tag{6.5}$$

Another performance metric is the source image to spatial distortion ratio (ISR) which essentially measures the distortions introduced by incorrect estimation of the spatial images of the sources. As such this measure is relevant when stereo mixtures are considered and requires knowledge of the original mixing matrix. This measure was also used for the testing of the proposed algorithms. In particular, when multi-channel signals are involved then the following decomposition is suggested:

$$s_{est_{ij}} = s_{true_{ij}} + e_{spat_{ij}} + e_{interf_{ij}} + e_{artif_{ij}} \tag{6.6}$$

where $s_{est_{ij}}$ is the estimated spatial image form source j on channel i, s_{true} is the true source image, and e_{spat}, e_{interf} and e_{artif} are error components representing spatial distortion, interference and artifacts respectively[2] and the corresponding ISR measure is given by:

Image to Spatial distortion Ratio (ISR)

$$ISR \stackrel{\text{def}}{=} 10\log_{10}\frac{\|s_{true}\|^2}{\|e_{spat}\|^2} \tag{6.7}$$

[2]http://www.irisa.fr/metiss/SASSEC07/?show=criteria

The motivation and detailed explanations of these metrics can be found in [214]. For the testing of the GMP algorithm only the SDR, ISR, SIR and SAR measures were used. Note that all these metrics are measured in dB units and higher values imply better quality. It should also be noted that these metrics provide a 'global' description of quality, that is each metric provides a single number which describes the whole signal, therefore extra care should be taken when interpreting the results. Sometimes it is desirable to evaluate performance locally and this is achieved by the BSS toolbox by segmenting the signal into frames and then applying the aforementioned metrics to each frame. This approach was not used for the testing presented here because of the large amount of data that was acquired which would have made presentation of results impractical.

6.1.2 Perceptually Motivated Evaluation Measures (PEASS toolbox)

The PEASS toolbox[3] in contrast to the BSS evaluation toolbox provides a set of perceptually motivated performance measures which correlate better with subjective ratings. The authors in [215] propose a multi-criteria protocol to asses the quality of each kind of distortion produced by a source separation algorithm such as the target source distortion, interference from unwanted sources and noise artifacts and then based on the results produced by listening tests, design the following set of objective measures:

- the Overall Perceptual Score (OPS)
- the Target-related Perceptual Score (TPS)
- the Interference-related Perceptual Score (IPS)
- the Artifacts-related Perceptual Score (APS)

These measures are expressed as a percentage and generally a higher score implies higher quality regarding the distortion that is being measured. For detailed explanation and derivation of the aforementioned scores the reader is referred to [215].

6.2 Testing Goals & Methodology

In section 5.2, four different atom selection functions were described that take advantage of prior information regarding the mixture signal. The goals of this testing are to firstly evaluate the quality of the separated sources when each one of these functions is used and secondly compare the produced results with results obtained by current state-of the art systems that operate on similar principles. In particular, these systems are the demixing pursuit described in section 2.4.2, an implementation of which is provided by the MPTK software package and for the rest of this chapter will be denoted as 'DEMIX' and three STFT-based source separation algorithms, provided by Emmanuel Vincent[4] namely source separation via binary masking, denoted as 'BINMASK', source separation via ℓ^p norm minimisation [216] denoted as 'L0min' and source separation via ideal binary masking, denoted as 'IBM'. The 'IBM' algorithm provides binary masks that achieve maximum SDR and is used as the comparison benchmark for all other algorithms. The STFT-based

[3] http://bass-db.gforge.inria.fr/peass/
[4] http://sisec2008.wiki.irisa.fr

algorithms were used since they are suggested by the SiSEC[5] evaluation campaigns. Regarding GMP the four atom selection functions will be denoted as 'SBASS1', 'SBASS2', 'SBASS3' and 'SBAS4' and their details are described in section 5.2.

Its should be noted that the testing performed here is not exhaustive in a sense that only certain test cases were evaluated. As already seen in section 4.4 there are several aspects of the GMP algorithm that affect the quality of the decomposition and the results of that step will subsequently affect the quality of the separated sources. The GMP implementation is very flexible and highly parametrisable and testing for all possible parameter settings and inputs is not feasible. In order to reduce the number of possible test cases a set of parameters was selected based on the performance results obtained from section 4.4. In particular, for all tests that were performed, GMP was used in LFMP mode, using hann windowed sine and cosine atoms with a maximum length of 8192 samples and 5 scales. For the pre-processing step the STFT was used with a Blackman analysis window, 8192 samples long and $4\times$ overlap factor. Although it is possible to obtain a better decomposition with other settings, these parameters were used since they produce relatively good and well behaved decompositions in a reasonable amount of time (as seen in section 4.4). Also these parameters are almost in accord with the parameters used in the MPTK algorithm, thus allowing for a fair comparison between these two algorithms. Regarding the MPTK settings, 5 Gabor blocks were used with a maximum size of 8192 samples but with $8\times$ overlap factor. Although this particular MPTK configuration seems to perform better than the GMP configuration described earlier, it was still used since using a $4\times$ overlap factor produced worse decomposition results. The settings for the STFT-based algorithms are set by the implementation functions themselves so these functions were used as is with no further modifications.

Having decided on the configuration parameters of each algorithm a generic test procedure was devised. First of all three mixtures with various number of sources each were selected. The details of each mixture are documented in the next section. Then the following procedure was applied to each mixture:

1. Load the original single channel sources and the mixing matrix and produce a stereo mixture.
2. Setup all algorithms using the configurations described earlier.
3. For the MP based algorithms set an iteration step limit.
4. Perform source separation using each algorithm.
5. Compute the BSS Eval and PEASS performance metrics using the results obtained.
6. Record the performance evaluation metrics.
7. Go to step 3, increase the number of iterations and repeat only for the MP-based algorithms.
8. After all test cases have been covered plot all results.

[5] `https://sisec2008.wiki.irisa.fr`

In the above procedure the MP-based algorithms are tested for various numbers of iterations. In particular the number of iterations tested were 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15} and 2^{16}. This was essentially the only parameter that was altered and the reason is that this parameter plays an important role in the quality of a decomposition and subsequently the quality of the separated sources. As already seen MP algorithms tend to converge fast in the first few iterations but as the algorithm progresses convergence becomes slow. This slow converge stage is an indication that the most important signal structures have been captured and what remains can be considered as noise. Generally speaking there is no single correct way to set the number of iterations. This setting can pose problems since if only few iterations are used then the algorithm might not capture all important signal structures whereas if too many iterations are used then the algorithm will start extracting atoms with very little energy which can hinder the quality of the separated sources. Although there are ways to handle this issue, as for example by using the stopping criterion proposed in section 5.4, in this batch of testing it was decided to try various iteration step sizes in order to inspect the behaviour of the algorithms as the number of iterations increased. Generally it is expected that the results will improve as the number of iterations increases.

6.3 Examples

The following table provides information about the mixtures that were tested:

Folder name	# Sources	Description	License
dev2_nodrums_inst	3	Mixture with acoustic guitar, electric guitar and bass	CC[6]
ins_mix_4src	4	Mixture with clarinet, violin, soprano and viola	CC
dev1_female4_inst	4	Four females speaking different languages	CC

Table 6.1: Mixture signals used in SBASStesting.

All development mixtures (folder names starting with dev) were obtained by the 2013 SiSEC campaign website[7]. These mixtures were selected because they provide a good selection of source configurations and signal types. The third mixture ('ins_mix_4src') was selected because it is a special anechoic recording of symphonic music found in the Aalto University website[8]. The aforementioned mixtures are also provided with the GMPC software package so there is no need to individually download them.

6.4 Results & Discussion

This section presents the most important results that were obtained during the testing phase. The testing procedure described in section 6.2 produced a huge amount of data, therefore only the most relevant are presented in this section.

[6] Creative Commons Attribution-NonCommercial-ShareAlike 2.0

[7] `https://sisec.wiki.irisa.fr/tiki-index.php?page=Underdetermined+speech+and+music+mixtures`

[8] `https://mediatech.aalto.fi/en/research/virtual-acoustics/research/acoustic-measurement-and-analysis/85-anechoic-recordings`

In order to facilitate easier inspection and discussion of the results the order and format they are presented will be described first. First of all a test refers to a mixture and a test case to a specific run of a source separation algorithm on that mixture. All figures of each test are included in their own sections and are placed in the same order. In boxplots and bar charts all figures are presented in the same following order: SBASS1, SBASS2, SBASS3, SBASS4, DEMIX, BINMASK, L0min, IBM and in figures where results are plotted on the same space, each algorithm has its own symbol: dot, circle, cross, star, diamond, left triangle, right triangle, up triangle. All figures are properly titled and clearly indicate which test case they represent and all have appropriately named legends. Finally each algorithm has the same color in every figure, regardless the type with light colours representing the four GMP algorithms and darker colours the DEMIX algorithm followed by the STFT-based algorithms.

The ordering and content of the figures in each section is as follows: The first figure depicts metrics of decomposition performance, in particular the SRR and execution times of each test case and for the MP-based algorithms only the test case with 2^{16} iteration steps is presented since these test cases provide most of the information. The second and third figures represent the BSS performance evaluation metrics and the PEASS evaluation metrics respectively. In these figure the x-axis represents the test cases that correspond to runs of the MP algorithms with different numbers of iterations ranging from 2^{10} to 2^{16}, that is six test cases for each algorithm. For the STFT-based algorithms the x-axis is not used since the separation task occurs only once, although their results are displayed at each step of the x-axis for making comparison easier. the STFT algorithms appear as straight lines in this figures which is the correct behaviour. The fourth and fifth figures depict boxplot representations of the results shown in the second and third figures. These representations are more compact and reveal certain trends that are not visible in the detailed views of the third and the fourth figures. The way to interpret the box-plot figures was discussed in section 4.4.5.

6.4.1 dev2nd - SBASS Test Results

This section presents and discusses the results obtained from the separation of the 'dev2nd' mixture. Figure 6.1 displays some decomposition metrics of that separation. First of all, one important observation is that the SBASS1 algorithm under performs. This algorithm fails to converge and as such fails to achieve any useful source separation. This behaviour was expected since by definition this algorithm takes advantage of DOA estimates alone without considering energy reduction whereas all other SBASS algorithms that take the energy of atoms into account, follow an SRR curve typical of an MP decomposition and do converge. As such algorithm SBASS1 is discarded and will not be considered in subsequent discussions. A second observation to be made is about the behaviour of the DEMIX algorithm. It seems that the algorithm fails to converge. The reason why this is happening is unclear and is solely related to the MPTK implementation. The GMPC toolbox merely calls the binary executables of the MPTK and after the completion of each decomposition the results are imported into the GMPC toolbox from the files generated by the MPTK binaries. Also this error is not produced during the import stage since a normal MPTK decomposition operates as expected. Although the DEMIX SRR curve fails to converge the performance metrics obtained are unaffected and audition of the produced signals (i.e. approximation, residual and individual sources) confirms the correct operation of the algorithm therefore

the DEMIX SRR results are assumed erroneous and will not be considered. A final observation to be made regarding the SRR decomposition metric is that algorithm SBASS4 outperforms all other MP-based algorithms and the BINAMSK algorithm and as the iterations increase the algorithm steadily approaches the IBM result. This outcome is positive since it indicates that the algorithm does a good job in absorbing most of the energy from the residual but this behaviour does not translate to good separation quality.

The bottom sub-plot of figure 6.1 displays the execution times for all algorithms (for MP-based algorithms the test case with the maximum number of iterations is displayed). It is clear from that plot, that the SBASS algorithms under perform in this domain. The DEMIX algorithm is approximately 5 to 8 times faster than the fastest SBASS algorithm which is SBASS4. This outcome was also expected and can be explained. First of all GMPC is coded in MATLAB which is an interpreted environment whereas MPTK is coded in C++, a compiled language, which implies a more optimised code (optimisations occurring at the compiler level). Of course this is not the only reason for the poor speed performance of the SBASS algorithms. Another main bottleneck in GMPC is the pre-processing step which at the moment only implements a basic two-level search three whereas in MPTK the created search tree adapts to the length of the input signal. Looking for a maximum value within a vector of values is a costly operation and informal performance testing of GMPC has shown that the algorithm does indeed spend most of its time at the pre-processing step. A possible solution would be the implementation of a more efficient search tree algorithm. Another bottleneck in the GMPC implementation is the computation of the inner products which although it is performed using an FFT based algorithm it becomes slow when the dynamic dictionaries become large in size. These issues are implementation specific and some of them can be fixed by writing more efficient code.

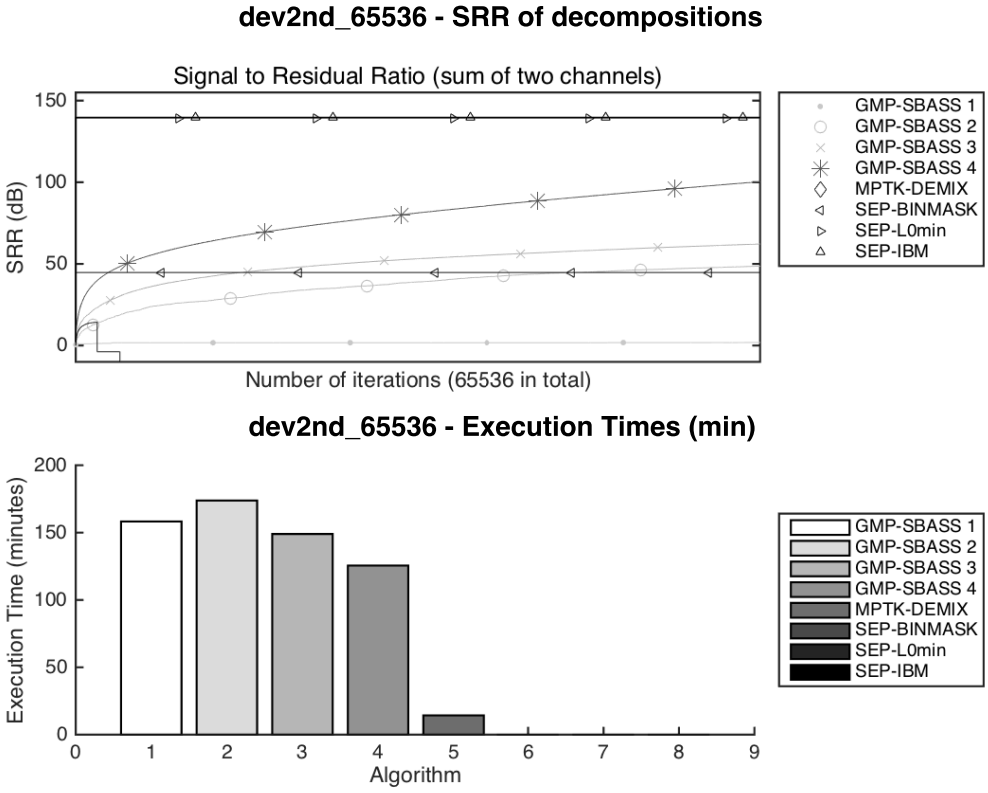


Figure 6.1: Decomposition measures for dev2nd mixture.

Figures 6.2 and 6.3 display the results of 'BSS Eval' and 'PEASS' evaluation metrics respectively, for each separated source and all test cases (i.e. all iteration runs). Looking at the sets of the corresponding figures there are a few trends that can be observed. First of all in many cases the results of the MP-based algorithms improve as the number of allowable iterations increase. This is probably the most correct behaviour since it implies that most of the selected atoms in each decomposition are correctly assigned to appropriate sources. Another observable trend is that some metrics improve significantly when few iterations are used worsen as the number of iterations is increased. This pattern probably occurs because of the iterative nature of MP. To explain further, at the first few iterations of the MP algorithms the extracted atoms typically correspond to important components within the residual and as such during these first iterations the atoms are correctly allocated to appropriate sources which translates to increased performance. As a decomposition is let to run for more iterations the extracted atoms will start to correlate less with the residual which could possibly lead to erroneous decisions of the atom selection functions. So the situation could be that the energy of a separated source increases but the selected atoms do not necessarily correspond to that source therefore leading to decreased performance. A final observable trend is that the metrics stay almost constant no matter the number of iterations used. This happens for both the GMP-based and MPTK algorithms and could be explained using an argument similar to that used in the previous case.

Inspecting figure 6.2 more closely, it can be seen that the MP-based algorithms achieve better SDR and SIR scores but worse ISR and ASR scores compared to the STFT-based algorithms. Also, for the MP-based algorithms the SBASS2 and SBASS3 do not perform as well as SBASS4 and the DEMIX algorithms. Regarding the STFT-based algorithms it can be seen that the L0min algorithm is on par with the IBM algorithm which seems to perform best in many cases whereas the BINMASK algorithm generally under performs. Looking at the PEASS evalution measures in figure 6.3 it can be seen that the MP-based algorithms achieve similar scores to the STFT-based algorithms. These claims can be better verified by looking at figures 6.4 and 6.5 which illustrate the results of figures 6.2 and 6.3 in a box-plot format. In figure 6.4 it can be seen that on average SBASS 4 and DEMIX achieve better SDR and ISR scores whereas L0min achieves better ISR and SAR scores. Also in figure 6.5 it can be seen that SBASS4 and DEMIX achieve similar OPS, TPS and APS scores with the STFT algorithms but have a lower IPS score.

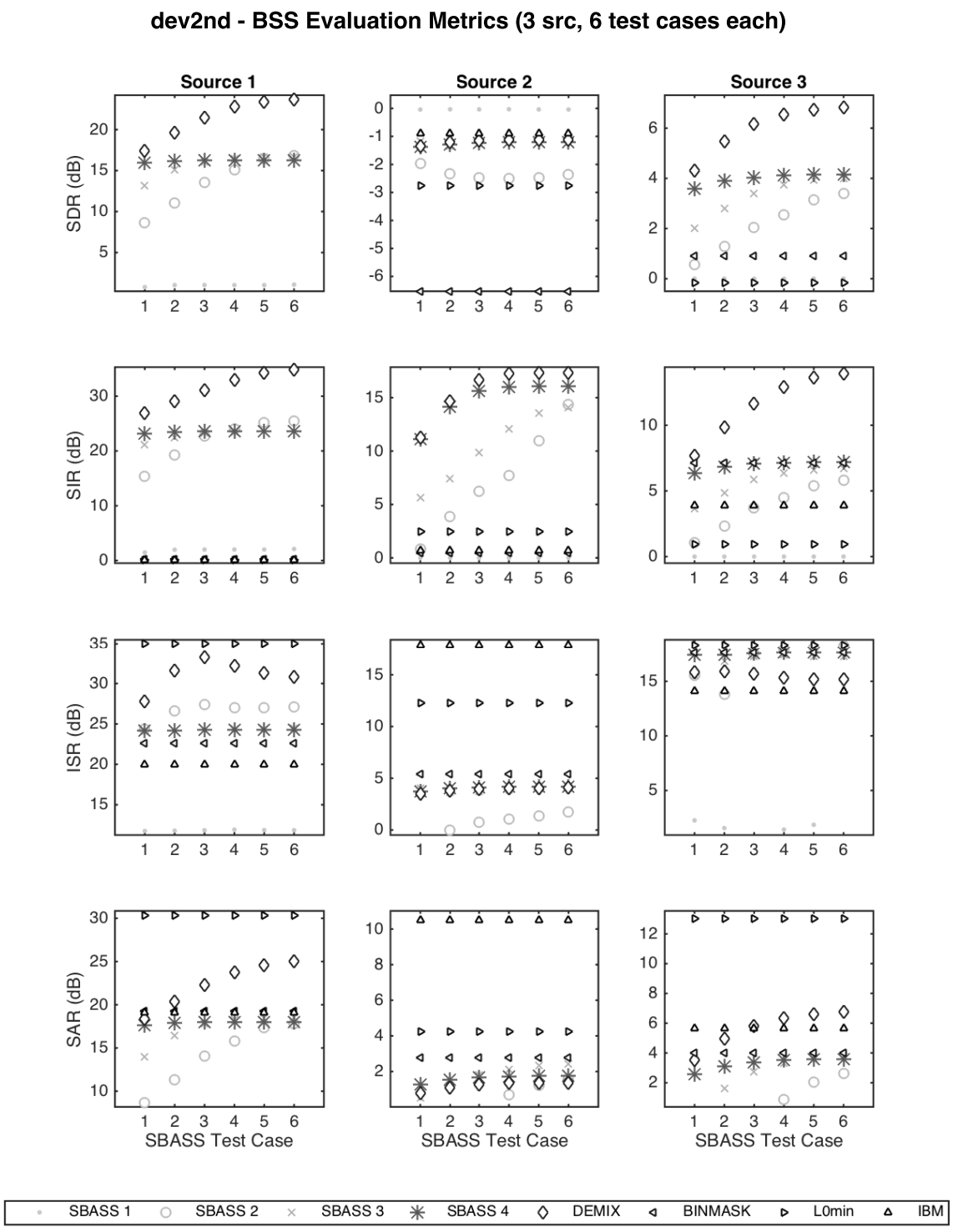


Figure 6.2: BSS evaluation metrics for 'dev2nd' mixture.

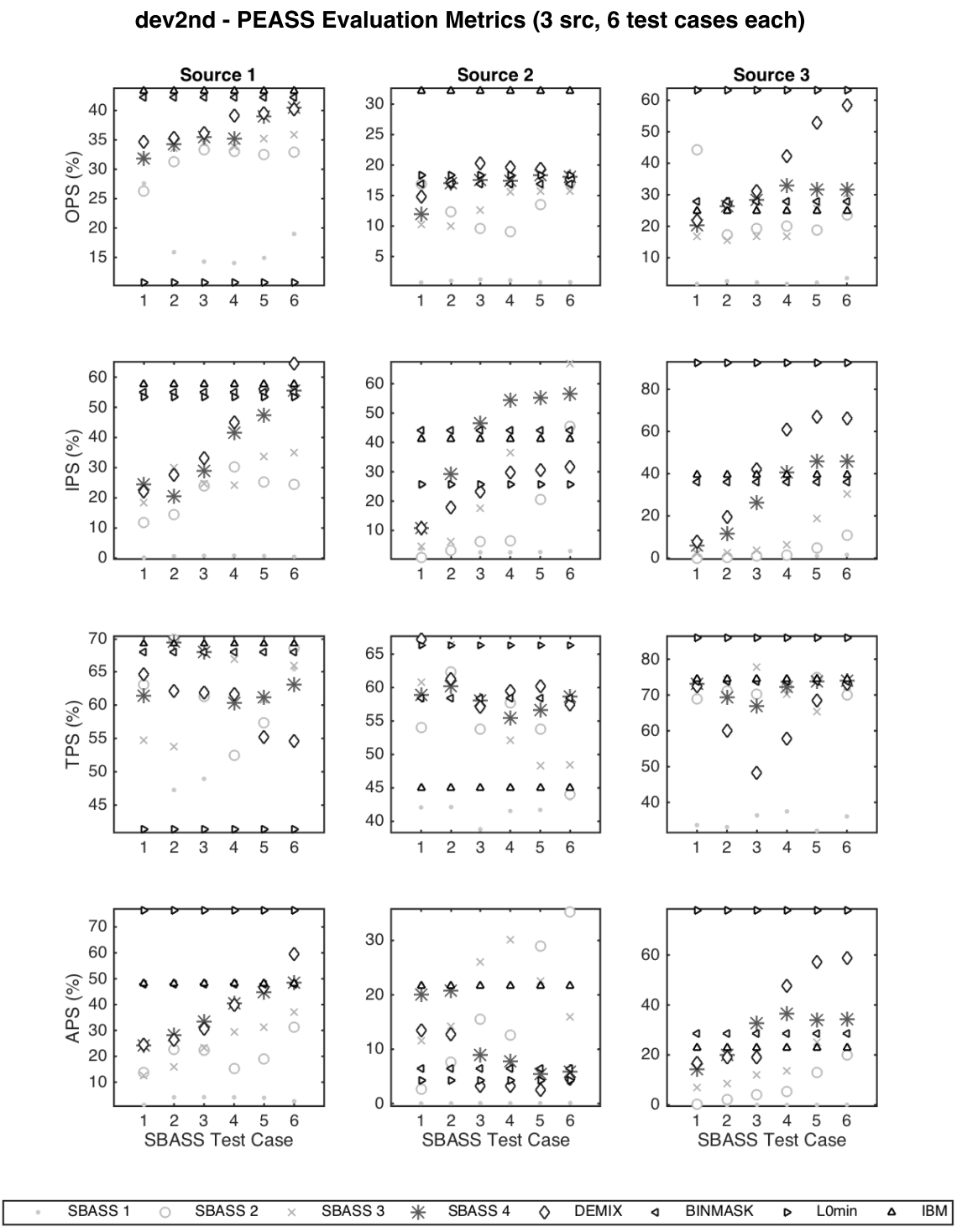


Figure 6.3: PEASS evaluation metrics for 'dev2nd' mixture.

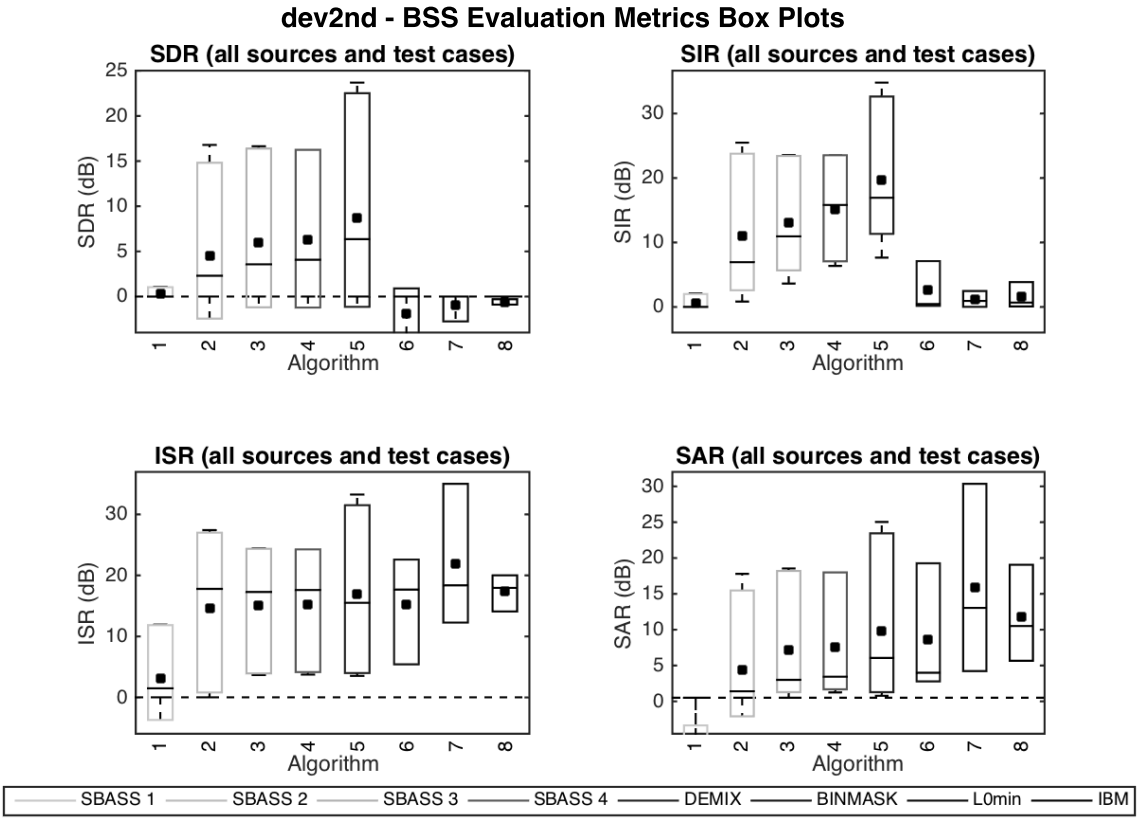


Figure 6.4: BSS evaluation metrics for 'dev2nd' mixture.

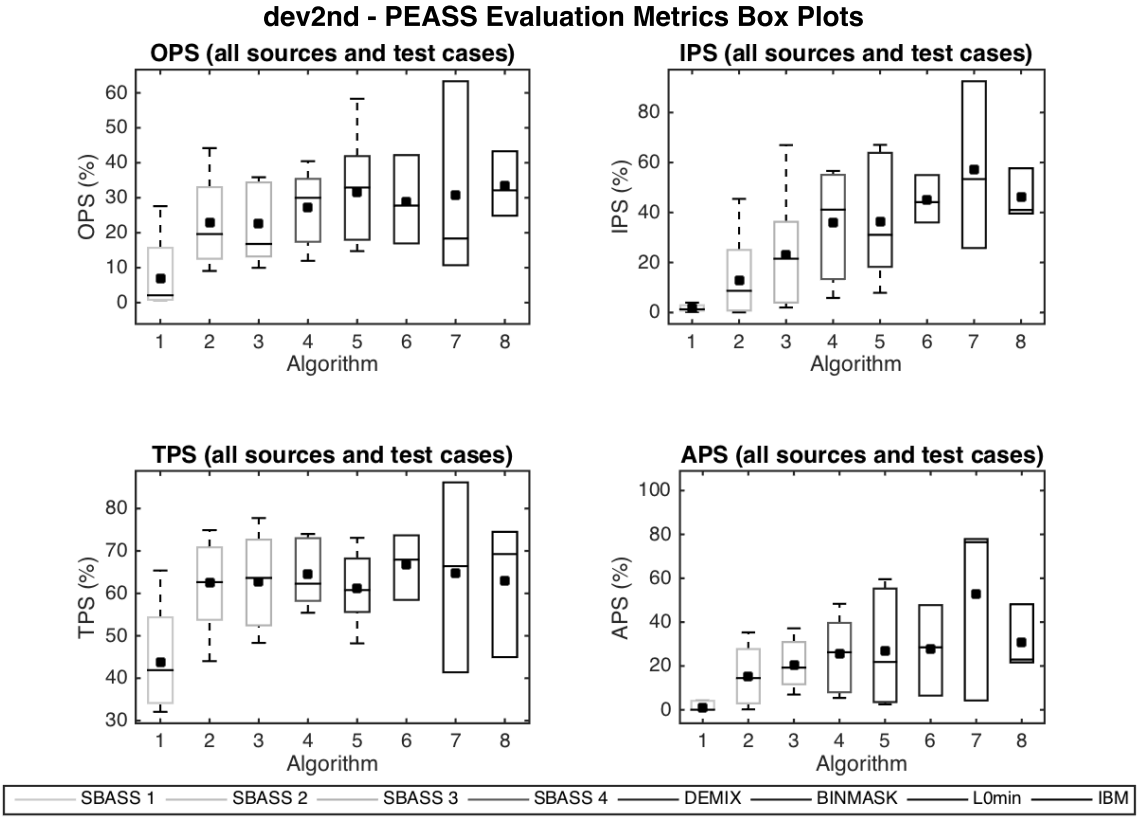


Figure 6.5: PEASS evaluation metrics for 'dev2nd' mixture.

6.4.2 insmix - SBASS Test Results

This section presents and discusses the results obtained from the separation of the 'insmix' mixture. In a similar fashion to the previous discussion, figure 6.6 displays the SRR decomposition metric and execution time for that mixture for the case of 65536 iteration. The SRR plot again clearly shows that algorithms SBASS1 and DEMIX fail to converge (see previous discussion) with SBASS4 converging faster and steadily approaching the SRR values of L0min and IBM algorithms. The execution time sub-plot shows that SBASS algorithms take the most time to execute with SBASS4 performing marginally better compared to the other SBASS algorithms and DEMIX being the faster among all MP-based algorithms. The STFT-based algorithms execute almost instantly, which is one of the main benefits of using such approaches.

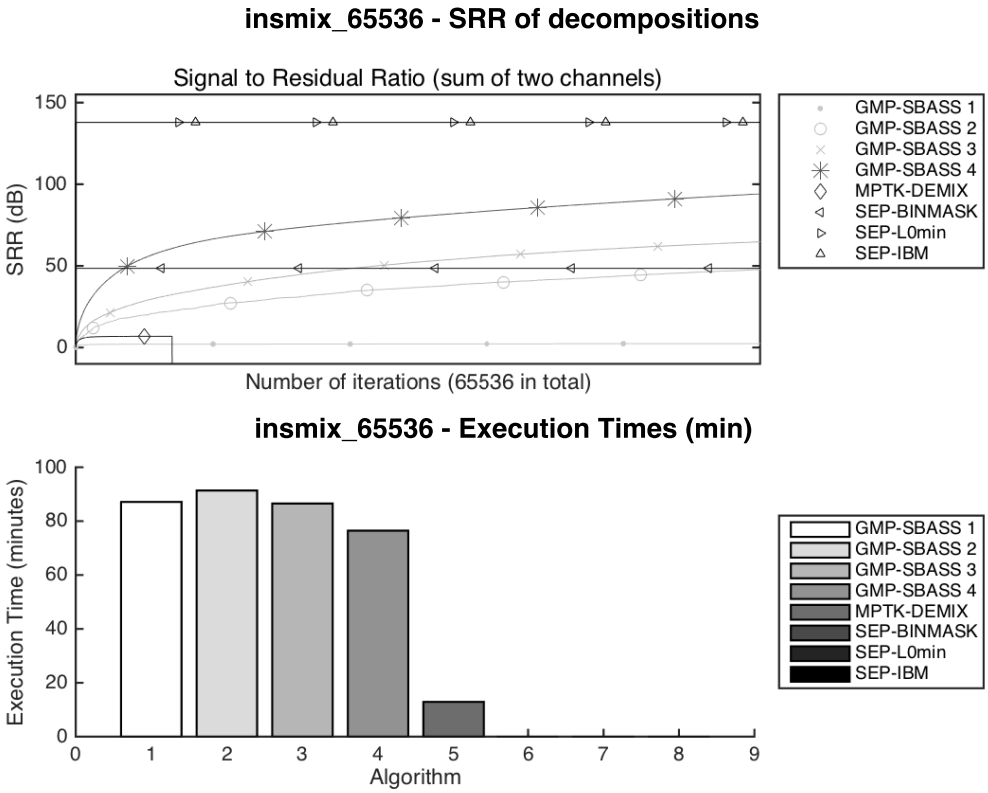


Figure 6.6: Decomposition measures for insmix mixture.

Figure 6.7 shows the BSS evaluation metrics for this mixture and again it can be seen that the MP-based algorithms achieve higher SDR and ISR scores compared to the STFT-based algorithms but generally lower ISR and SAR scores. Also in this case both SBASS4 and DEMIX algorithms obtain almost identical results across all metrics. In the PEASS evaluation metrics shown in figure 6.8 the situation is less clear but in quite a few cases MP-based algorithms achieve high scores for the $6th$ test case (i.e. 65536 iterations).

Figure 6.9 depicts the BSS evaluation metrics in a box-plot format where again it can be observed that STFT-based algorithms obtained low SDR and SIR scores but better ISR and SAR scores compared to MP-based algorithms. Figure 6.10 displays the PEASS evaluation metrics in a box-plot format where it can be seen that SBASS4 and DEMIX obtain similar scores for all metrics and are comparable to the STFT based algorithms.

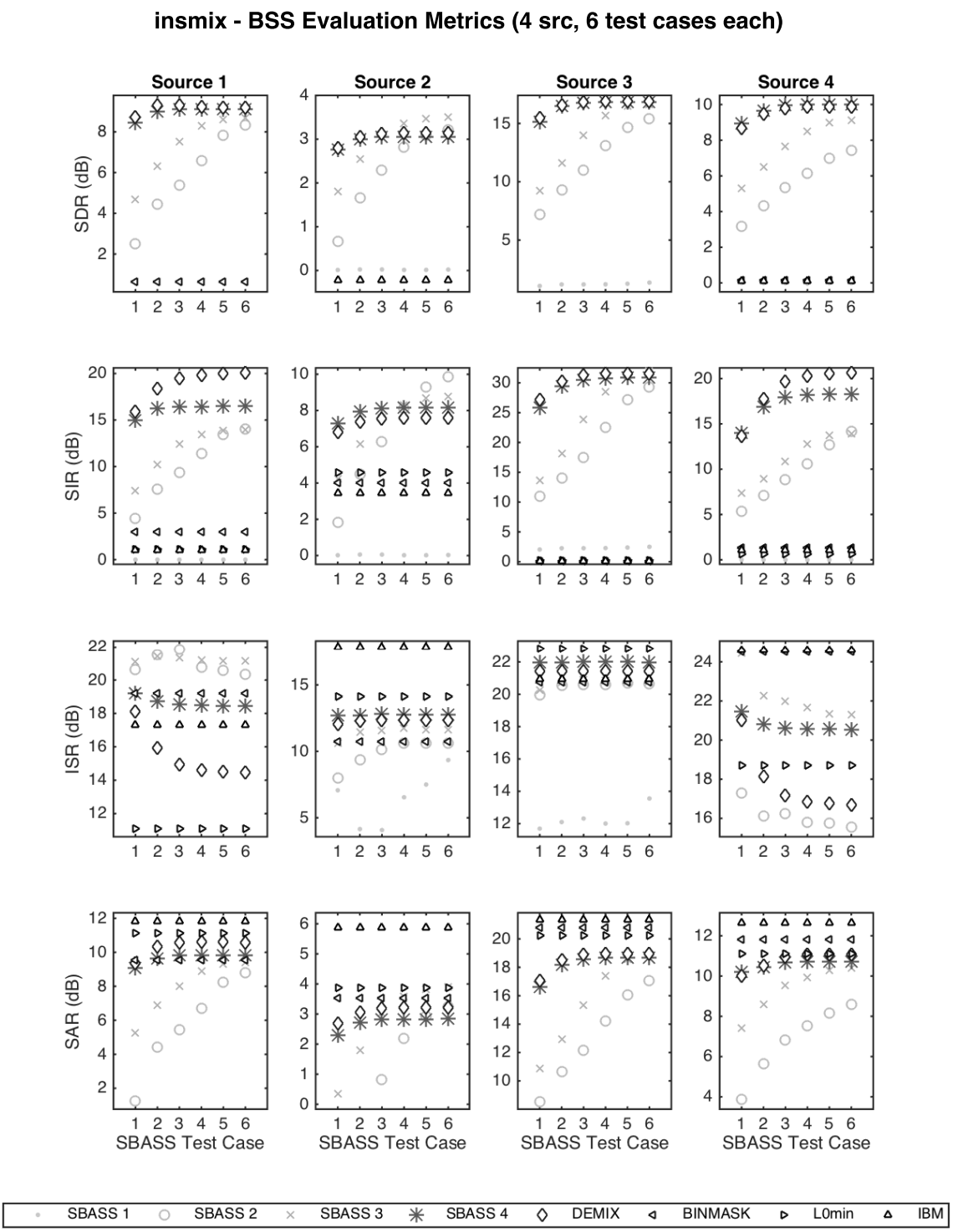


Figure 6.7: BSS evaluation metrics for 'insmix' mixture.

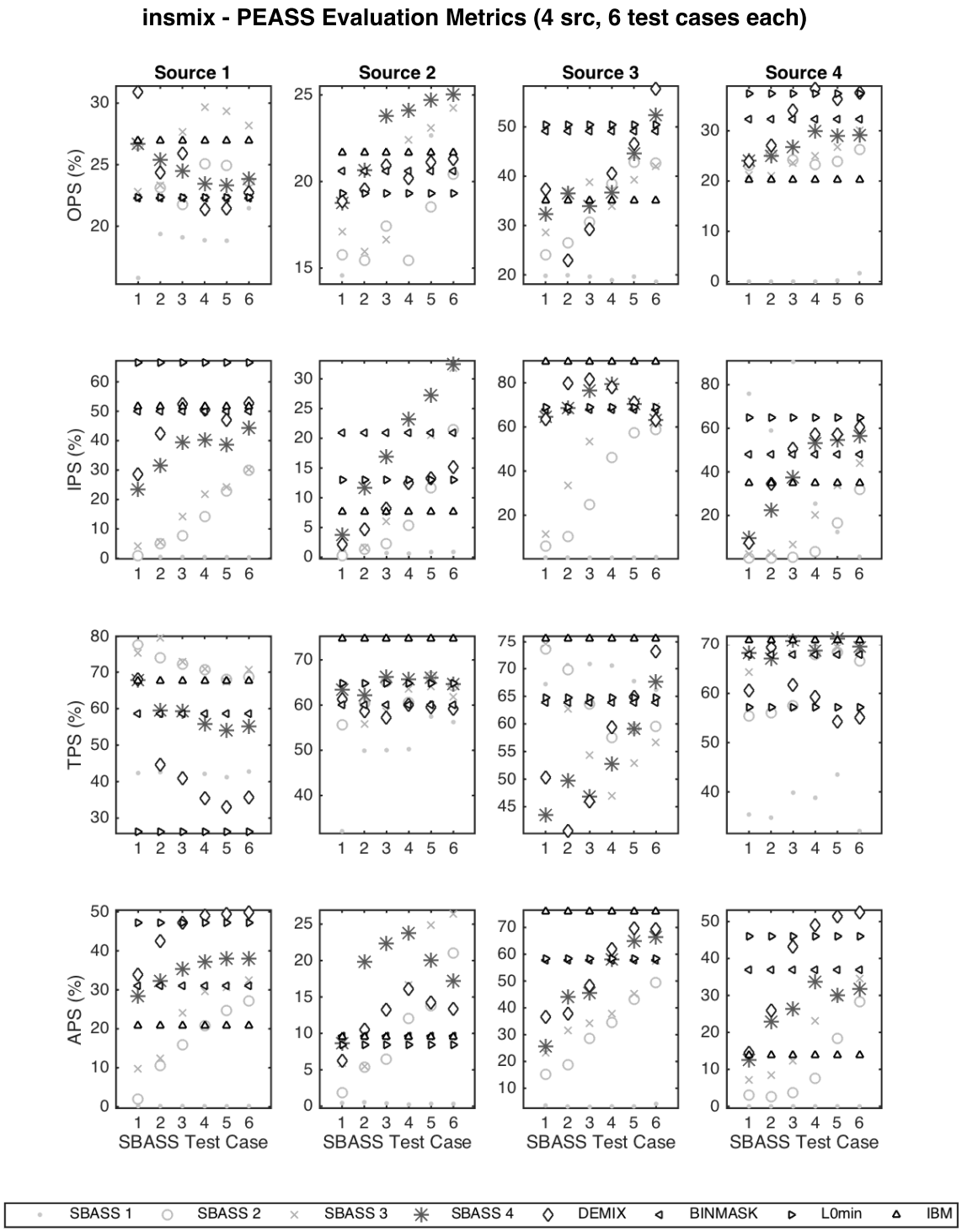


Figure 6.8: PEASS evaluation metrics for 'insmix' mixture.

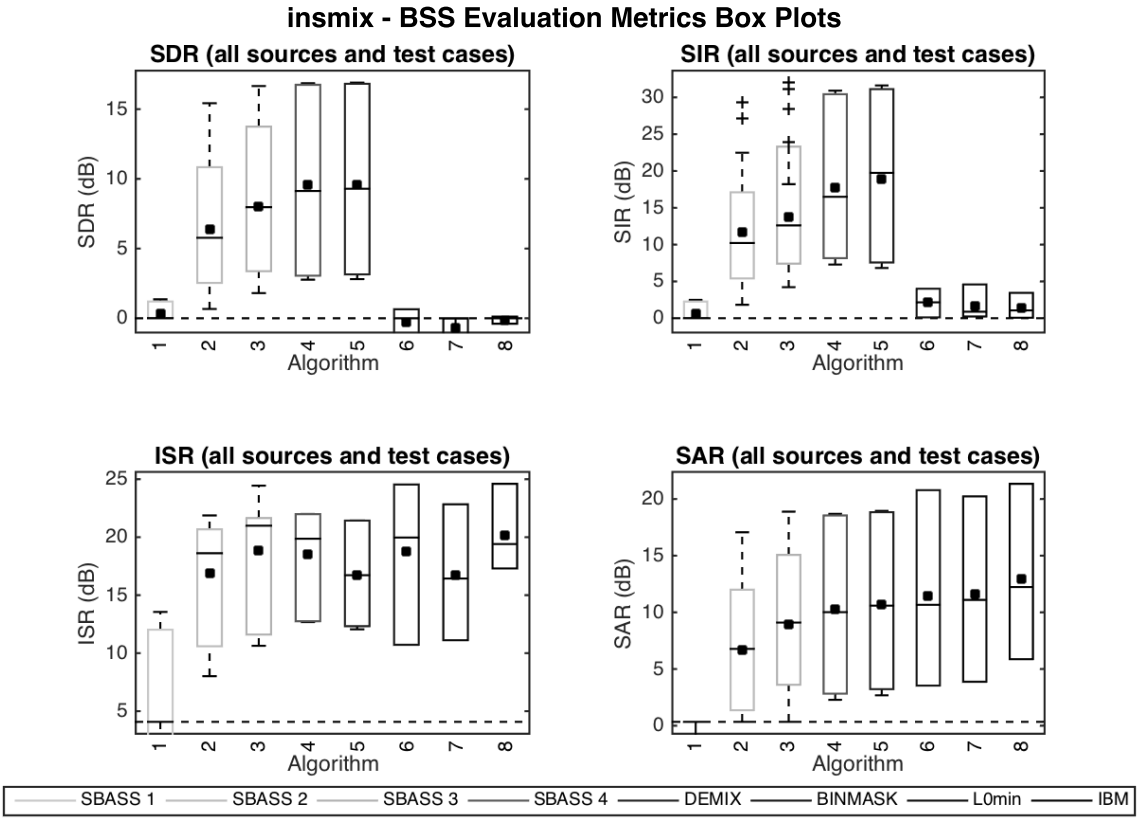


Figure 6.9: BSS evaluation metrics for 'insmix' mixture.

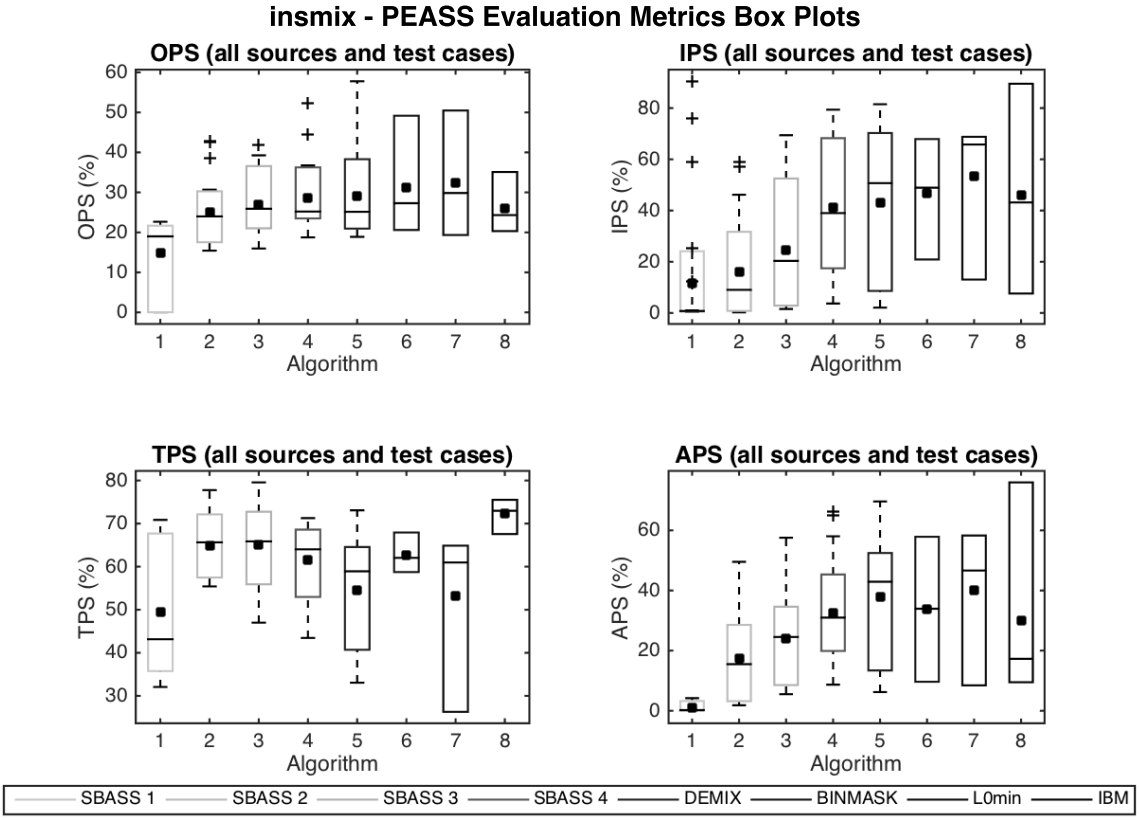


Figure 6.10: PEASS evaluation metrics for 'insmix' mixture.

6.4.3 dev1sf4 - SBASS Test Results

This section presents and discusses the results obtained from the separation of the 'dev1sf4' mixture. Figure 6.11 displays the SRR metric and execution times of this separation example where SRR convergence and execution times follow the same trend as i the previous examples.

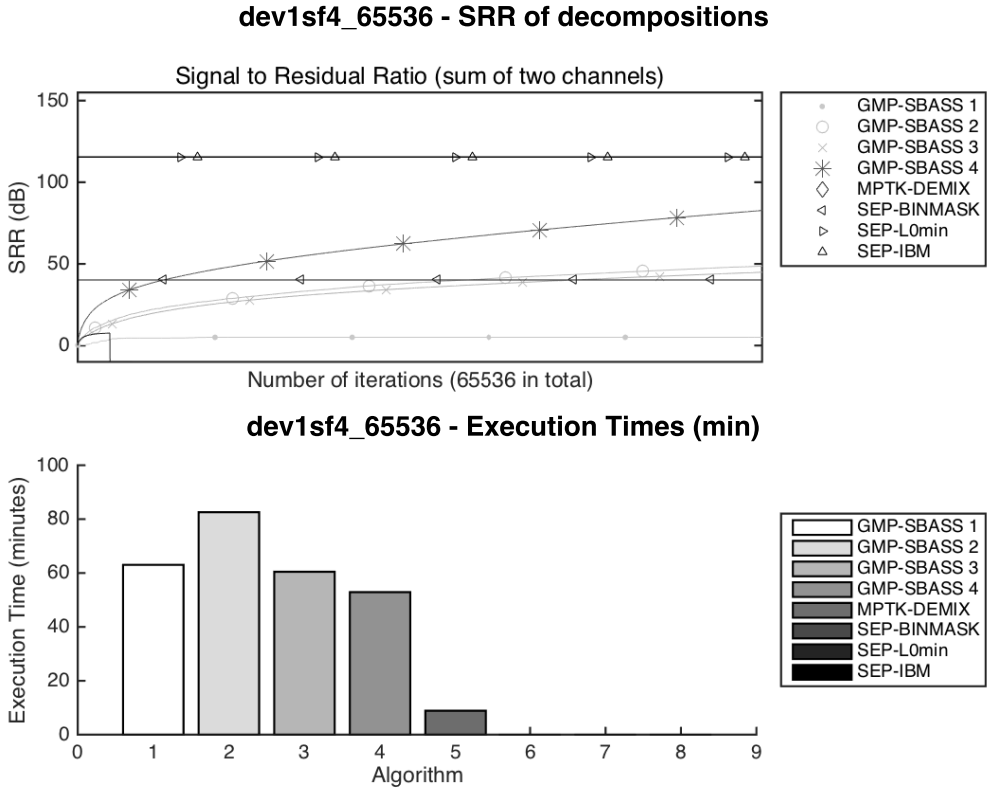


Figure 6.11: Decomposition measures for dev1sf4 mixture.

Figure 6.12 displays the BSS evalution metrics for all test cases where again, it can be seen that the MP-based algorithms (and especially SBASS4 and DEMIX) achieve higher SDR and SIR but lower ISR and SAR scores compared to the STFT-based algorithms where the L0min and the IBM algorithms seem to out-perform the BINMASK algorithm.

Observing the PEASS meaures in figure 6.13 it can be seen that the MP-based algorithms achieve high scores in all metrics at the test cases with large iteration numbers and are comparable to the scores obtained by the L0min and IBM algorithms.

Looking at the box plots of these metrics in figures 6.14 and 6.15 reveal similar trends.

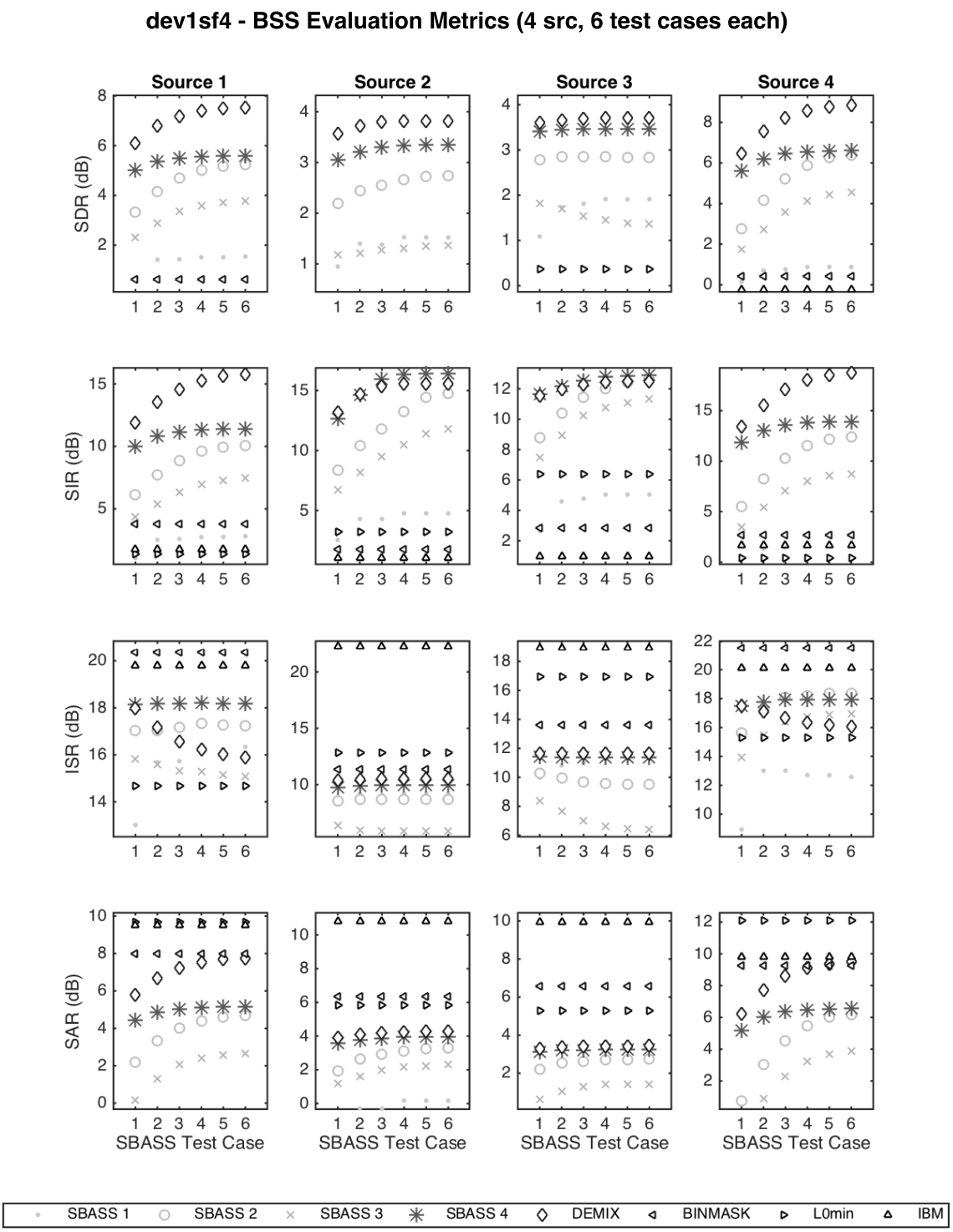


Figure 6.12: BSS evaluation metrics for 'dev1sf4' mixture.

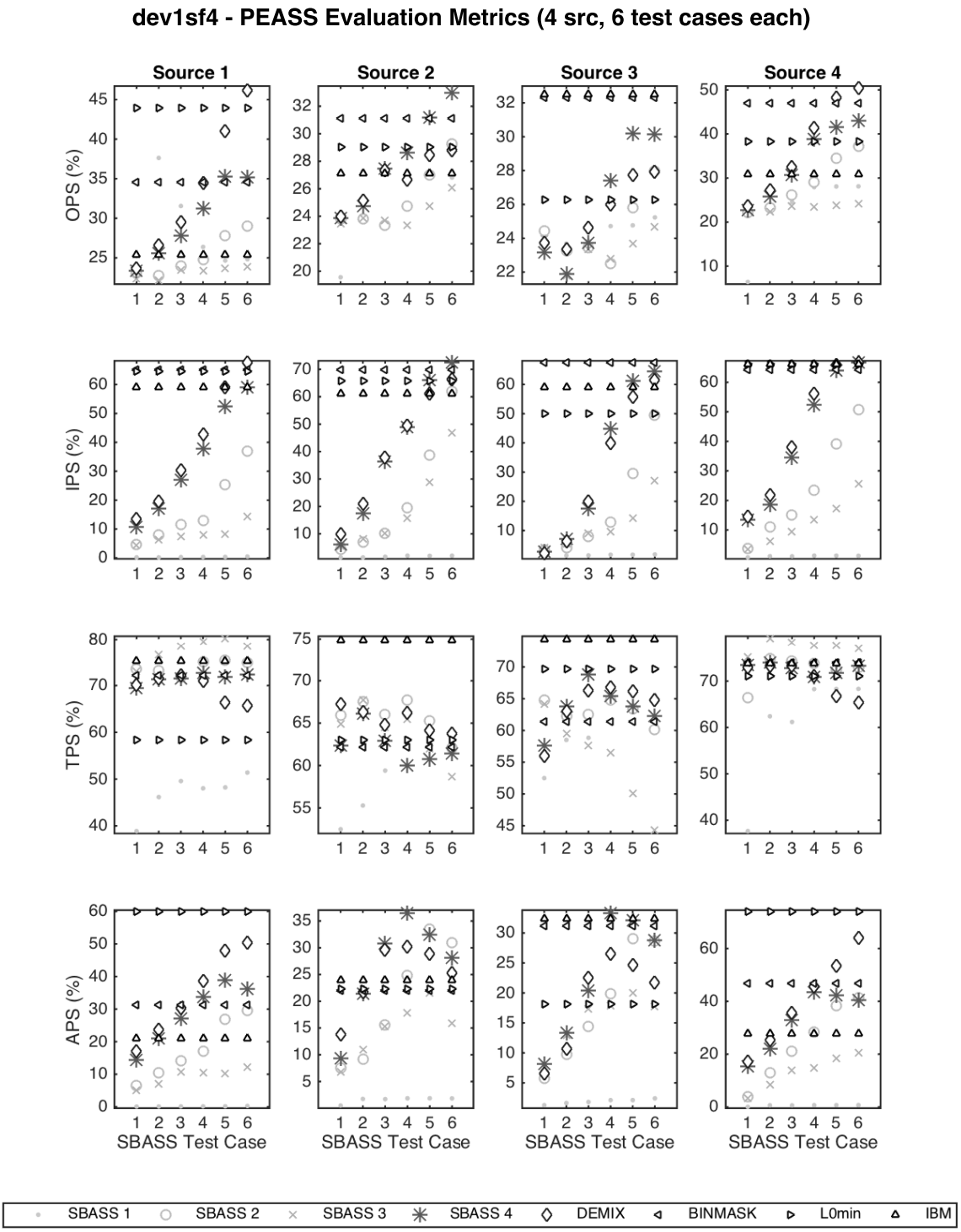


Figure 6.13: PEASS evaluation metrics for 'dev1sf4' mixture.

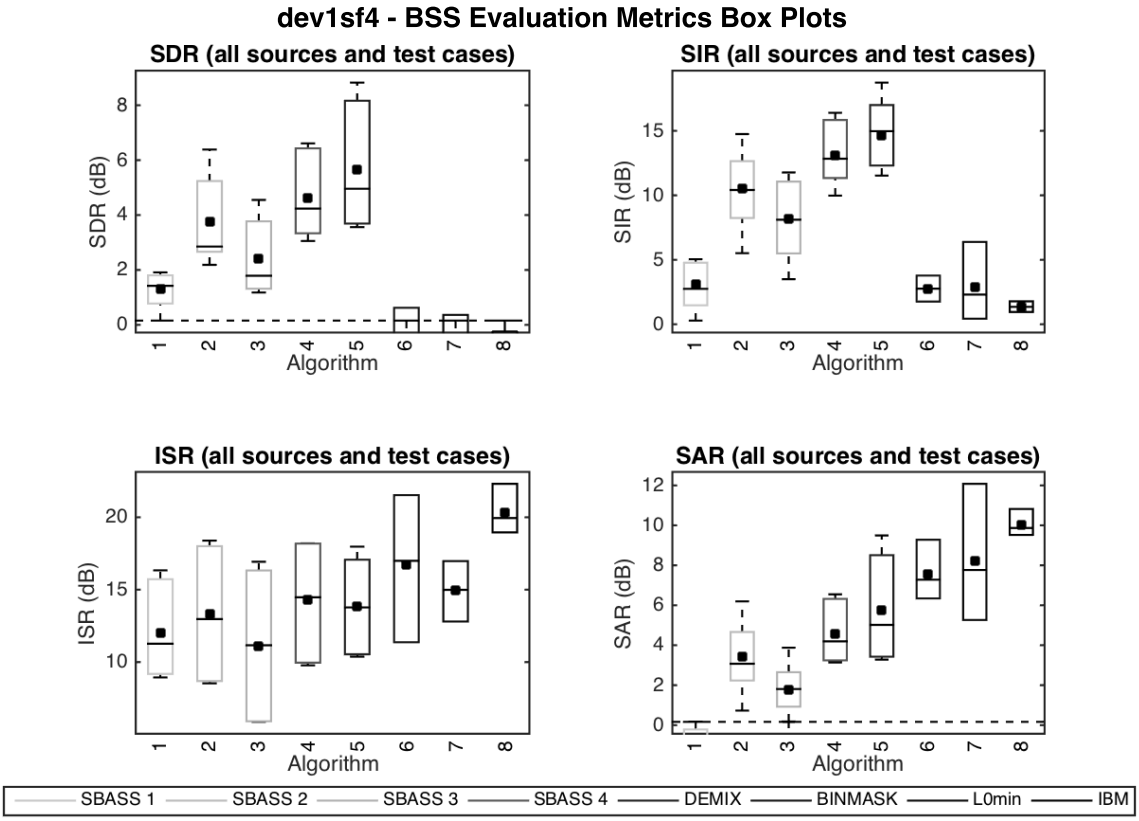


Figure 6.14: BSS evaluation metrics for 'dev1sf4' mixture.

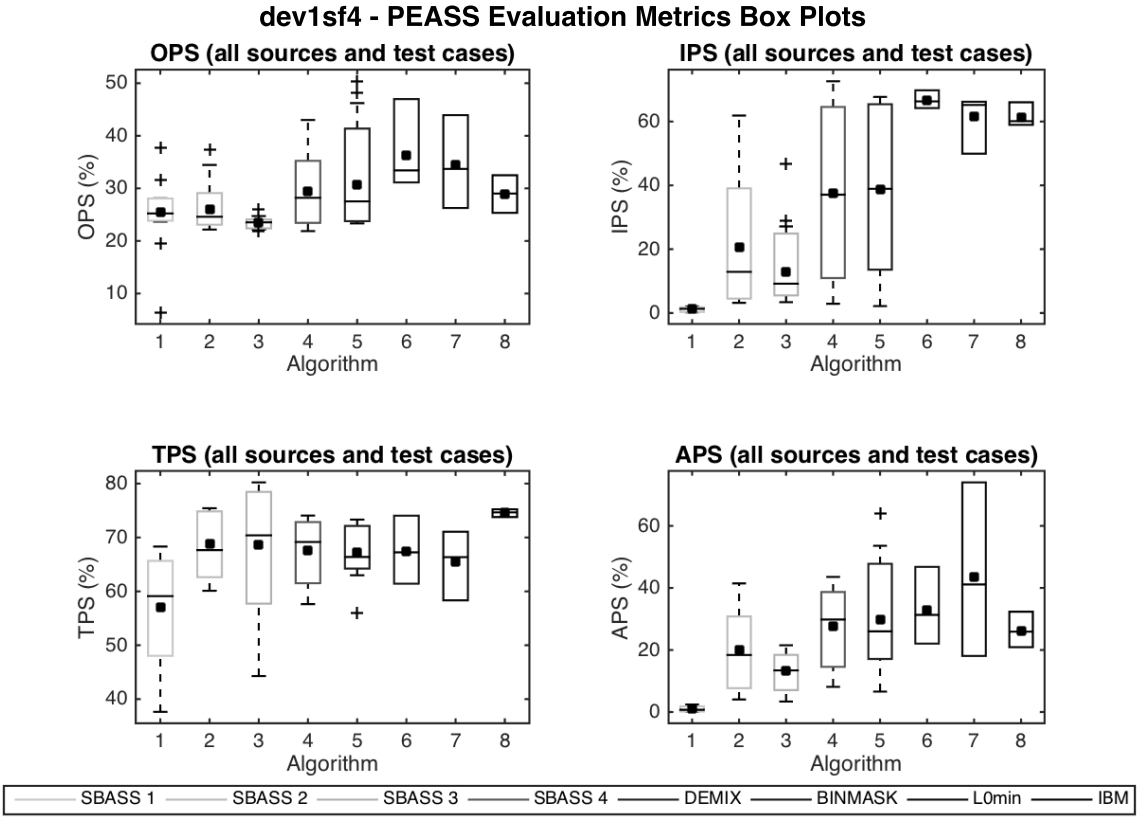


Figure 6.15: PEASS evaluation metrics for 'dev1sf4' mixture.

6.5 Summary and Final Discussion of Results

This chapter dealt with the testing and evaluation of the source separation algorithms presented in section 5.2. In particular the proposed algorithms were evaluated using common source separation evaluation metrics and compared with the DEMIX algorithm, which is an MP-based source separation algorithm and STFT-based algorithms including L0min which is based on ℓ^τ minimisation, BINMASK which is based on time-frequency binary masking and the IBM algorithm which uses the binary masks of the original sources to perform separation.

During analysis of the results certain trends appeared with regards to the performance of the MP-based and STFT-based algorithms. In all cases it was found that MP-based algorithms (and especially SBASS4 and DEMIX) achieved higher SDR and SIR but lower ISR and SAR scores compared to the STFT-based algorithms. When comparing the objective evaluation measures the results were mixed but in many cases the MP-based algorithms achieved similar (or better) evaluation scores with the STFT-based algorithms with an increasing number of iterations. It should be noted that the tests performed are not meant to provide a robust, statistical evaluation of the algorithms but rather inspect the behaviour of the proposed algorithms and verify their correct operation.

Another point to be made is that the separation performance metrics that were used are single numbers describing the 'global' quality of the separated sources and extra care must be taken when interpreting these results. Although these metrics provide a good overall quality description of an algorithm that can be used to quickly identify bad implementations (as was the case with algorithm SBASS1) they should not be taken for granted, especially when the differences between the metrics of two algorithms are small. It is a fact that the best possible evaluation occurs by auditioning the estimated sources and by performing appropriate listening tests. As such the reader is encouraged to listen to the produced results which are provided with the electronic copy of this dissertation.

Regardless of the inherent risks of interpreting global performance measures, the results obtained can allow us to make some generalisations, first of all about the proposed algorithms and secondly their performance against similar separation methods. In particular, the following points can be made:

- Out of all four proposed algorithms SBASS4 achieves the best performance overall with SBASS1 failing to converge in every case and algorithms SBASS2 and SBASS3 producing mixed results. This is a clear indication that the first three approaches should not be pursued any further.
- The SBASS4 algorithm is comparable to the DEMIX algorithm with both algorithms achieving similar scores. This is a very important outcome since it reinforces even further the suitability of the proposed GMP algorithm.
- MP-based algorithms are comparable to STFT-based algorithms for the problem of sound source separation.
- In the examples presented here, the L0min algorithm outperforms the BINASK in most cases and obtains similar (and in some cases better) scores as the IBM algorithm.

Chapter 7

Conclusion

In the previous chapters a new matching pursuit variant was introduced, described and applied to source separation of linear instantaneous stereophonic mixtures. While doing so a number of contributions were made and potential areas for future research were identified. This section reviews the main contributions that were made and areas for future work are considered.

7.1 Contributions

The main contribution of this work is the introduction of guided matching pursuit (GMP) which is a variant of the MP algorithm. The main modification of the standard algorithm is the inclusion of a pre-processing in the main sequence of events. Although this is a slight modification it has major consequences especially regarding the implementation of the algorithm. The main idea is to perform some kind of analysis to a signal, before decomposition takes place, and extract important features that can be used to dynamically create mini-dictionaries at each iteration step of the algorithm. This approach is completely different to the conventional MP approach where usually the first step is to decide and create a dictionary. Even if the dictionary is implied by a fast transform, like for example the STFT or MDCT, the user still needs to decide on its creation before signal decomposition starts. This leads to fixed dictionaries in a sense that they remain the same throughout the duration of the decomposition. In contrast, GMP uses the information obtained at the pre-processing step to create dynamic mini-dictionaries that are expected to correlate well with particular supports of the signal thus leading to an adaptive approach of dictionary creation. The kind of processing to be performed during the pre-processing step could be anything that provides important information about the analysed signal and can be as simple as an energy map or as complex as a source separation algorithm. The correct choice for a pre-processing step depends on the application at hand and the available computational power and memory. Several useful types of processing were described in section 3.4. The idea of guidance is not new and has been exploited in one way or another in several MP variants but as far as the author knows it has never been explicitly stated the way it has in this work.

Another major contribution of this work is the development and implementation of a MATLAB® toolbox, called GMPC, capable of performing GMP. Although during initial development GMPC had a specific application in mind, throughout the last three years it has evolved into a more

generic framework tailored to the investigation of MP in general. GMPC was designed using advanced MATLAB® OOP practices and is loosely based on MATLAB® 's System Objects; a set of classes tailored to real time digital signal processing. As such GMPC has solid foundations and is modular and highly parametrisable. The fact that it is written in MATLAB® makes it an attractive solution for researching and prototyping. Apart from its ability to perform pretty much all GMP algorithms that were described in this work it can be also extended to incorporate any available MP algorithm; that is its design is generic enough so that any MP implementation can become part of the toolbox. An example of this flexibility is the inclusion of the MPTK software package and all the STFT-based source separation algorithms (mentioned in chapter 6) in the main interface of the toolbox. All these algorithms are invoked through a common interface thus leading to a seamless experience for the end user. The toolbox is generic enough that one could perform a decomposition using the MPTK binaries (from within the toolbox) and reconstruct the signal using a different algorithm. All MP algorithms share similar settings and GMPC s the glue that keeps everything in place. GMPC also allows for the comparison of various MP implementations, and all results in chapters 4 and 6 were generated using a powerful interface that aids and streamlines the creation of experiments thus promoting reproducible research. GMPC also has poweful visualisation capabilities some of which are demonstrated in section 2.3.4. Note that all MP related figures and more were produced using this toolbox. Finally since the design is based on MATLAB® 's System Object all the infrastructure for performing real-time MP is in place.

During this work a number of smaller contributions were also made. First of all the short-time matching pursuit (STMP) was introduced and formally expressed using the overlap-add decomposition. Several issues regarding this representation were discussed and explored. Again, although the idea of frame based MP is not new it has never been explicitly stated and as far as the author knows, treating the STMP representation in similar ways to STFT has not been attempted. Processing of long signals was also discussed and two new names for such processing were proposed, namely LFMP and VLMP. These three approaches are generic enough to describe any kind of signal segmentation which is an important subject when considering an MP decomposition but rarely mentioned in the literature. In a similar fashion various ways to process multi-channel signals were described and categorised into joint, disjoint and weakly joint (or constrained disjoint) multi-channel processing.

Another small contribution was the unifying treatment of analytic atoms. In section 3.6 a standardised approach to atom creation is described and several atom types are defined. A new atom type that has not been used in the literature was also introduced namely the AFM atom (Arbitrary Frequency Modulated) which is generic enough to capture any type of frequency modulation within a signal. A possible implementation of the AFM atom within the GMP framework was also proposed.

Finally in chapter 5 the application of GMP on various source separation problems was discussed and demonstrated. The use of GMP in the separation and classification of crackles from pulmonary sounds is a novel approach that has not been explored in the literature.

7.2 Closing

In this work a new matching pursuit variant was described and its application to sound source separation was demonstrated and evaluated. Also the design and implementation of a new MATLAB® toolbox were discussed and the toolbox was used to evaluate GMP's performance. In this chapter the contributions that were made during this work are highlighted and potential future directions are proposed. GMP is an interesting algorithm with great potential and the accompanying toolbox provides a generic framework for MP experimentation which hopefully could prove useful to MP research communities.

Appendix A

GMPC Performance Testing - Extra Results

A.1 Pre-Processing Step Testing - Extra Results

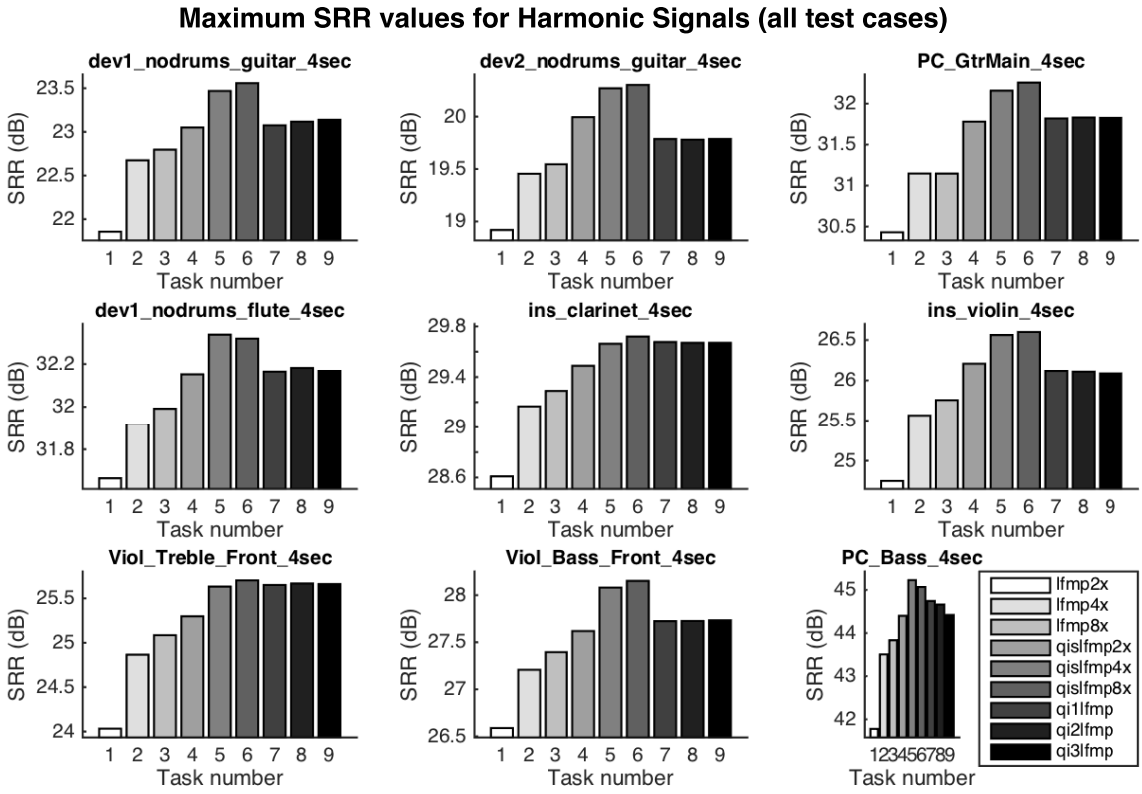


Figure A.1: Pre-processing step testing. SRR values of harmonic signals.

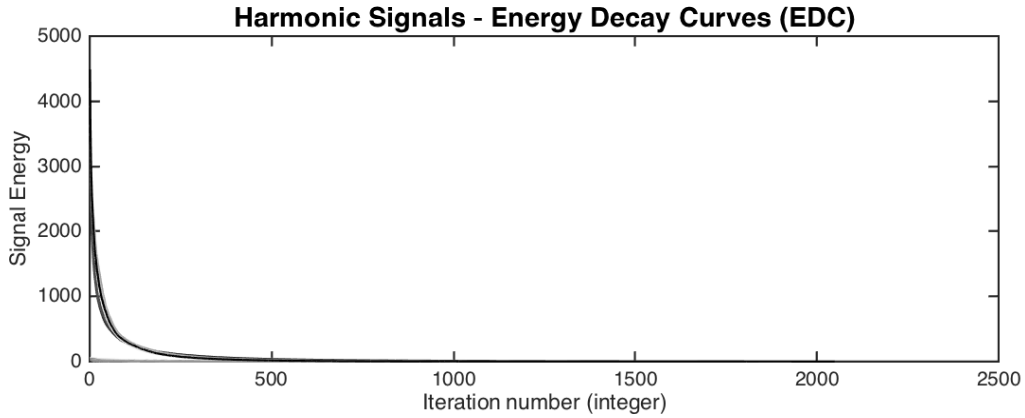


Figure A.2: EDC of harmonic signals.

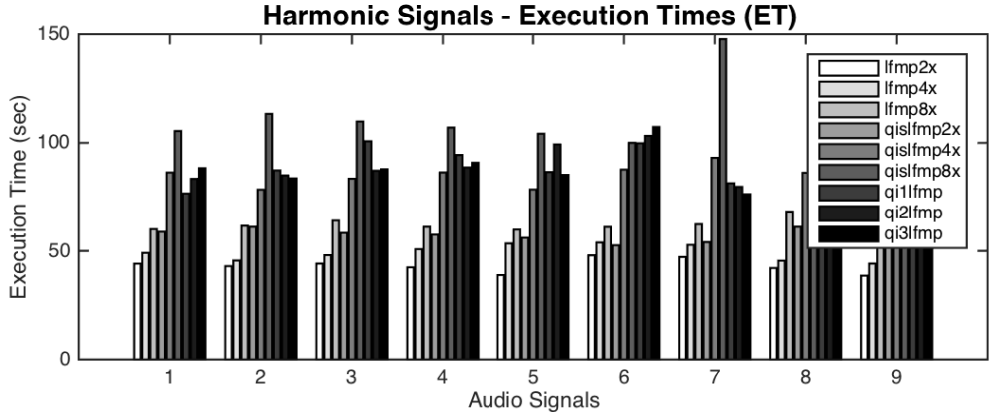


Figure A.3: ET of harmonic signals.

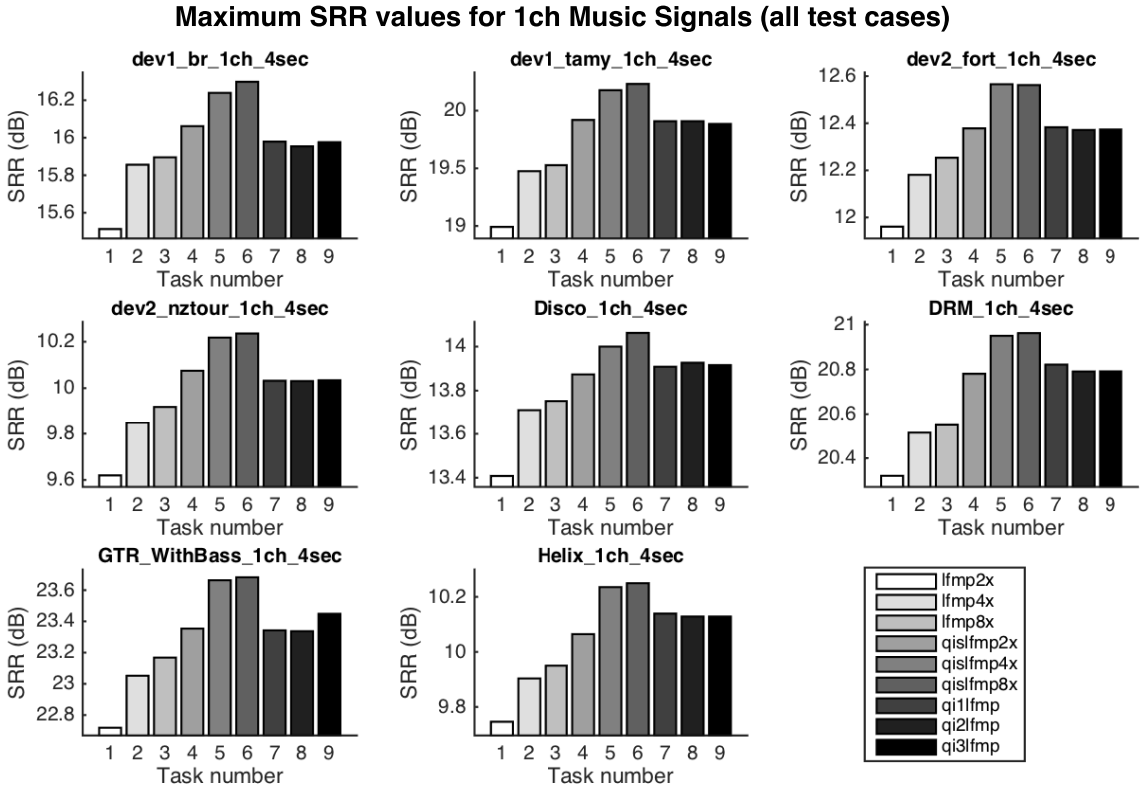


Figure A.4: SRR values of mixture signals.

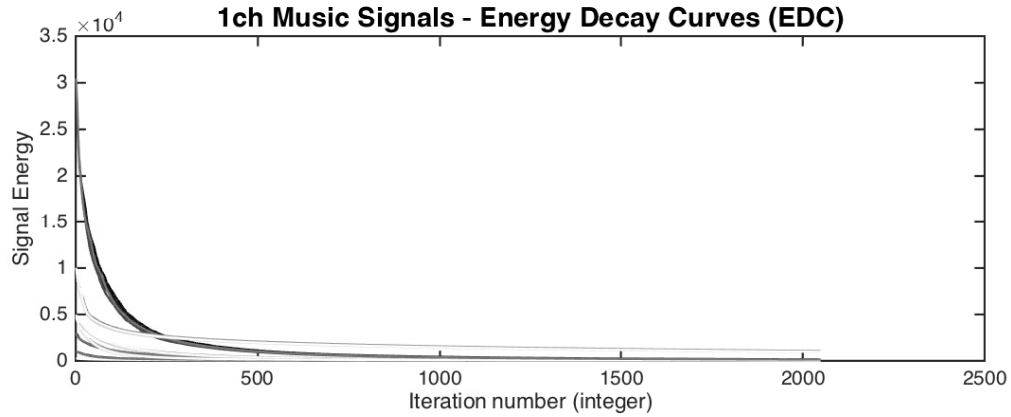


Figure A.5: EDC of mixture signals.

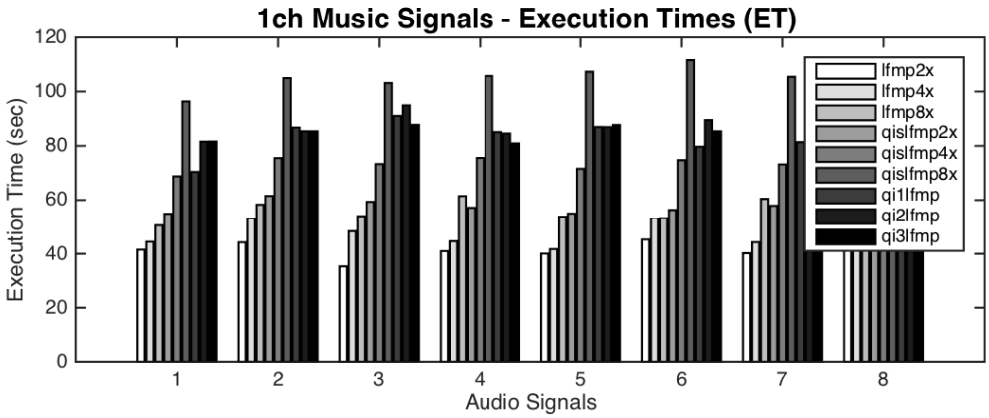


Figure A.6: ET of mixture signals.

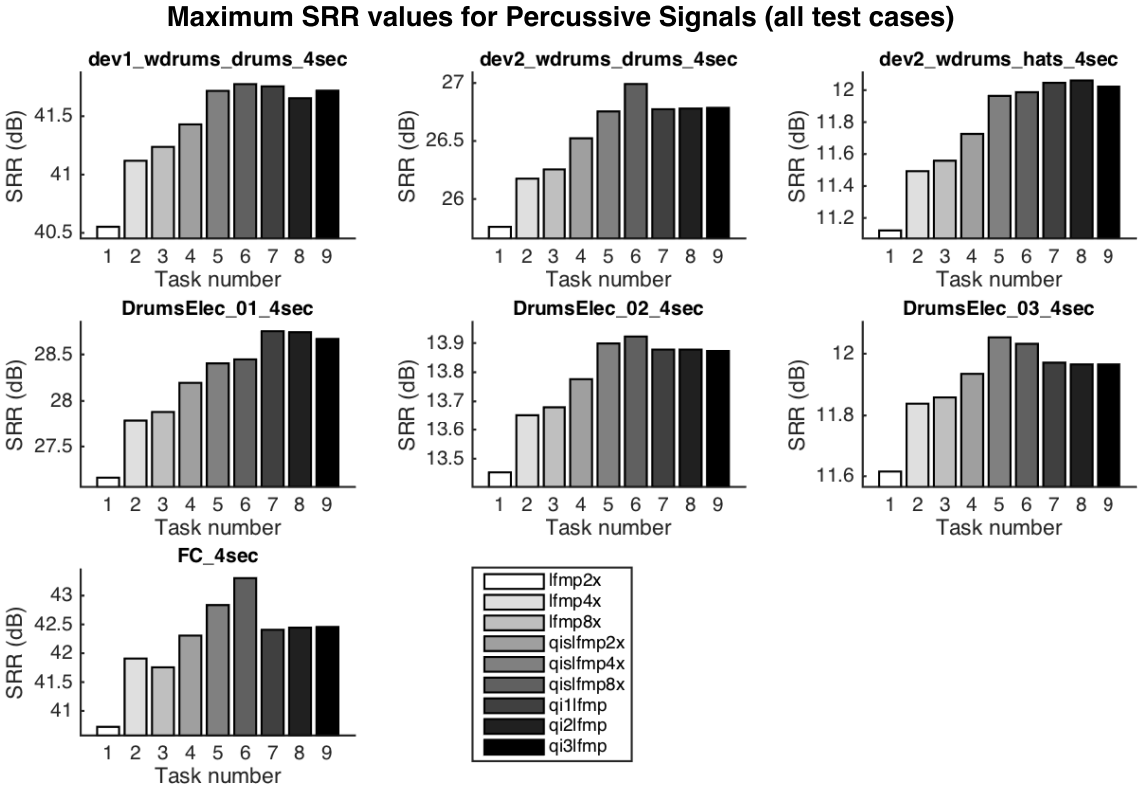


Figure A.7: SRR values of percussive signals.

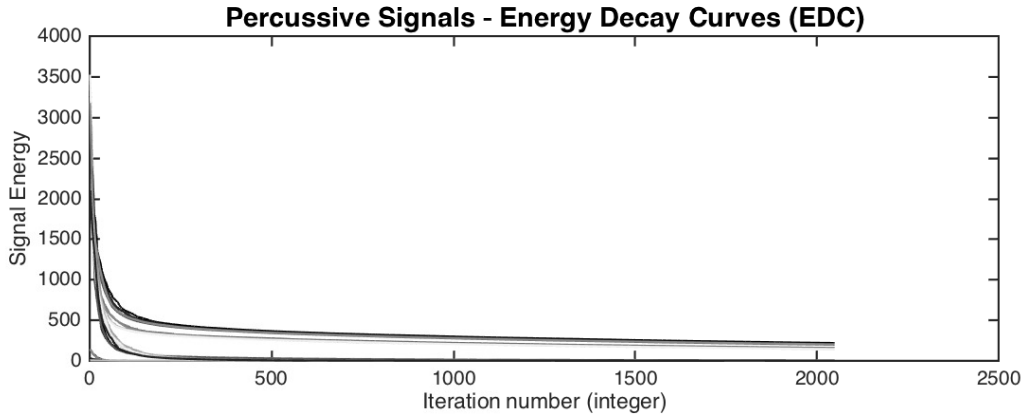


Figure A.8: EDC of percussive signals.

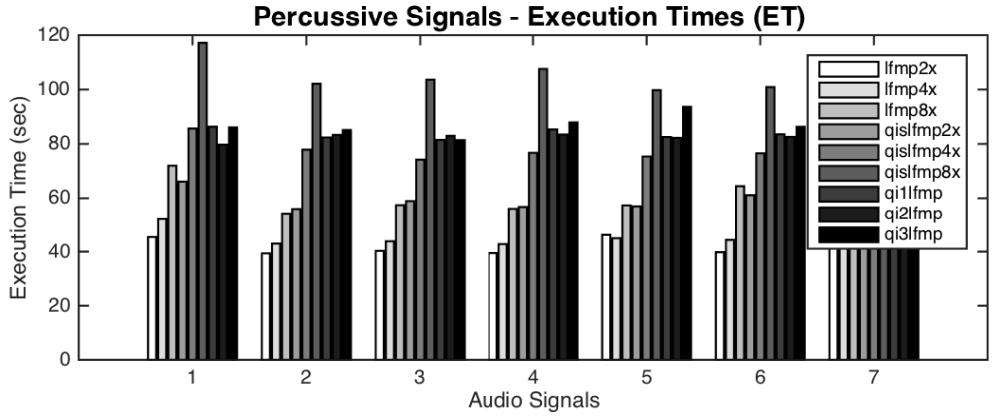


Figure A.9: ET of percussive signals.

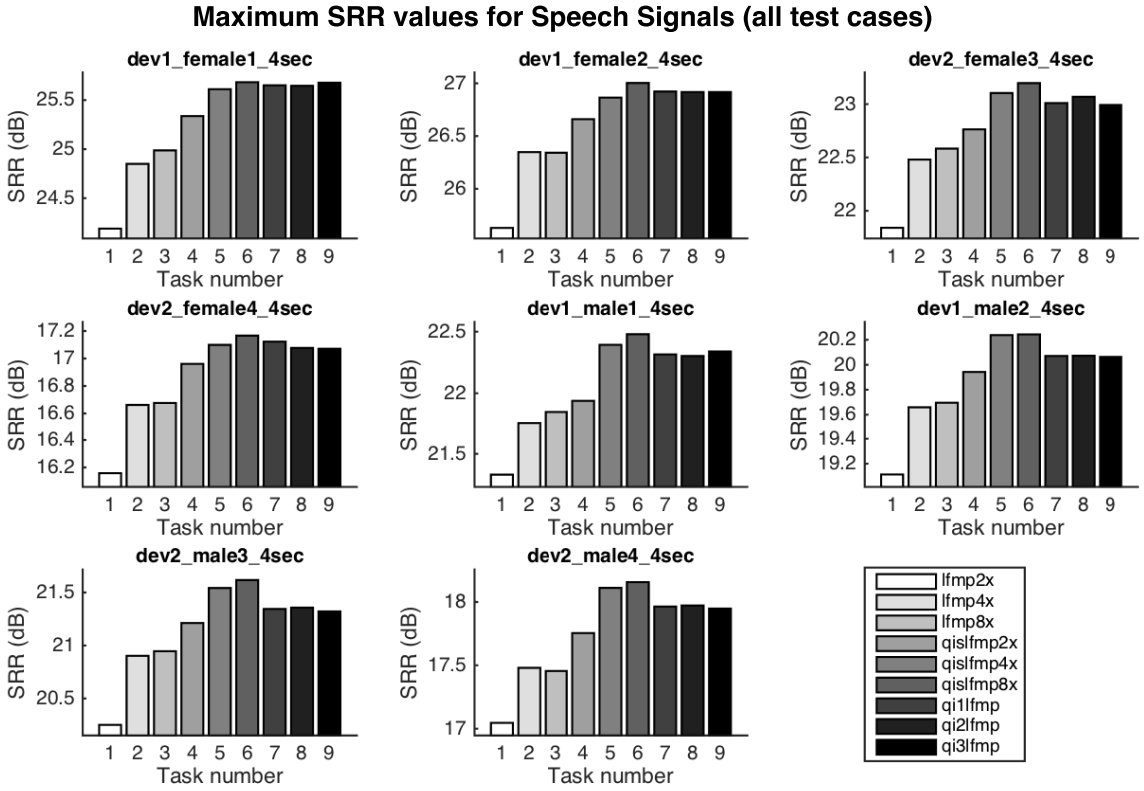


Figure A.10: SRR values of speech signals.

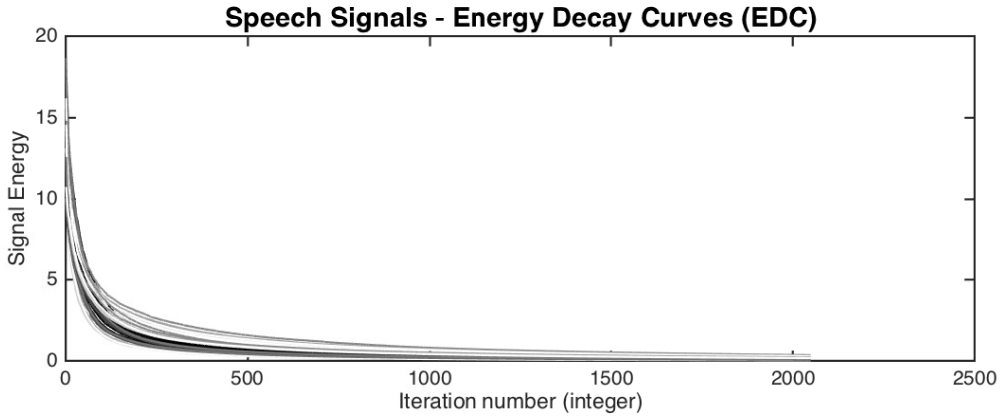


Figure A.11: EDC of speech signals.

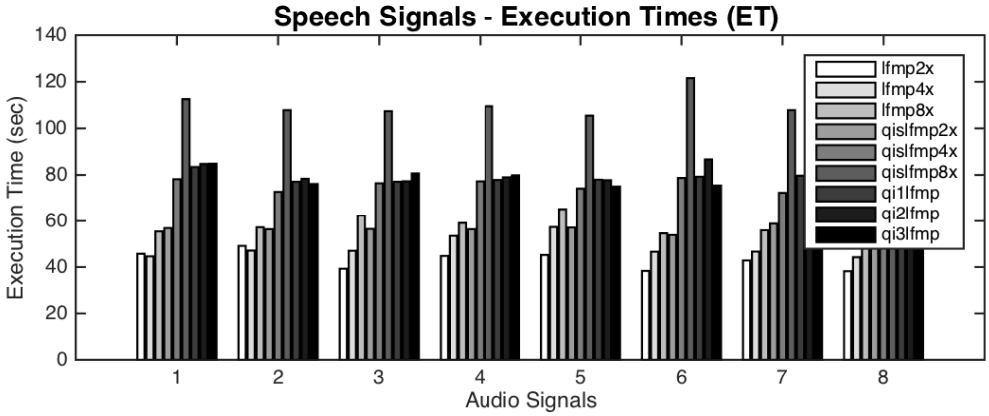


Figure A.12: ET of speech signals.

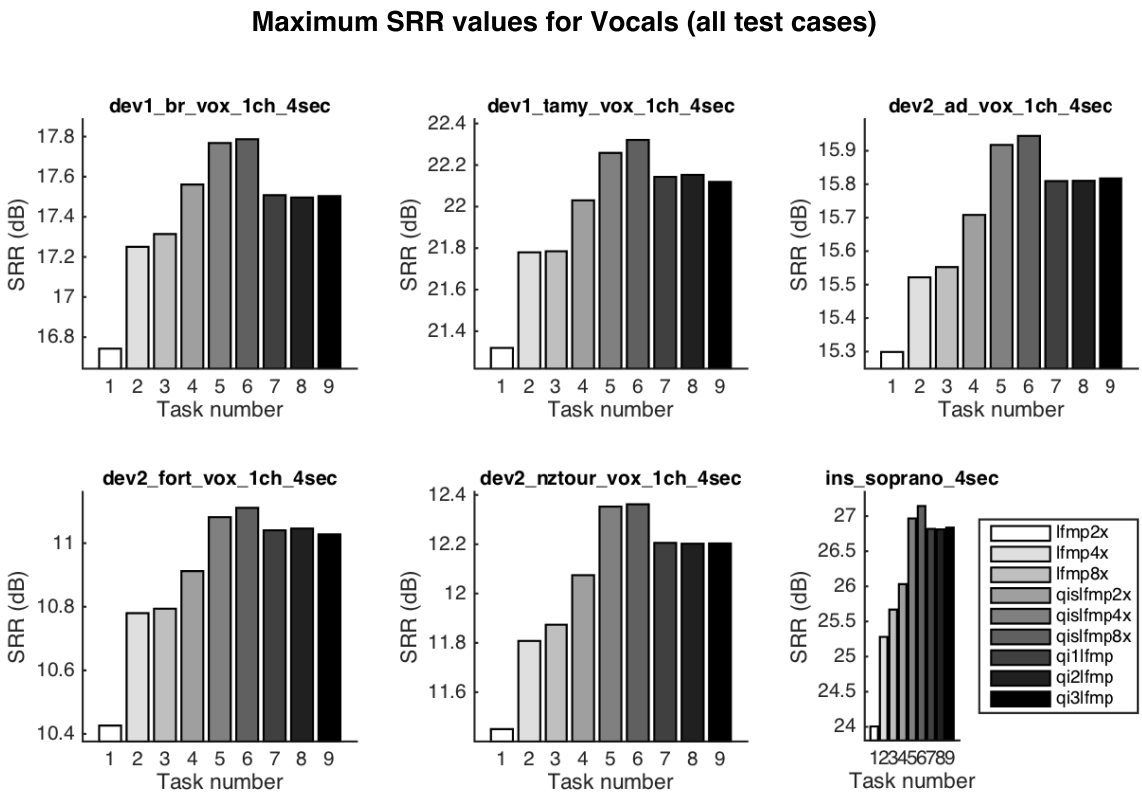


Figure A.13: SRR values of vocal signals.

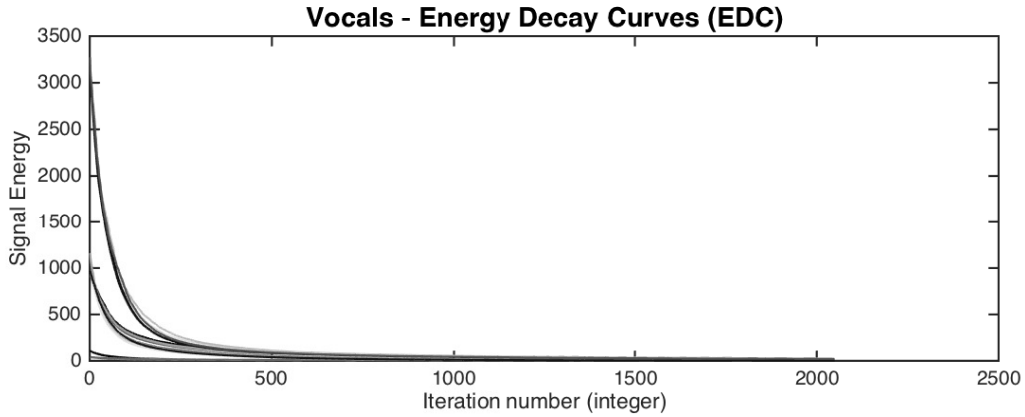


Figure A.14: EDC of vocal signals.

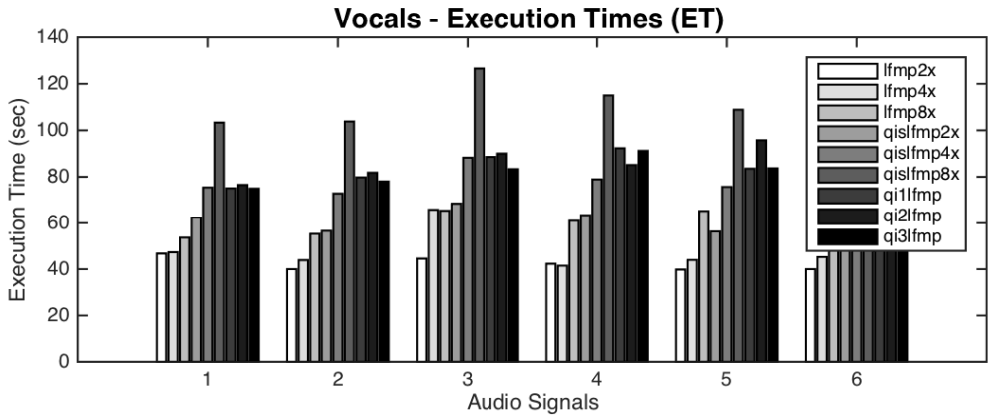


Figure A.15: ET of vocal signals.

A.2 Dictionary Testing - Extra Results

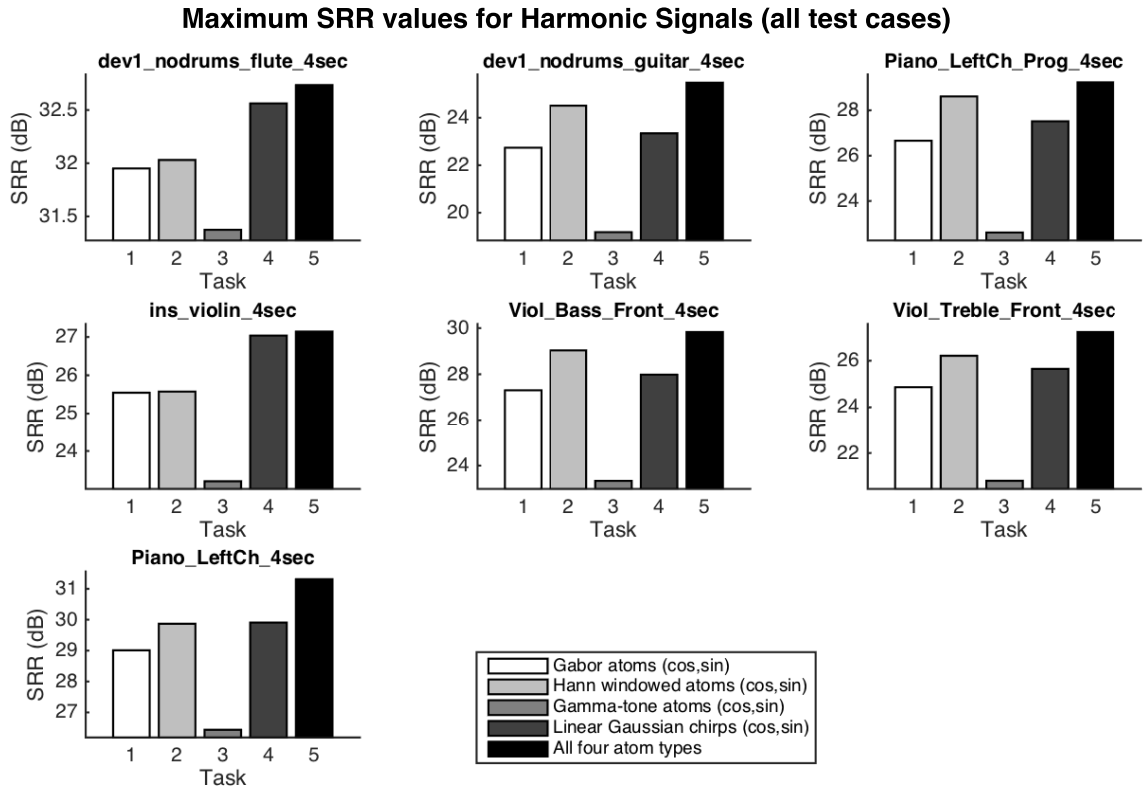


Figure A.16: Dictionary testing. SRR values of harmonic signals.

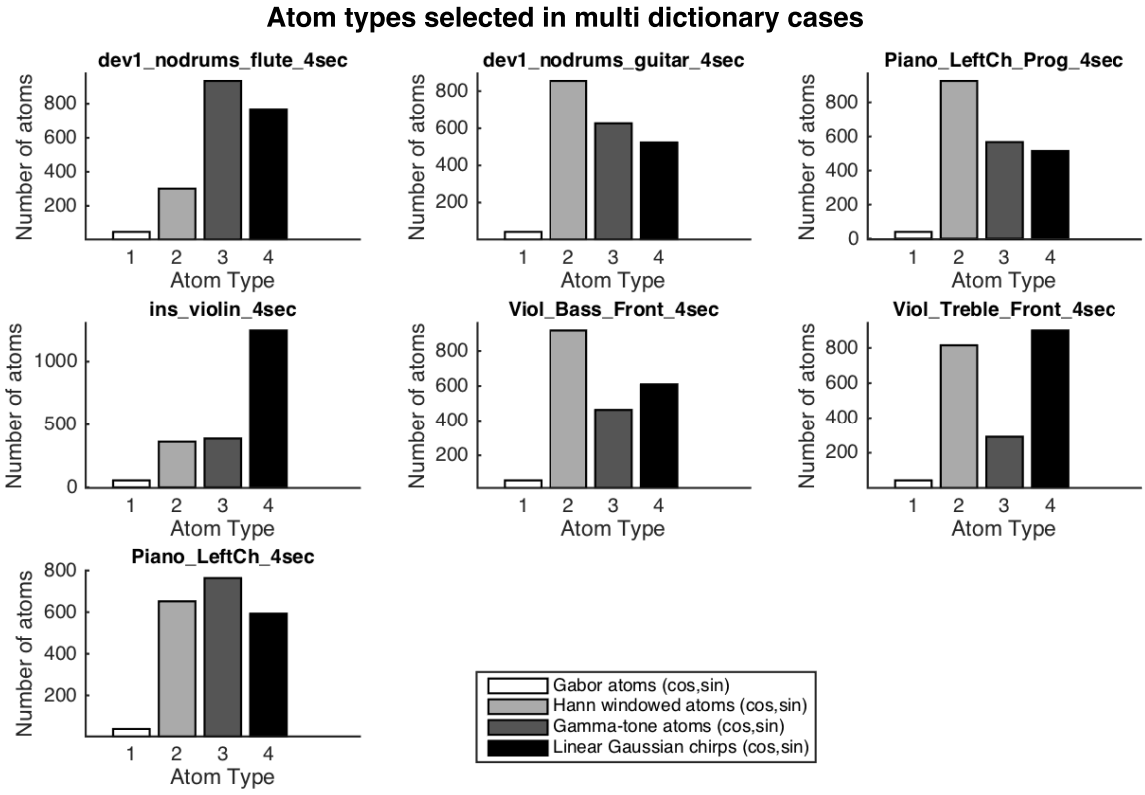


Figure A.17: Dictionary testing. Selected atom types in multi-dictionary case of harmonic signals.

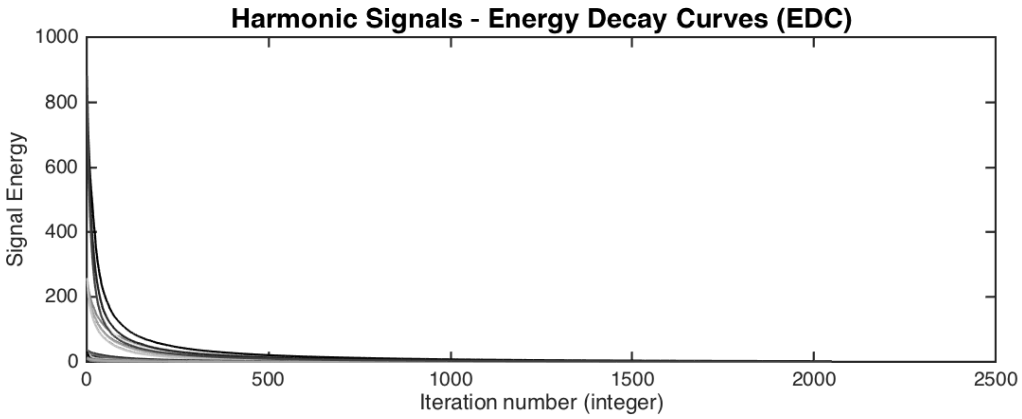


Figure A.18: Dictionary testing. EDC of harmonic signals.

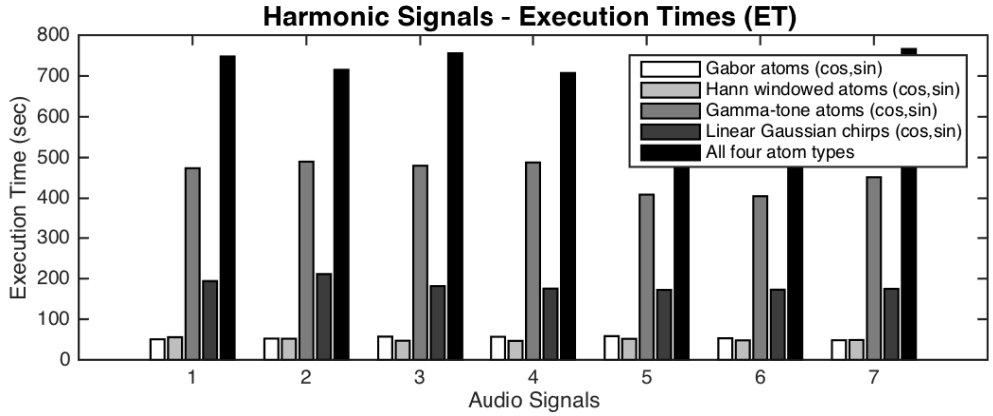


Figure A.19: Dictionary testing. ET of harmonic signals.

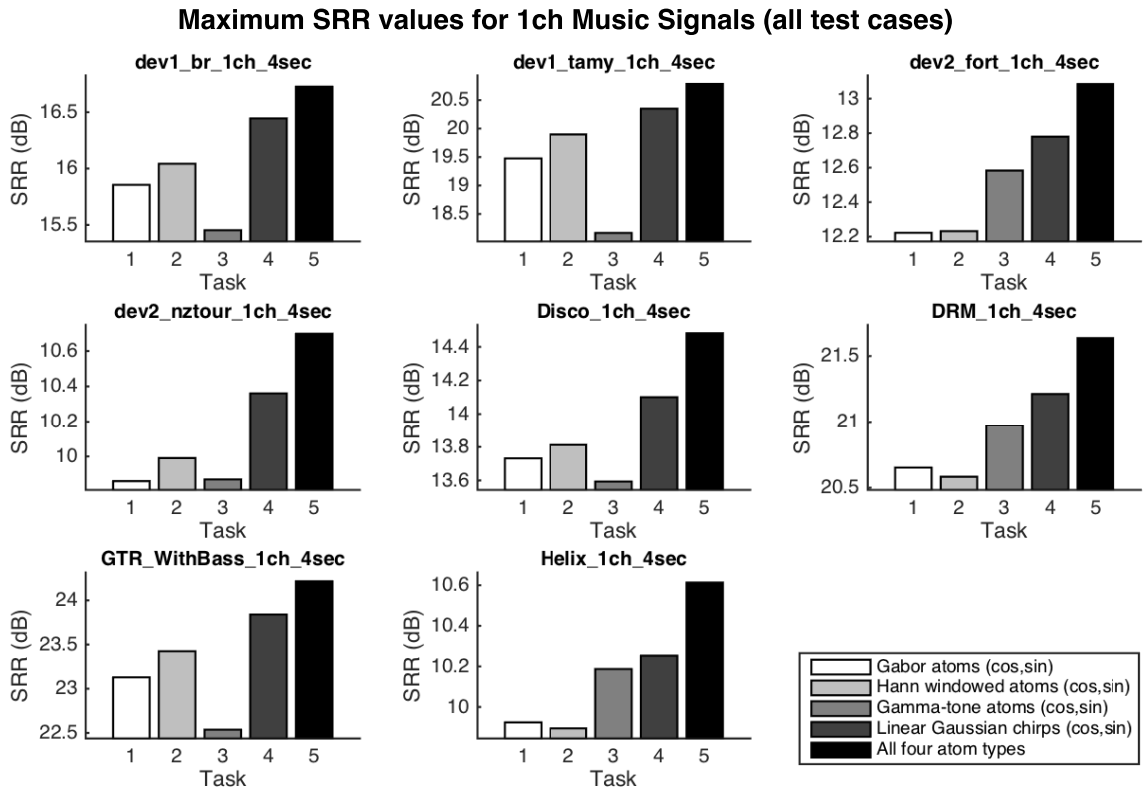


Figure A.20: Dictionary testing. SRR values of mixture signals.

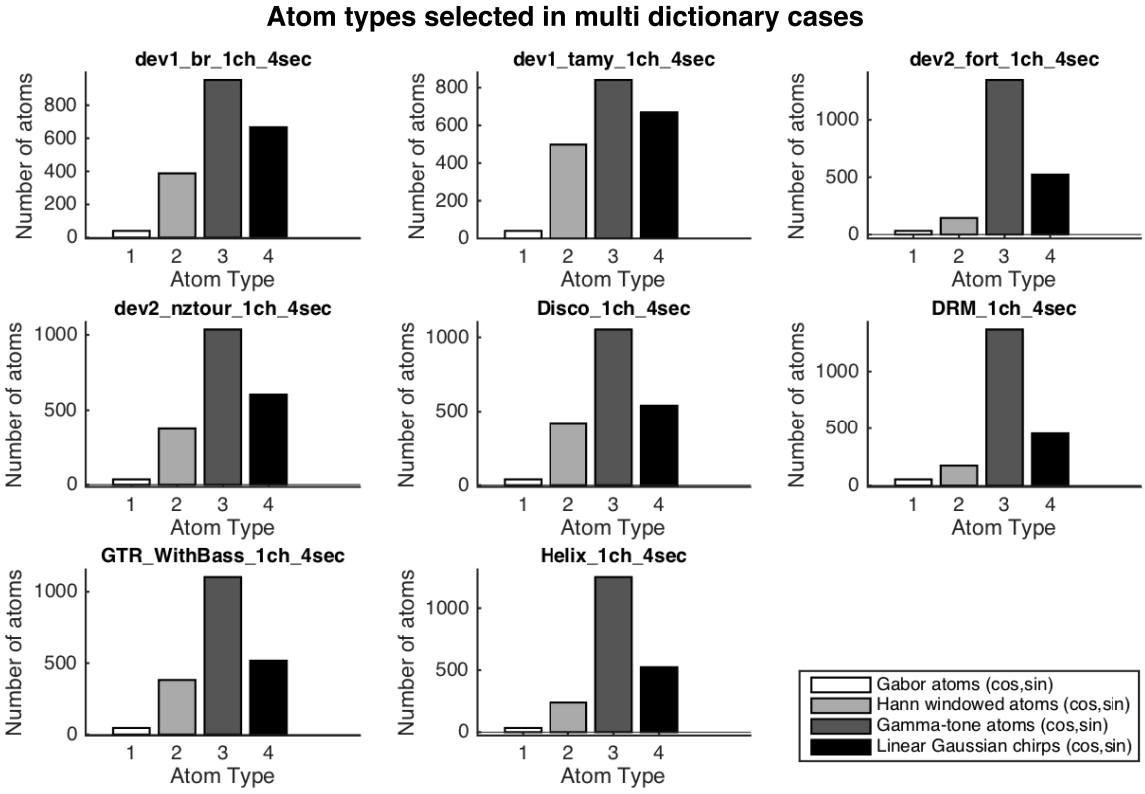


Figure A.21: Dictionary testing. Selected atom types in multi-dictionary case of mixture signals.

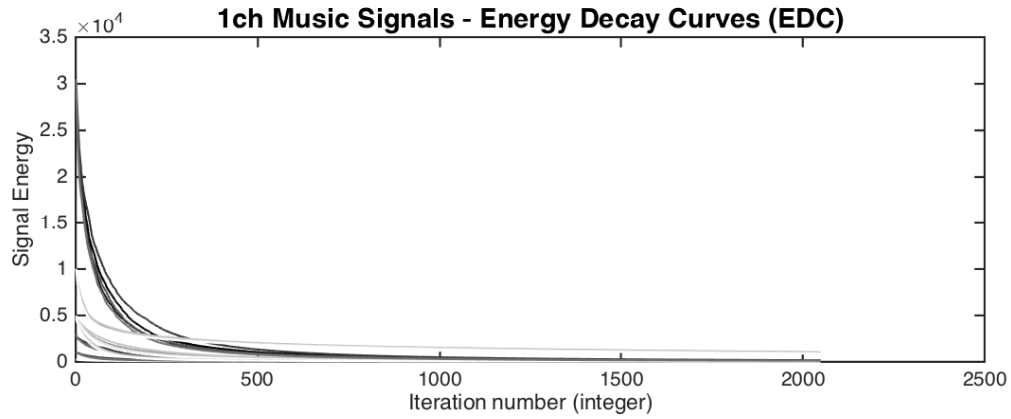


Figure A.22: Dictionary testing. EDC of mixture signals.

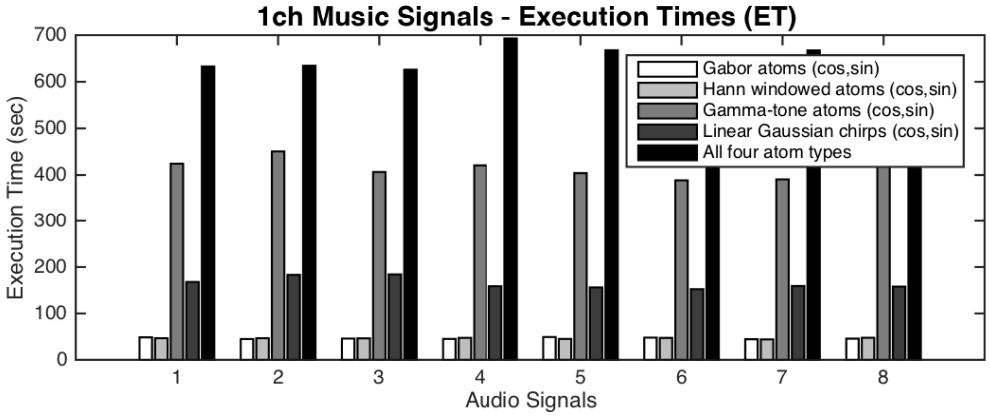


Figure A.23: Dictionary testing. ET of mixture signals.

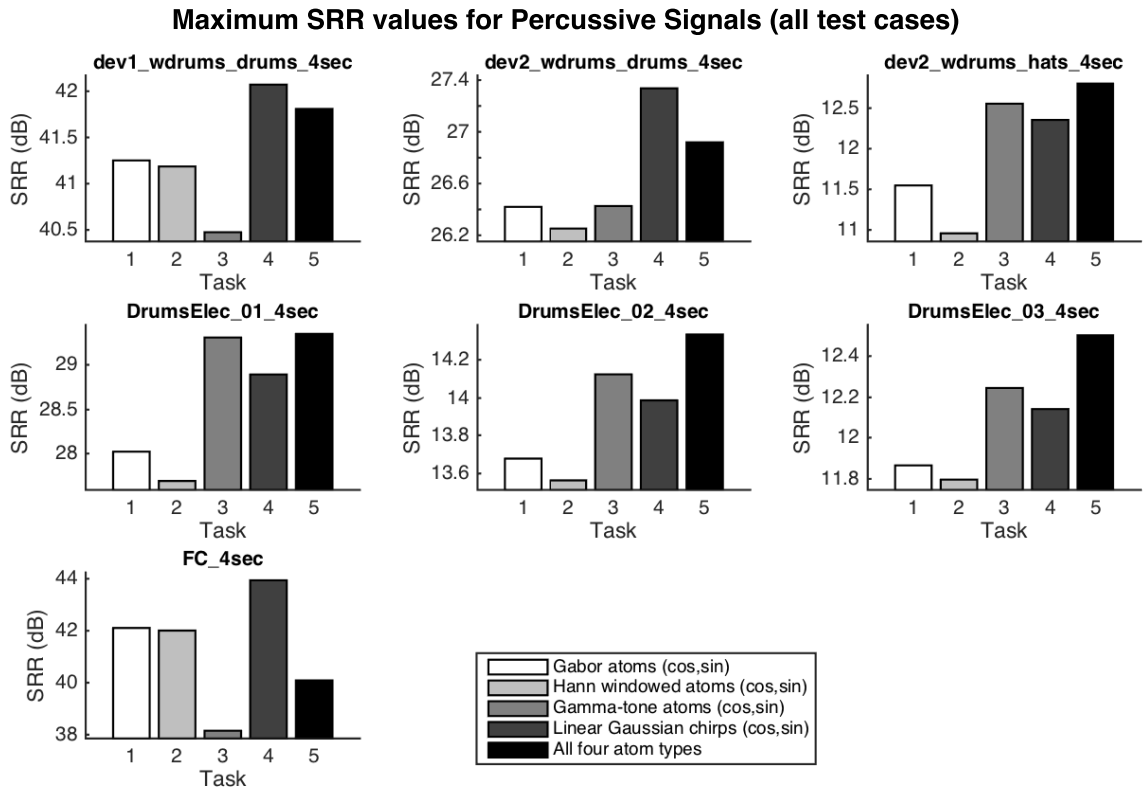


Figure A.24: Dictionary testing. SRR values of percussive signals.

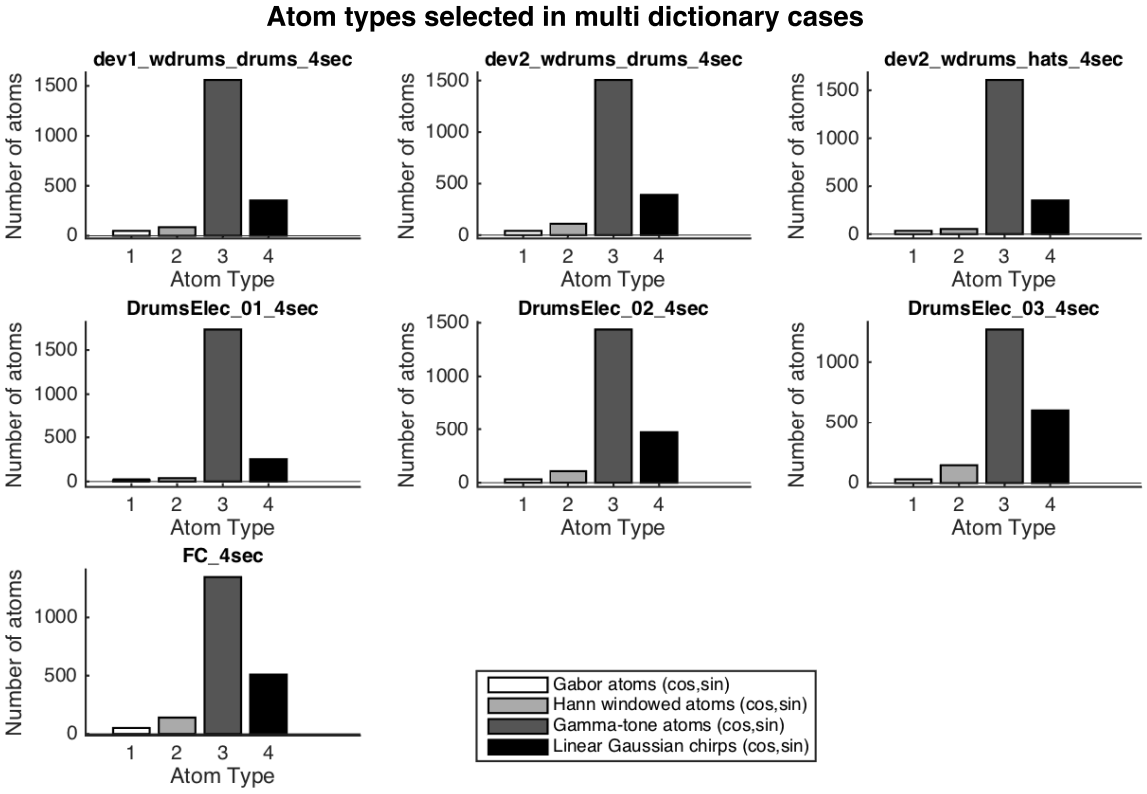


Figure A.25: Dictionary testing. Selected atom types in multi-dictionary case of percussive signals.

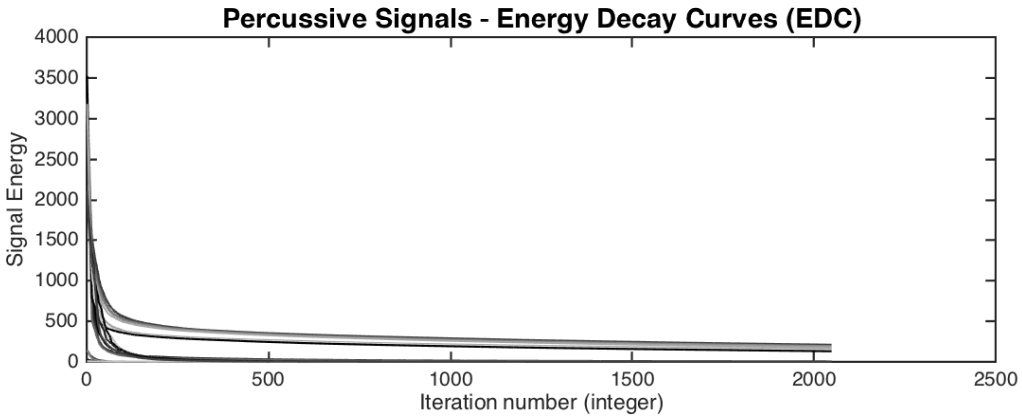


Figure A.26: Dictionary testing. EDC of percussive signals.

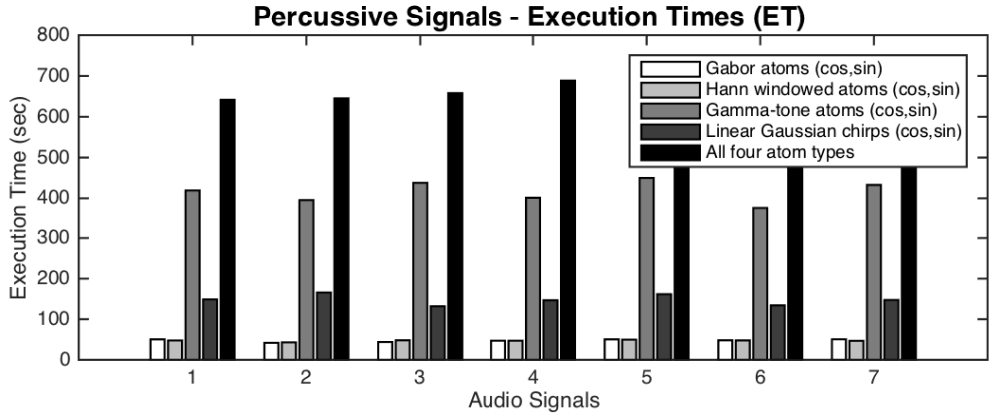


Figure A.27: Dictionary testing. ET of percussive signals.

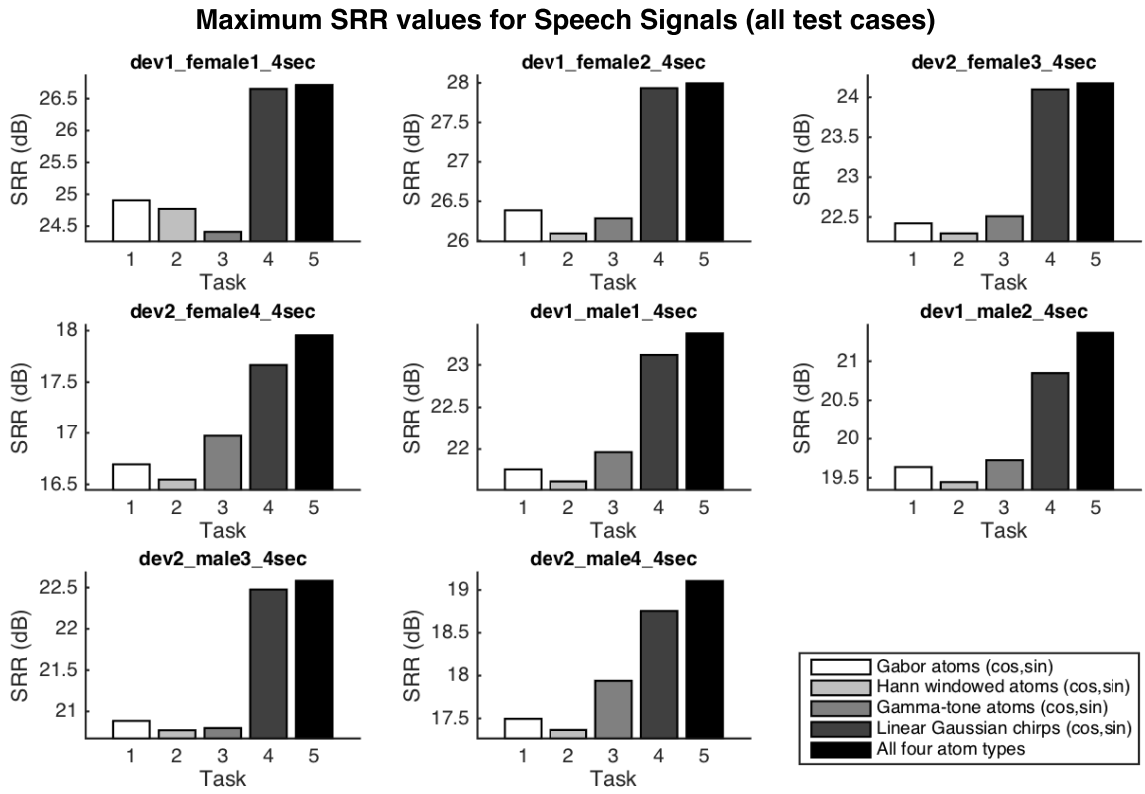


Figure A.28: Dictionary testing. SRR values of speech signals.

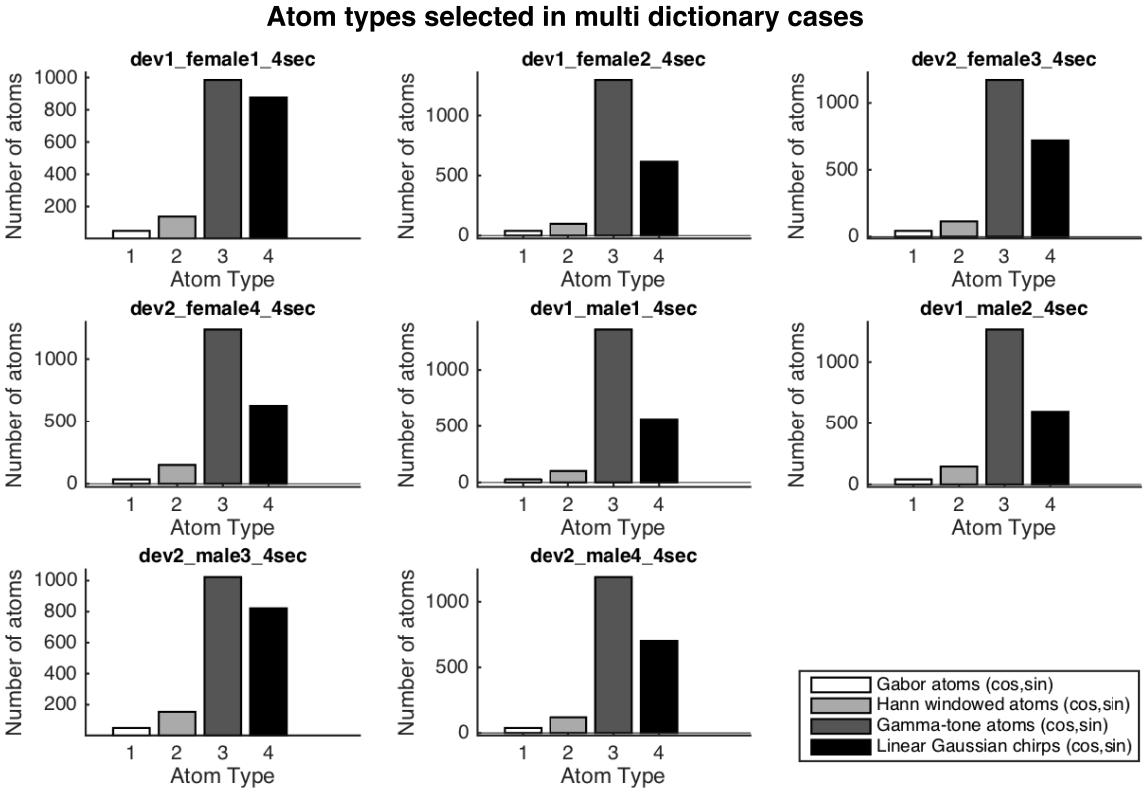


Figure A.29: Dictionary testing. Selected atom types in multi-dictionary case of speech signals.

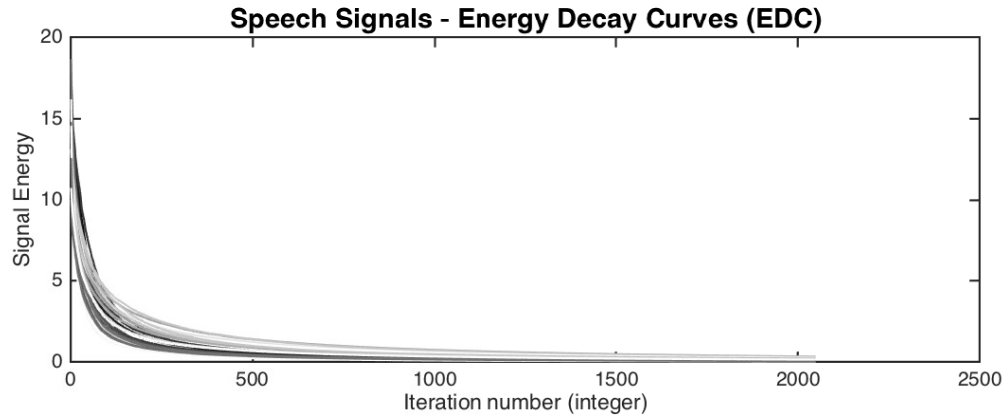


Figure A.30: Dictionary testing. EDC of speech signals.

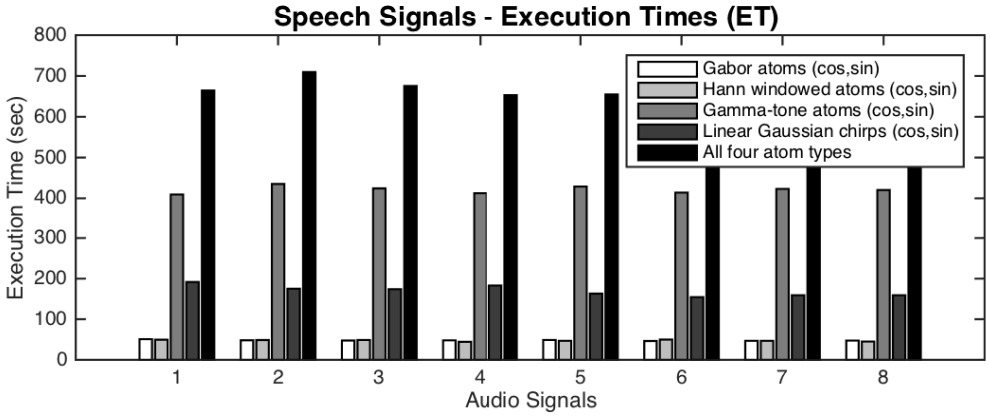


Figure A.31: Dictionary testing. ET of speech signals.

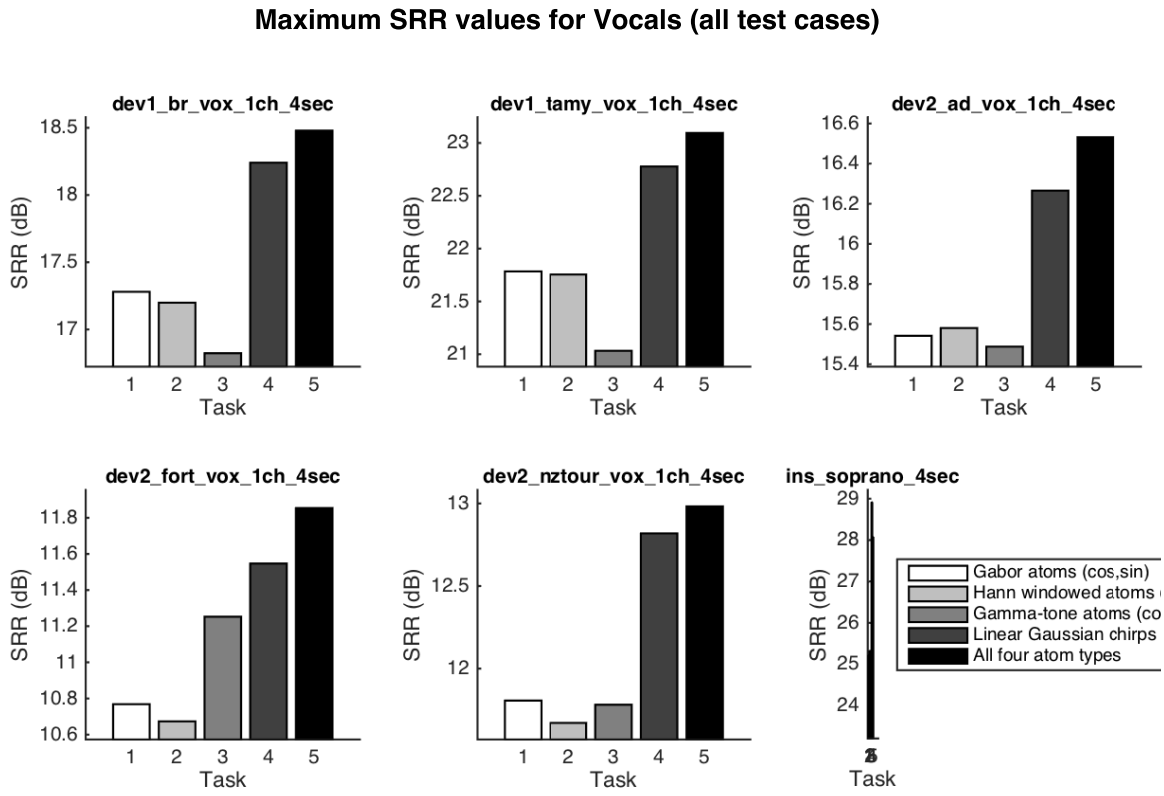


Figure A.32: Dictionary testing. SRR values of vocal signals.

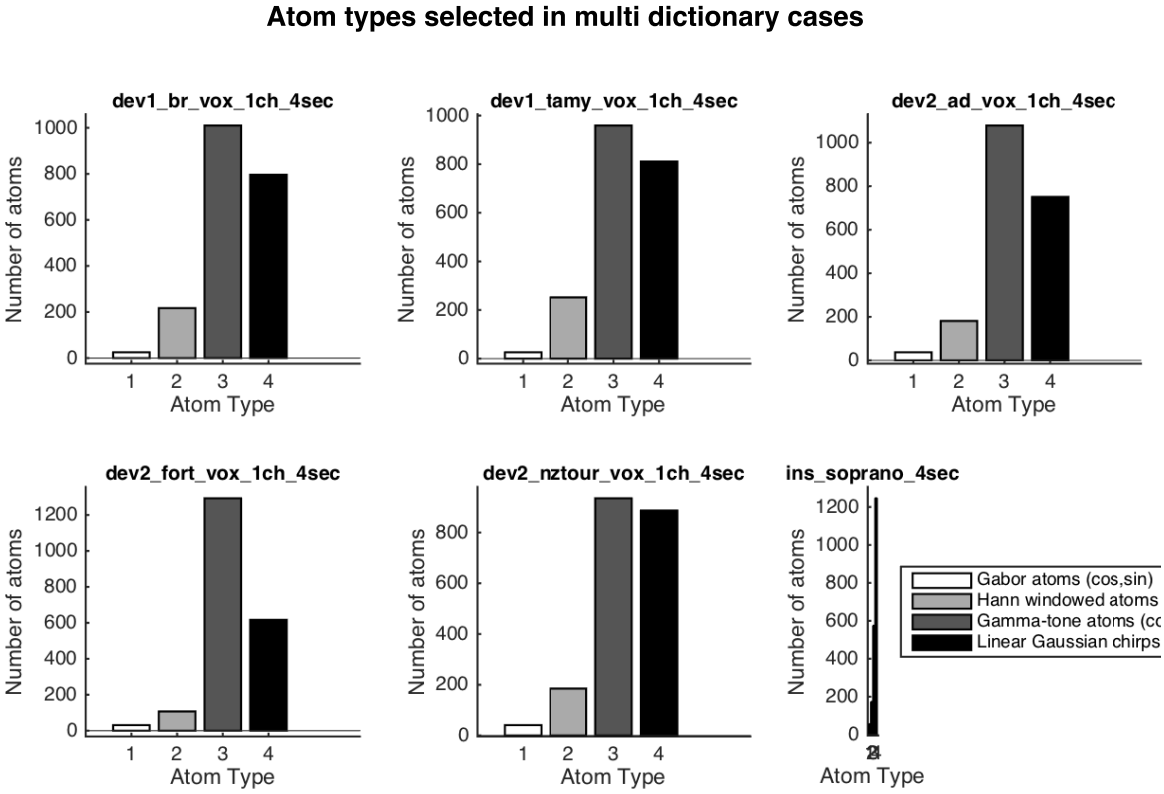


Figure A.33: Dictionary testing. Selected atom types in multi-dictionary case of vocal signals.

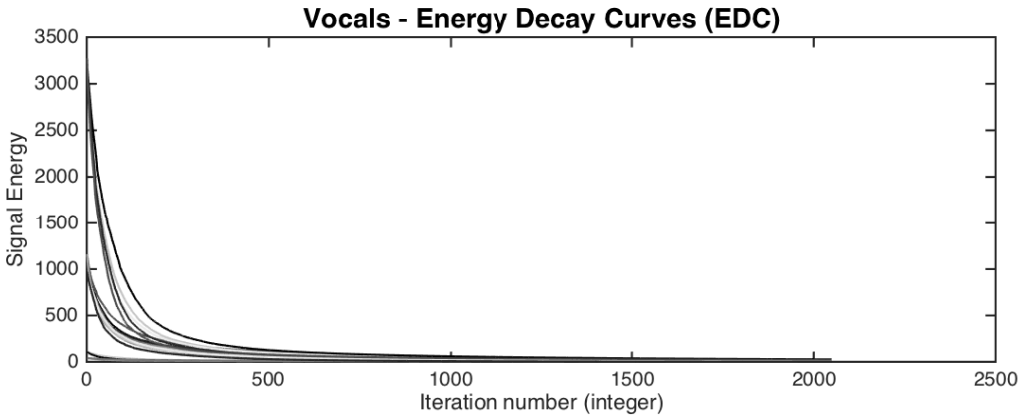


Figure A.34: Dictionary testing. EDC of vocal signals.

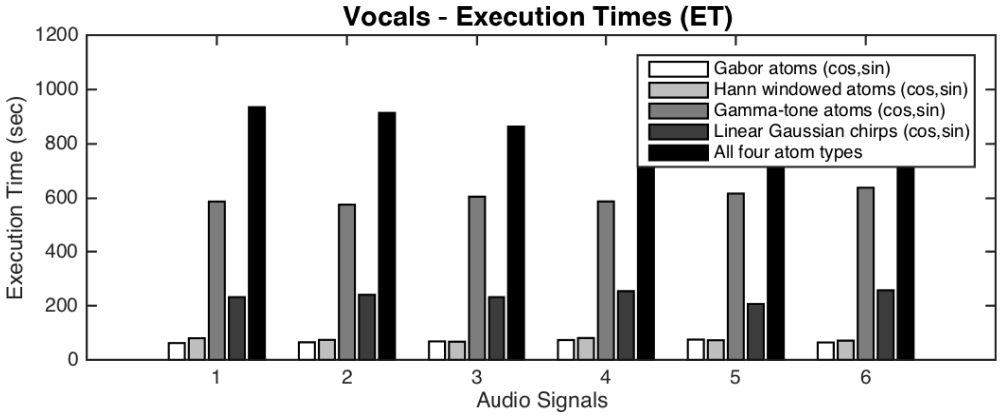


Figure A.35: Dictionary testing. ET of vocal signals.

A.3 GMP - MPTK Comparison Testing - Extra Results

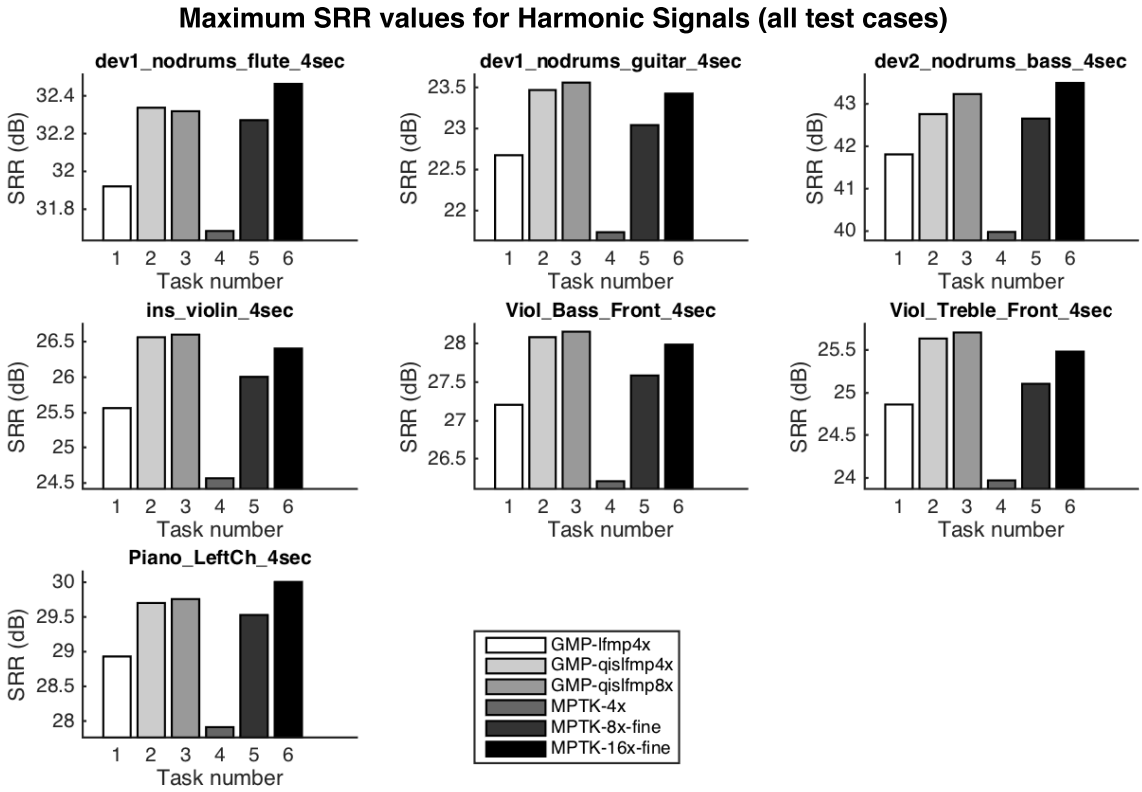


Figure A.36: SRR values of harmonic signals.

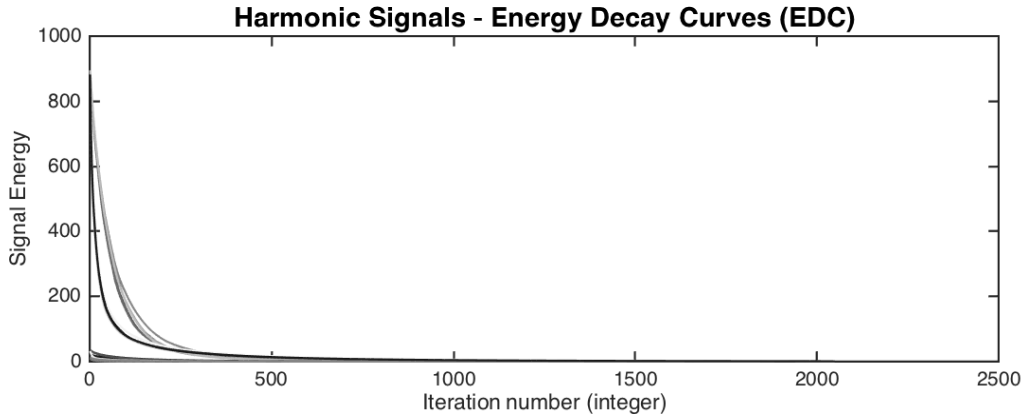


Figure A.37: EDC of harmonic signals.

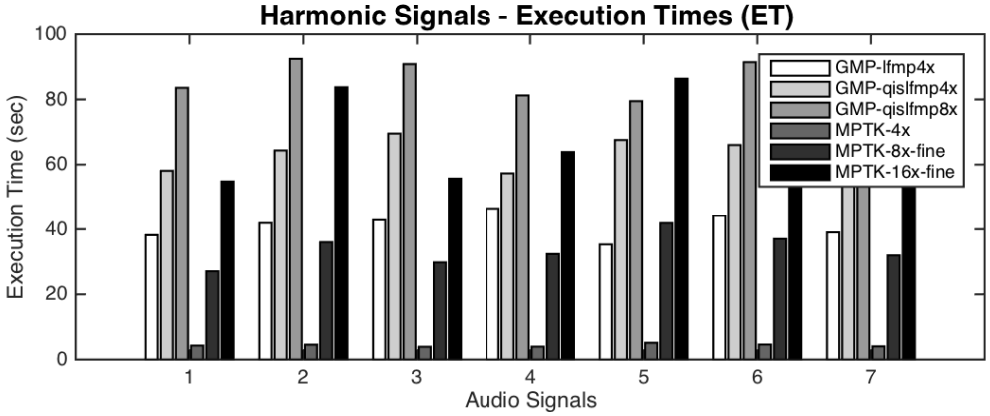


Figure A.38: ET of harmonic signals.

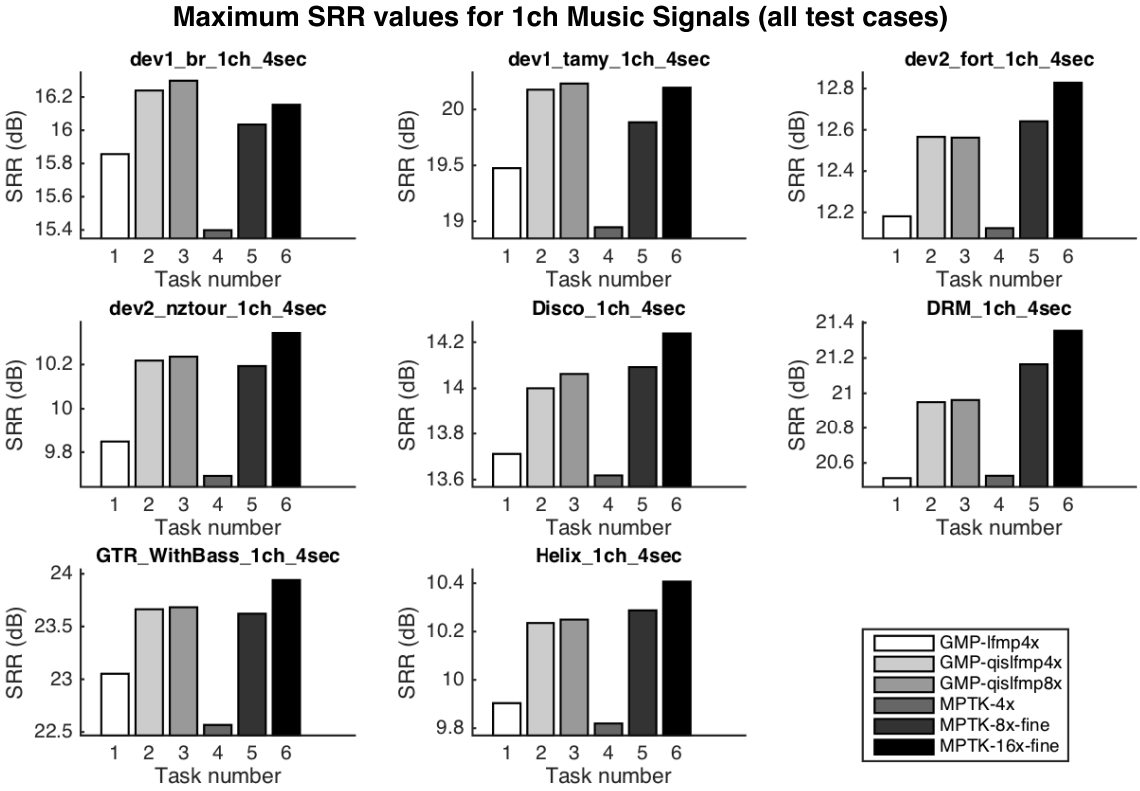


Figure A.39: SRR values of mixture signals.

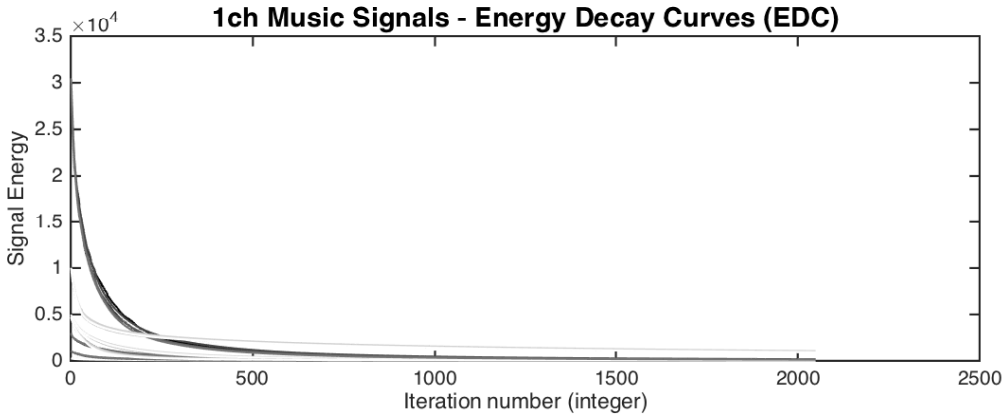


Figure A.40: EDC of mixture signals.

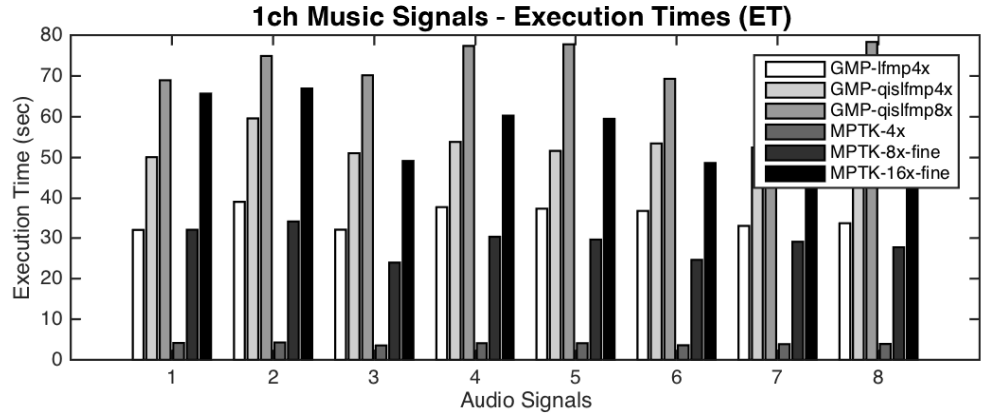


Figure A.41: ET of mixture signals.

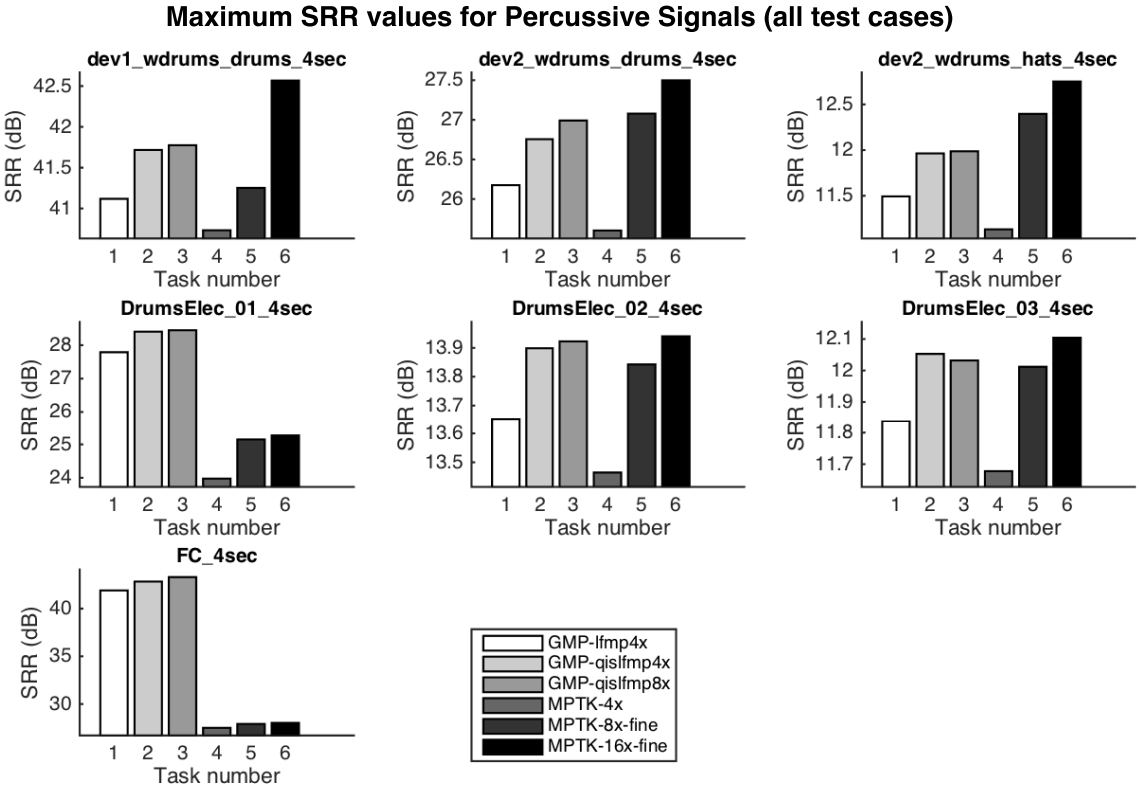


Figure A.42: SRR values of percussive signals.

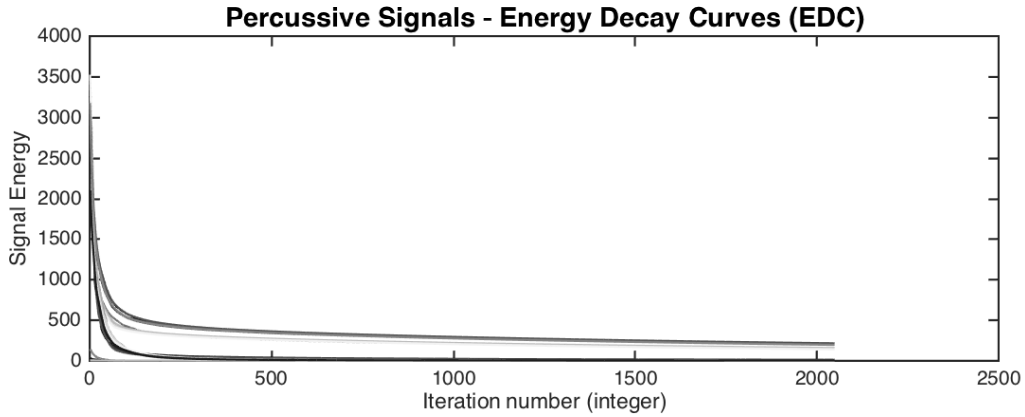


Figure A.43: EDC of percussive signals.

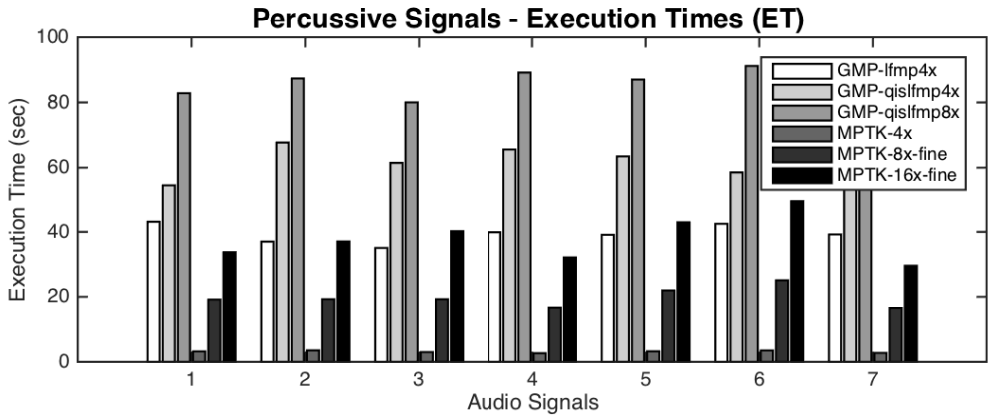


Figure A.44: ET of percussive signals.

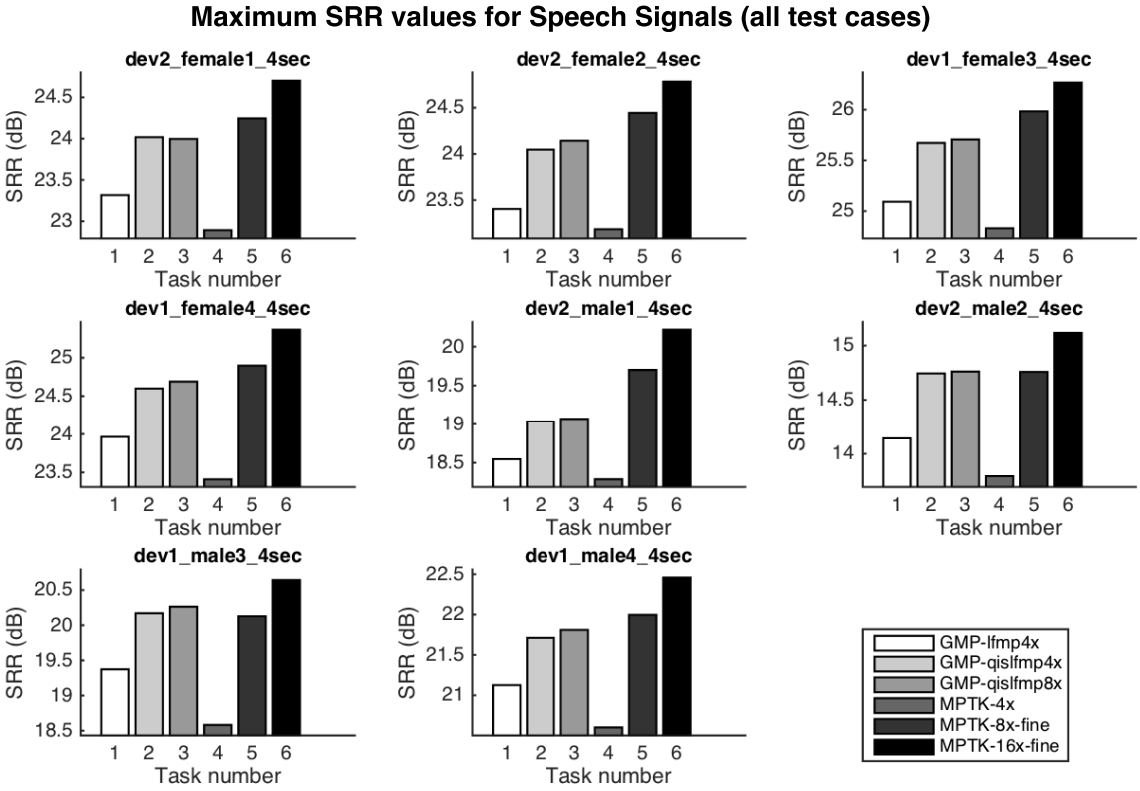


Figure A.45: SRR values of speech signals.

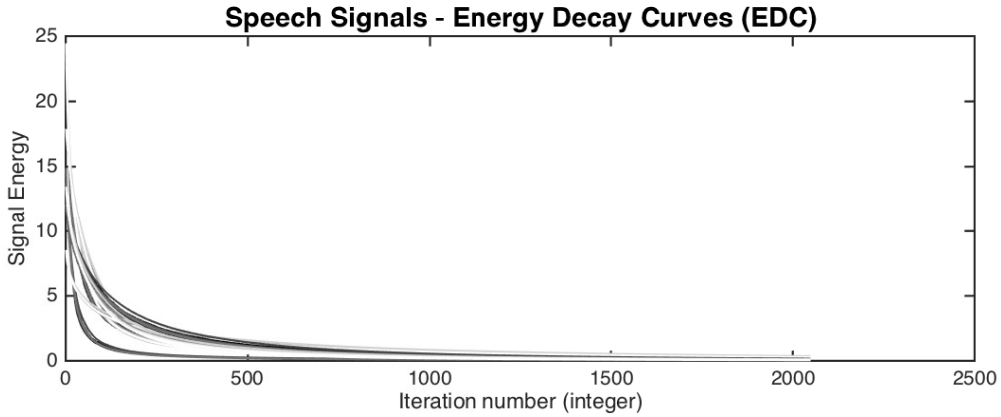


Figure A.46: EDC of speech signals.

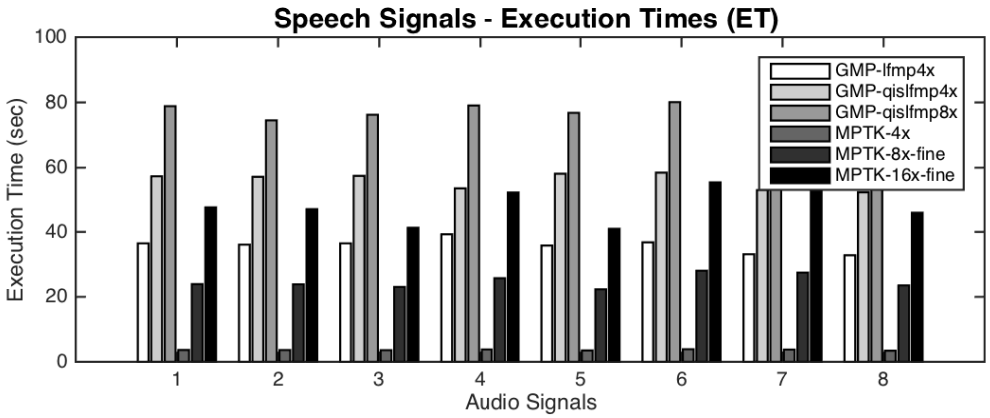


Figure A.47: ET of speech signals.

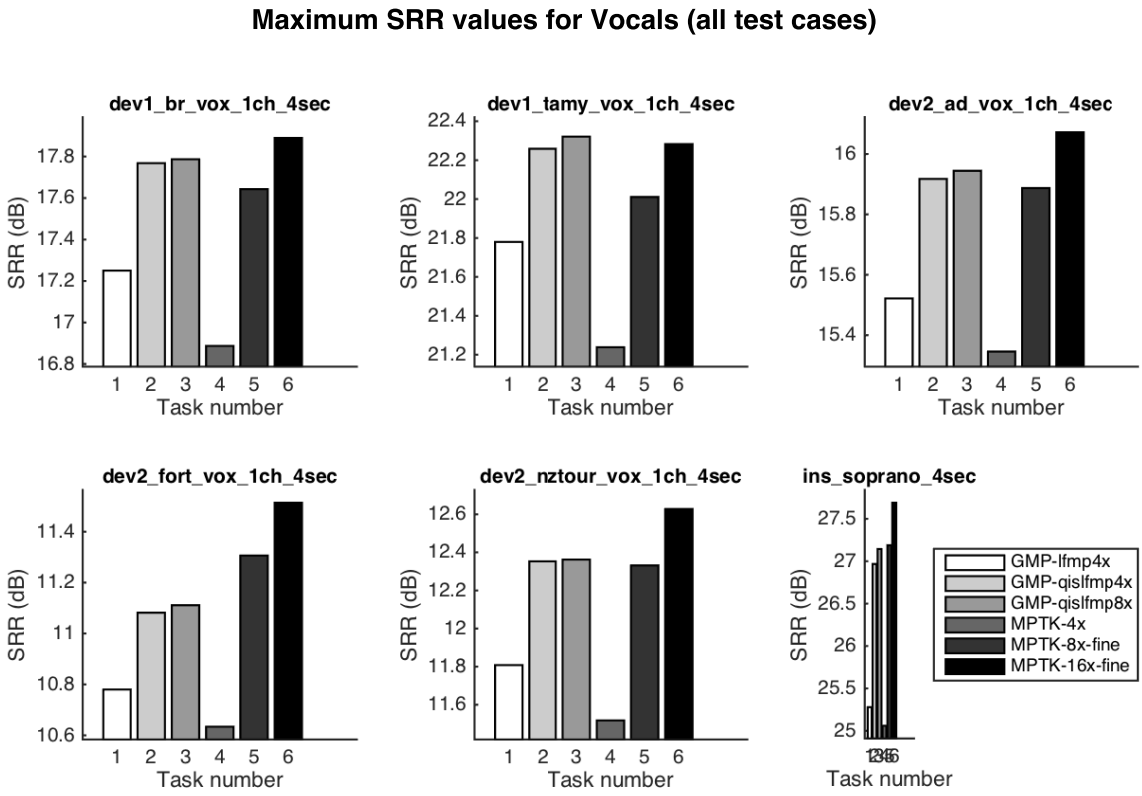


Figure A.48: SRR values of vocal signals.

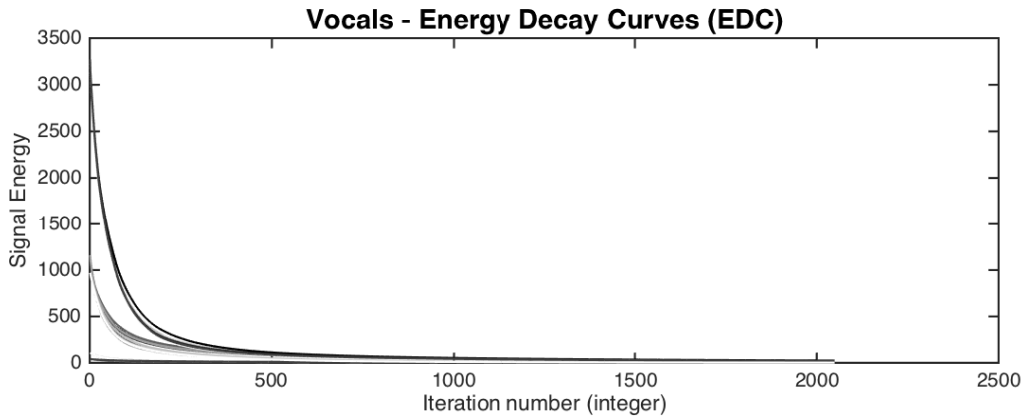


Figure A.49: EDC of vocal signals.

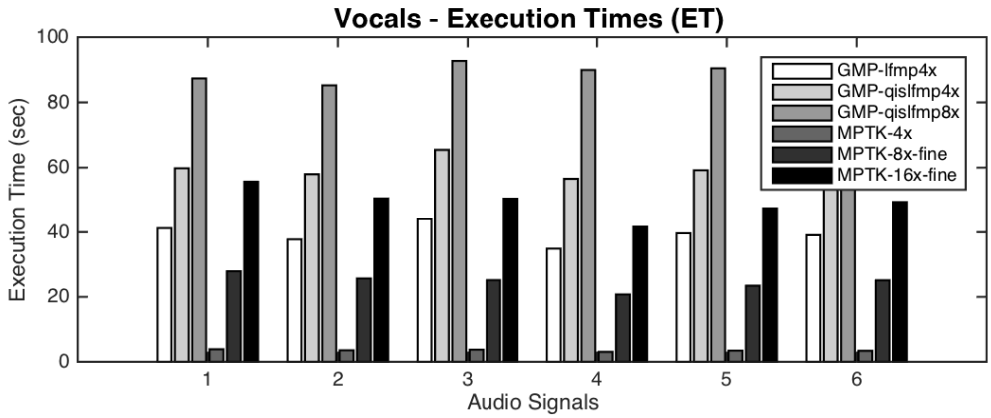


Figure A.50: ET of vocal signals.

Abbreviations

1-D	One-dimensional signals
2-D	Two-dimensional signals
AC	Agglomerative Clustering
ADC	Analogue to Digital Converter / Conversion
AFM	Arbitrary Frequency Modulation
AM	Amplitude Modulation
BMP	Basic Matching Pursuit
BOB	Best Orthogonal Basis
BP	Band Pass filter
BP	Basis Pursuit
CCQ	Coefficient Cumulated Quality
CLSA	Computerised Lung Sound Analysis
COLA	Constant OverLap Add
CPU	Central Processing Unit
CWT	Continuous Wavelet Transform
DFT	Discrete Fourier Transform
DMP	Dual Matching Pursuit
DP	De-mixing Pursuit

DSPs Digital Signal Processors

DT-CWPD Dual Tree Complex Wavelet Packet Decomposition

DT-CWT Dual Tree Complex Wavelet Transform

DWT Discrete Wavelet Transform

EDC Energy Decay Curve

EEG Electroencephalography

ERB Equivalent Rectangular Bandwidth

FFT Fast Fourier Transform

FIR Finute Impulse Response

FM Frequency Modulation

FPGA Field Programmable Gate Array

FPGA Field Programmable Gate Array

FT Fourier Transform

GMP Guided Matching Pursuit

GMPC Guided Matching Pursuit Coder

HMP Harmonic Matching Pursuit

HP High Pass filter

HRP High Resolution Pursuit

Hz hertz : unit of frequency in the Internation System of Units (SI)

ICWT Inverse Continuous Wavelet Transform

IFFT Inverse Fast Fourier Transform

IIR Inifinte Impulse Response

IPA International Phonetic Alphabet

ISTFT Invesre Short Tome Fourier Transform

LFMP Long Frame Matching Pursuit

LP Linear Programming

LP Low Pass filter

LTI Linear Time Invariant

MAC Multiplier Accumulator unit

MDCT Modified Discrete Cosine Transform

MIR Music Information Retrieval

MMP Molecular Matching Pursuit

MOF Method of Frames

MP Matching Pursuit

MP-CAD Matching Pursuit with Content Adaptive Dictionaries

MP-SSD Mathcing Pursuit with Source Specific Dictionaries

NP Notch Pass filter

OMP Orthogonal Matching Pursuit

OO Object Oriented

OOP Object Oriented Programming

PC Personal Computer

PR Perfect Reconstruction conditions

QIFFT Quadratically Interpolated Fast Fourier Transform

sec Seconds

SMP Stereo Matching Pursuit

smp Samples

SRR Signal to Residual Ratio

STFT Short-Time Fourier Transform

TMP Temporal Matching Pursuit

VLMP Variable Length Matching Pursuit

WMP Weak Matching Pursuit

WPD Wavelet Packet Decomposition

Bibliography

[1] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[2] E. C. Cherry, "Some experiments on the recognition of speech, with one and with two ears," *The Journal of the acoustical society of America*, vol. 25, no. 5, pp. 975–979, 1953.

[3] W. Wang, *Machine Audition: Principles, Algorithms and Systems: Principles, Algorithms and Systems*. IGI Global, 2010.

[4] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[5] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[6] R. B. Dannenberg, W. P. Birmingham, B. Pardo, N. Hu, C. Meek, and G. Tzanetakis, "A comparative evaluation of search techniques for query-by-humming using the musart testbed," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 5, pp. 687–701, 2007.

[7] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger, "Incarmusic: Context-aware music recommendations in a car." in *EC-Web*, vol. 11, 2011, pp. 89–100.

[8] X. Wang, D. Rosenblum, and Y. Wang, "Context-aware mobile music recommendation for daily activities," in *Proceedings of the 20th ACM international conference on Multimedia*, 2012, pp. 99–108.

[9] Y. C. Zhang, D. Ó. Séaghdha, D. Quercia, and T. Jambor, "Auralist: introducing serendipity into music recommendation," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 13–22.

[10] A. Gurung, C. G. Scrafford, J. M. Tielsch, O. S. Levine, and W. Checkley, "Computerized lung sound analysis as diagnostic aid for the detection of abnormal lung sounds: a systematic review and meta-analysis," *Respiratory medicine*, vol. 105, no. 9, pp. 1396–1403, 2011.

[11] S. Reichert, R. Gass, C. Brandt, and E. Andrès, "Analysis of respiratory sounds: state of the art," *Clinical Medicine Insights. Circulatory, Respiratory and Pulmonary Medicine*, vol. 2, p. 45, 2008.

[12] E. Vincent, G. M. Jafari, A. S. Abdallah, D. M. Plumbley, and E. M. Davies, "Blind audio source separation, technical report c4dm-tr-05-01," Queen Mary, University of London, Tech. Rep., 2005.

[13] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. John Wiley & Sons, 2004.

[14] A. D. Blumlein, "British patent specification 394,325 (improvements in and relating to sound-transmission, sound-recording and sound-reproducing systems)," *J. Audio Eng. Soc*, vol. 6, no. 2, pp. 91–98, 130, 1958.

[15] H. A. M. Clark, G. F. Dutton, and P. B. Vanderlyn, "The 'stereosonic' recording and reproducing system: A two-channel systems for domestic tape records," *J. Audio Eng. Soc*, vol. 6, no. 2, pp. 102–117, 1958.

[16] F. Rumsey, *Spatial Audio*, ser. Music Technology Series. CRC Press, 2012.

[17] A. S. Master, "Stereo music source separation via bayesian modeling," Ph.D. dissertation, Stanford University, 2006.

[18] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[19] M. Puigt, E. Vincent, and Y. Deville, "Validity of the independence assumption for the separation of instantaneous and convolutive mixtures of speech and music sources," in *Independent Component Analysis and Signal Separation*, 2009, pp. 613–620.

[20] F. Abrard and Y. Deville, "Blind separation of dependent sources using the" time-frequency ratio of mixtures" approach," in *Signal Processing and Its Applications, 2003. Proceedings. Seventh International Symposium on*, vol. 2, 2003, pp. 81–84.

[21] F. Abrard and Y. Deville, "A time–frequency blind signal separation method applicable to underdetermined mixtures of dependent sources," *Signal Processing*, vol. 85, no. 7, pp. 1389–1403, 2005.

[22] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.

[23] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural computation*, vol. 9, no. 7, pp. 1483–1492, 1997.

[24] A. Hyvärinen, "A family of fixed-point algorithms for independent component analysis," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 5, 1997, pp. 3917–3920.

[25] Z. Koldovsky, P. Tichavsky, and E. Oja, "Efficient variant of algorithm FastICA for independent component analysis attaining the cramer-rao lower bound," *Neural Networks, IEEE Transactions on*, vol. 17, no. 5, pp. 1265–1277, 2006.

[26] H. Li and T. Adali, "A class of complex ica algorithms based on the kurtosis cost function," *Neural Networks, IEEE Transactions on*, vol. 19, no. 3, pp. 408–420, 2008.

[27] V. Zarzoso and P. Comon, "Robust independent component analysis by iterative maximization of the kurtosis contrast with algebraic optimal step size," *Neural Networks, IEEE Transactions on*, vol. 21, no. 2, pp. 248–261, 2010.

[28] M. Girolami and C. Fyfe, "Negentropy and kurtosis as projection pursuit indices provide generalised ica algorithms," *Advances in Neural Information Processing Systems*, vol. 9, 1996.

[29] M. Girolami and C. Fyfe, "Generalised independent component analysis through unsupervised learning with emergent bussgang properties," in *Neural Networks, 1997., International Conference on*, vol. 3, 1997, pp. 1788–1791.

[30] M. Novey and T. Adali, "Complex ica by negentropy maximization," *Neural Networks, IEEE Transactions on*, vol. 19, no. 4, pp. 596–609, 2008.

[31] A. Hyvarinen, "New approximations of differential entropy for independent component analysis and projection pursuit," *Advances in neural information processing systems*, vol. 10, no. 2, pp. 273–279, 1998.

[32] H. Yang and S.-i. Amari, "Adaptive online learning algorithms for blind separation: maximum entropy and minimum mutual information," *Neural computation*, vol. 9, no. 7, pp. 1457–1482, 1997.

[33] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[34] D. Pahm, P. Garrat, and C. Jutten, "Separation of a mixture of independent sources through a ml approach," in *Proc. European Signal Processing Conf*, 1992, p. 771.

[35] D. T. Pham and P. Garat, "Blind separation of mixture of independent sources through a maximum likelihood approach," in *In Proc. EUSIPCO*, 1997.

[36] B. Pearlmutter and L. Parra, "Maximum likelihood blind source separation: A context-sensitive generation of ica," *Advances in Neural Information Processing Systems, vl*, vol. 9, pp. 613—-619, 1997.

[37] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[38] S.-i. Amari, A. Cichocki, H. H. Yang *et al.*, "A new learning algorithm for blind signal separation," *Advances in neural information processing systems*, pp. 757–763, 1996.

[39] J.-F. Cardoso, "Infomax and maximum likelihood for blind source separation," *Signal Processing Letters, IEEE*, vol. 4, no. 4, pp. 112 –114, 1997.

[40] J.-F. Cardoso, "High-order contrasts for independent component analysis," *Neural computation*, vol. 11, no. 1, pp. 157–192, 1999.

[41] A. Hyvärinen, "Fast and robust fixed-point algorithms for independent component analysis," *Neural Networks, IEEE Transactions on*, vol. 10, no. 3, pp. 626–634, 1999.

[42] K. Torkkola, "Blind separation of convolved sources based on information maximization," in *Neural Networks for Signal Processing [1996] VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, 1996, pp. 423–432.

[43] S.-i. Amari, S. C. Douglas, A. Cichocki, and H. H. Yang, "Multichannel blind deconvolution and equalization using the natural gradient," in *Signal Processing Advances in Wireless Communications, First IEEE Signal Processing Workshop on*, 1997, pp. 101–104.

[44] S. C. Douglas and X. Sun, "Convolutive blind separation of speech mixtures using the natural gradient," *Speech Communication*, vol. 39, no. 1, pp. 65–78, 2003.

[45] P. Smaragdis, "Blind separation of convolved mixtures in the frequency domain," *Neurocomputing*, vol. 22, no. 1, pp. 21–34, 1998.

[46] L. Parra and C. Spence, "Convolutive blind separation of non-stationary sources," vol. 8, no. 3, pp. 320–327, 2000.

[47] R. Mukai, H. Sawada, S. Araki, and S. Makino, "Frequency domain blind source separation for many speech signals," in *Independent Component Analysis and Blind Signal Separation*, 2004, pp. 461–469.

[48] H. Sawada, R. Mukai, S. Araki, and S. Makino, "Frequency-domain blind source separation," in *Speech enhancement*, 2005, pp. 299–327.

[49] S. Makino, H. Sawada, R. Mukai, and S. Araki, "Blind source separation of convolutive mixtures of speech in frequency domain," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 88, no. 7, pp. 1640–1655, 2005.

[50] M. Z. Ikram and D. R. Morgan, "A beamforming approach to permutation alignment for multichannel frequency-domain blind speech separation," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 1, 2002, pp. I–881.

[51] H. Sawada, R. Mukai, S. Araki, and S. Makino, "A robust and precise method for solving the permutation problem of frequency-domain blind source separation," *Speech and Audio Processing, IEEE Transactions on*, vol. 12, no. 5, pp. 530–538, 2004.

[52] T.-W. Lee, M. S. Lewicki, M. Girolami, and T. J. Sejnowski, "Blind source separation of more sources than mixtures using overcomplete representations," *Signal Processing Letters, IEEE*, vol. 6, no. 4, pp. 87–90, 1999.

[53] P. Bofill and M. Zibulevsky, "Underdetermined blind source separation using sparse representations," *Signal processing*, vol. 81, no. 11, pp. 2353–2362, 2001.

[54] M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural computation*, vol. 13, no. 4, pp. 863–882, 2001.

[55] P. Georgiev, F. Theis, and A. Cichocki, "Sparse component analysis and blind source separation of underdetermined mixtures," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 992–996, 2005.

[56] Y. Li, S.-I. Amari, A. Cichocki, D. W. Ho, and S. Xie, "Underdetermined blind source separation based on sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 2, pp. 423–437, 2006.

[57] A. Jourjine, S. Rickard, and O. Yilmaz, "Blind separation of disjoint orthogonal signals: Demixing n sources from 2 mixtures," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 5, 2000, pp. 2985–2988.

[58] O. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *Signal Processing, IEEE transactions on*, vol. 52, no. 7, pp. 1830–1847, 2004.

[59] M. M. Van Hulle, "Clustering approach to square and non-square blind source separation," in *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, 1999, pp. 315–323.

[60] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[61] J. B. Allen and L. R. Rabiner, "A unified approach to short-time fourier analysis and synthesis," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[62] J. P. Princen and A. B. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 5, pp. 1153–1161, 1986.

[63] S. Mallat, *A wavelet tour of signal processing: the sparse way*. Academic press, 2009.

[64] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[65] I. F. Gorodnitsky, J. S. George, and B. D. Rao, "Neuromagnetic source imaging with focuss: a recursive weighted minimum norm algorithm," *Electroencephalography and clinical Neurophysiology*, vol. 95, no. 4, pp. 231–251, 1995.

[66] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using focuss: A re-weighted minimum norm algorithm," *Signal Processing, IEEE Transactions on*, vol. 45, no. 3, pp. 600–616, 1997.

[67] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on pure and applied mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[68] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3397–3415, 1993.

[69] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, 1993, pp. 40–44.

[70] M. A. Figueiredo, "Adaptive sparseness for supervised learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 9, pp. 1150–1159, 2003.

[71] D. P. Wipf and B. D. Rao, "Sparse bayesian learning for basis selection," *Signal Processing, IEEE Transactions on*, vol. 52, no. 8, pp. 2153–2164, 2004.

[72] M. D. Plumbley, S. A. Abdallah, T. Blumensath, and M. E. Davies, "Sparse representations of polyphonic music," *Signal Processing*, vol. 86, no. 3, pp. 417–431, 2006.

[73] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural computation*, vol. 12, no. 2, pp. 337–365, 2000.

[74] F. J. Theis, A. Jung, C. G. Puntonet, and E. W. Lang, "Linear geometric ica: Fundamentals and algorithms," *Neural computation*, vol. 15, no. 2, pp. 419–439, 2003.

[75] M. Puigt and Y. Deville, "Time–frequency ratio-based blind separation methods for attenuated and time-delayed sources," *Mechanical Systems and Signal Processing*, vol. 19, no. 6, pp. 1348–1379, 2005.

[76] S. Arberet, R. Gribonval, and F. Bimbot, "A robust method to count and locate audio sources in a stereophonic linear instantaneous mixture," *Independent Component Analysis and Blind Signal Separation*, pp. 536–543, 2006.

[77] T. Melia, S. Rickard, and C. Fearon, "Extending the duet blind source separation technique," in *Proc. First Workshop on Signal Processing with Sparse/Structured Representations (SPARS'05)*, 2005, pp. 67–70.

[78] H. Sawada, S. Araki, R. Mukai, and S. Makino, "Blind extraction of dominant target sources using ica and time-frequency masking," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 6, pp. 2165–2173, 2006.

[79] H. Sawada, S. Araki, and S. Makino, "A two-stage frequency-domain blind source separation method for underdetermined convolutive mixtures," in *Applications of Signal Processing to Audio and Acoustics, 2007 IEEE Workshop on*, 2007, pp. 139–142.

[80] S. Araki, H. Sawada, R. Mukai, and S. Makino, "Underdetermined blind sparse source separation for arbitrarily arranged multiple sensors," *Signal Processing*, vol. 87, no. 8, pp. 1833–1847, 2007.

[81] D. Wang and G. J. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications*. Wiley-IEEE Press, 2006.

[82] D. P. Ellis, "Prediction-driven computational auditory scene analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 1996.

[83] G. J. Brown, "Computational auditory scene analysis: A representational approach," Ph.D. dissertation, University of Sheffield, 1992.

[84] G. Siamantas, "Towards an automatic musical source separation system for single-channel recordings using music-inspired assumptions," Ph.D. dissertation, University of York, 2006.

[85] M. Every, "Separation of musical sources and structures from single-channel polyphonic recordings," Ph.D. dissertation, University of York, 2006.

[86] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals and Systems*, ser. Prentice-Hall signal processing series. Prentice Hall, 1997.

[87] S. Soliman and M. Srinath, *Continuous and discrete signals and systems*, ser. Prentice-Hall information and system sciences series. Prentice Hall, 1998.

[88] B. Lathi, *Signal Processing and Linear Systems*. Berkeley Cambridge Press, 1998.

[89] A. V. Oppenheim, R. W. Schafer, J. R. Buck *et al.*, *Discrete-Time Signal Processing: International Version*. Prentice hall Englewood Cliffs, NJ, 2009.

[90] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2009.

[91] S. Kay, *Fundamentals of Statistical Signal Processing: Estimation theory*, ser. Fundamentals of Statistical Signal Processing. Prentice-Hall PTR, 1993.

[92] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[93] MathWorks. MATLAB Wavelet Toolbox. Last Accessed: 2015-08-13. [Online]. Available: http://www.mathworks.co.uk/help/wavelet/index.html

[94] G. Strang and T. Nguyen, *Wavelets and filter banks*. SIAM, 1996.

[95] D. Gabor, "Theory of communication. part 1: The analysis of information," *Electrical Engineers-Part III: Radio and Communication Engineering, Journal of the Institution of*, vol. 93, no. 26, pp. 429–441, 1946.

[96] A. Papoulis, *The Fourier integral and its applications*, ser. McGraw-Hill electronic sciences series. McGraw-Hill, 1962.

[97] J. L. Flanagan and R. Golden, "Phase vocoder," *Bell System Technical Journal*, vol. 45, no. 9, pp. 1493–1509, 1966.

[98] R. W. Schafer and L. R. Rabiner, "Design and simulation of a speech analysis-synthesis system based on short-time fourier analysis," *Audio and Electroacoustics, IEEE Transactions on*, vol. 21, no. 3, pp. 165–174, 1973.

[99] M. R. Portnoff, "Implementation of the digital phase vocoder using the fast fourier transform," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 24, no. 3, pp. 243–248, 1976.

[100] J. Allen, "Short term spectral analysis, synthesis, and modification by discrete Fourier transform," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 25, no. 3, pp. 235–238, 1977.

[101] M. R. Portnoff, "Time-frequency representation of digital signals and systems based on short-time Fourier analysis," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 1, pp. 55–69, 1980.

[102] R. E. Crochiere, "A weighted overlap-add method of short-time Fourier analysis/Synthesis," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 1, pp. 99–102, 1980.

[103] U. Zölzer, X. Amatriain, and D. Arfib, *DAFX: Digital Audio Effects*. Wiley, 2011.

[104] Zplane. élastique Time Stretching & Pitch Shifting SDKs. Last Accessed: 2015-08-13. [Online]. Available: http://licensing.zplane.de/index.php?page=description-elastique

[105] Zynaptiq. ZTX - Precision Time Stretching & Pitch Shifting Technology. Last Accessed: 2015-08-13. [Online]. Available: http://www.zynaptiq.com/ztx/

[106] Prosoniq. MPEX - Minimum Perceived Loss Time Compression/Expansion. Last Accessed: 2015-08-13. [Online]. Available: http://mpex.prosoniq.com/

[107] J. O. Smith and X. Serra, "PARSHL An Analysis/Synthesis Program for non- Harmonic Sounds Based on a Sinusoidal Representation," in *International Computer Music Conference*, 1987, pp. 290–297.

[108] X. Serra, "A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition," Ph.D. dissertation, Stanford University, 1989.

[109] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 4, pp. 744–754, 1986.

[110] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[111] X. Serra, "Musical Sound Modeling with Sinusoids plus Noise," in *Musical Signal Processing*, ser. Studies on New Music Research, C. Roads, S. Pope, A. Picialli, and G. De Poli, Eds. Swets & Zeitlinger, 1997, pp. 91–122.

[112] T. S. Verma, "A perceptually based audio signal model with application to scalable audio compression," Ph.D. dissertation, Stanford University, 2000.

[113] X. Serra. Spectral Modeling Relevant References. Last Accessed: 2015-08-13. [Online]. Available: http://mtg.upf.edu/technologies/sms?p=Relevant%20references

[114] A. Graps, "An introduction to wavelets," *Computational Science & Engineering, IEEE*, vol. 2, no. 2, pp. 50–61, 1995.

[115] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE signal processing magazine*, vol. 8, no. 4, pp. 14–38, 1991.

[116] P. S. Addison, *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press, 2002.

[117] M. S. Lewicki, "Efficient coding of natural sounds," *Nature neuroscience*, vol. 5, no. 4, pp. 356–363, 2002.

[118] L. Daudet and B. Torrésani, "Hybrid representations for audiophonic signal encoding," *Signal Processing*, vol. 82, no. 11, pp. 1595–1617, 2002.

[119] Ç. Ö. Etemoğlu and V. Cuperman, "Matching pursuits sinusoidal speech coding," *Speech and Audio Processing, IEEE Transactions on*, vol. 11, no. 5, pp. 413–424, 2003.

[120] P. Vera-Candeas, N. Ruiz-Reyes, M. Rosa-Zurera, D. Martinez-Munoz, and F. Lopez-Ferreras, "Transient modeling by matching pursuits with a wavelet dictionary for parametric audio coding," *Signal Processing Letters, IEEE*, vol. 11, no. 3, pp. 349–352, 2004.

[121] M. G. Christensen and S. H. Jensen, "The cyclic matching pursuit and its application to audio modeling and coding," in *Asilomar Conference Signals, Systems, and Computers*, 2007, pp. 550–554.

[122] E. Ravelli, G. Richard, and L. Daudet, "Union of mdct bases for audio coding," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 1361–1372, 2008.

[123] T. S. Verma and T. H. Meng, "Sinusoidal modeling using frame-based perceptually weighted matching pursuits," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 2, 1999, pp. 981–984.

[124] C. Févotte, L. Daudet, S. J. Godsill, and B. Torrésani, "Sparse regression with structured priors: Application to audio denoising," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 3, 2006, pp. III–III.

[125] C. Févotte, B. Torrésani, L. Daudet, and S. J. Godsill, "Sparse linear regression with structured priors and application to denoising of musical audio," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 174–185, 2008.

[126] R. Gribonval, E. Bacry, S. Mallat, P. Depalle, and X. Rodet, "Analysis of sound signals with high resolution matching pursuit," in *Time-Frequency and Time-Scale Analysis, 1996., Proceedings of the IEEE-SP International Symposium on*, 1996, pp. 125–128.

[127] S. A. Abdallah and M. D. Plumbley, "Unsupervised analysis of polyphonic music by sparse coding," *Neural Networks, IEEE Transactions on*, vol. 17, no. 1, pp. 179–196, 2006.

[128] T. Blumensath and M. Davies, "Sparse and shift-invariant representations of music," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 1, pp. 50–57, 2006.

[129] P. Leveau, E. Vincent, G. Richard, and L. Daudet, "Instrument-specific harmonic atoms for mid-level music representation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 116–128, 2008.

[130] G. Kling and C. Roads, "Audio analysis, visualization, and transformation with the matching pursuit algorithm," in *Proceedings of the 7th International Conference on Digital Audio Effects (DAFx-04)*, 2004, pp. 33–37.

[131] B. L. Sturm, C. Roads, A. McLeran, and J. J. Shynk, "Analysis, visualization, and transformation of audio signals using dictionary-based methods†," *Journal of New Music Research*, vol. 38, no. 4, pp. 325–341, 2009.

[132] A. P. Lobo and P. C. Loizou, "Voiced/unvoiced speech discrimination in noise using gabor atomic decomposition," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 1, 2003, pp. I–820.

[133] K. Huang and S. Aviyente, "Sparse representation for signal classification," in *Advances in neural information processing systems*, 2006, pp. 609–616.

[134] S. Chu, S. Narayanan, and C.-C. J. Kuo, "Environmental sound recognition using mp-based features," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, 2008, pp. 1–4.

[135] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[136] C. Kereliuk and P. Depalle, "Sparse atomic modeling of audio: A review," in *Proc. of the 14th Int. Conference on Digital Audio Effects (DAFx-11), Paris, France*, 2011.

[137] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constructive approximation*, vol. 13, no. 1, pp. 57–98, 1997.

[138] R. R. Coifman and M. V. Wickerhauser, "Entropy-based algorithms for best basis selection," *Information Theory, IEEE Transactions on*, vol. 38, no. 2, pp. 713–718, 1992.

[139] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *Information Theory, IEEE Transactions on*, vol. 47, no. 7, pp. 2845–2862, 2001.

[140] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[141] P. J. Green, "Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 149–192, 1984.

[142] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk, "Iteratively reweighted least squares minimization for sparse recovery," *Communications on Pure and Applied Mathematics*, vol. 63, no. 1, pp. 1–38, 2010.

[143] S. F. Cotter, B. D. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *Signal Processing, IEEE Transactions on*, vol. 53, no. 7, pp. 2477–2488, 2005.

[144] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[145] J. M. Bioucas-Dias and M. A. Figueiredo, "A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration," *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2992–3004, 2007.

[146] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[147] M. Fornasier and H. Rauhut, "Recovery algorithms for vector-valued data with joint sparsity constraints," *SIAM Journal on Numerical Analysis*, vol. 46, no. 2, pp. 577–613, 2008.

[148] S. Jaggi, W. C. Karl, S. Mallat, and A. S. Willsky, "High resolution pursuit for feature extraction," *Applied and Computational Harmonic Analysis*, vol. 5, no. 4, pp. 428–449, 1998.

[149] J. H. Friedman and W. Stuetzle, "Projection pursuit regression," *Journal of the American statistical Association*, vol. 76, no. 376, pp. 817–823, 1981.

[150] S. Qian and D. Chen, "Signal representation using adaptive normalized gaussian functions," *Signal processing*, vol. 36, no. 1, pp. 1–11, 1994.

[151] G. M. Davis, S. G. Mallat, and Z. Zhang, "Adaptive time-frequency decompositions with matching pursuit," in *SPIE's International Symposium on Optical Engineering and Photonics in Aerospace Sensing*, 1994, pp. 402–413.

[152] MathWorks. MATLAB Matching Pursuit Algorithm Implementation wmpalg. Last Accessed: 2015-08-13. [Online]. Available: http://uk.mathworks.com/help/wavelet/ref/wmpalg.html

[153] P. J. Durka, D.Ircha, and K. J. Blinowska, "Stochastic time-frequency dictionaries for matching pursuit," *Signal Processing, IEEE Transactions on*, vol. 49, no. 3, pp. 507–510, 2001.

[154] S. Krstulović and R. Gribonval, "MPTK: Matching pursuit made tractable," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, 2006.

[155] MathWorks. MATLAB Matching Pursuit Algorithms. Last Accessed: 2015-08-13. [Online]. Available: http://uk.mathworks.com/help/wavelet/ug/matching-pursuit-algorithms.html

[156] M. Goodwin, "Matching pursuit with damped sinusoids," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 3, 1997, pp. 2037–2040.

[157] M. M. Goodwin and M. Vetterli, "Matching pursuit and atomic signal models based on recursive filter banks," *Signal Processing, IEEE Transactions on*, vol. 47, no. 7, pp. 1890–1902, 1999.

[158] R. Gribonval, "Fast matching pursuit with a multiscale dictionary of gaussian chirps," *Signal Processing, IEEE Transactions on*, vol. 49, no. 5, pp. 994–1001, 2001.

[159] R. Gribonval and E. Bacry, "Harmonic decomposition of audio signals with matching pursuit," *Signal Processing, IEEE Transactions on*, vol. 51, no. 1, pp. 101–111, 2003.

[160] L. Daudet, "Sparse and structured decompositions of signals with the molecular matching pursuit," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 5, pp. 1808–1816, 2006.

[161] B. L. Sturm, J. J. Shynk, and S. Gauglitz, "Agglomerative clustering in sparse atomic decompositions of audio signals." in *ICASSP*, 2008.

[162] R. Gribonval, "Sparse decomposition of stereo signals with matching pursuit and application to blind separation of more than two sources from a stereo mixture," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, 2002.

[163] S. Lesage, S. Krstulović, and R. Gribonval, "Under-determined source separation: comparison of two approaches based on sparse decompositions," in *Independent Component Analysis and Blind Signal Separation*, 2006.

[164] P. Bofill and M. Zibulevsky, "Blind separation of more sources than mixtures using sparsity of their short-time fourier transform," in *Proc. ica*, vol. 2000, 2000, pp. 87–92.

[165] R. Gribonval and M. Nielsen, "Beyond sparsity: Recovering structured representations by ℓ^1 minimization and greedy algorithms," *Advances in computational mathematics*, vol. 28, no. 1, pp. 23–41, 2008.

[166] P. Sugden and N. Canagarajah, "Underdetermined noisy blind separation using dual matching pursuits," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, 2004.

[167] P. Sugden and N. Canagarajah, "Underdetermined blind separation using learned basis function sets," *Electronics Letters*, vol. 39, no. 1, pp. 158–160, 2003.

[168] N. Cho, Y. Shiu, and C. J. Kuo, "Audio source separation with matching pursuit and content-adaptive dictionaries (mp-cad)," in *Applications of Signal Processing to Audio and Acoustics, 2007 IEEE Workshop on*, 2007.

[169] N. Cho and C. J. Kuo, "Sparse music representation with source-specific dictionaries and its application to signal separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 2, pp. 326–337, 2011.

[170] N. Cho and C.-C. J. Kuo, "Sparse representation of musical signals using source-specific dictionaries," *Signal Processing Letters, IEEE*, vol. 17, no. 11, pp. 913–916, 2010.

[171] M. Aharon, M. Elad, and A. Bruckstein, "K -SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[172] D. Zantalis and J. Wells, "Semi-blind audio source separation of linearly mixed two-channel recordings via guided matching pursuit," in *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, 2014.

[173] P. Leveau, L. Daudet, S. Krstulovic, and R. Gribonval, "Model-based matching pursuit-estimation of chirp factors and scale of gabor atoms with iterative extension," in *Signal Processing with Adaptive Sparse Structured Representations (SPARS'05)*, 2005.

[174] P. Mineault. Matching pursuit for 1d signals. Last Accessed: 2015-08-13. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/32426-matching-pursuit-for-1d-signals

[175] M. M. Goodwin, "Multiscale overlap-add sinusoidal modeling using matching pursuit and refinements," in *Proc. of Workshop on Application of Signal Processing to Audio and Acoustics (WASPAA)*, 2001.

[176] R. Heusdens, R. Vafin, and W. B. Kleijn, "Sinusoidal modeling of audio and speech using psychoacoustic-adaptive matching pursuits," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 5, 2001, pp. 3281–3284.

[177] R. Heusdens, R. Vafin, and W. B. Kleijn, "Sinusoidal modeling using psychoacoustic-adaptive matching pursuits," *Signal Processing Letters, IEEE*, vol. 9, no. 8, pp. 262–265, 2002.

[178] MathWorks. MATLAB Stream Processing. Last Accessed: 2015-08-13. [Online]. Available: http://uk.mathworks.com/discovery/stream-processing.html

[179] J. O. Smith, *Spectral audio signal processing*. W3K, 2011, last Accessed: 2015-08-13. [Online]. Available: https://ccrma.stanford.edu/~jos/sasp/

[180] J. T. Foote and M. L. Cooper, "Media segmentation using self-similarity decomposition," in *Electronic Imaging 2003*, 2003.

[181] C. Harte, M. Sandler, and M. Gasser, "Detecting harmonic change in musical audio," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, 2006.

[182] O. Lartillot and P. Toiviainen, "A MATLAB toolbox for musical feature extraction from audio," in *International Conference on Digital Audio Effects*, 2007.

[183] M. Abe and J. O. Smith III, "AM/FM rate estimation for time-varying sinusoidal modeling," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, 2005.

[184] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[185] A. H. Nuttall, "Some windows with very good sidelobe behavior," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 29, no. 1, pp. 84–91, 1981.

[186] P. J. Durka. Matching Pursuit. Last Accessed: 2015-08-13. [Online]. Available: http://www.scholarpedia.org/article/Matching_pursuit

[187] P. J. Durka, A. Matysiak, E. M. Montes, P. V. Sosa, and K. J. Blinowska, "Multichannel matching pursuit and eeg inverse solutions," *Journal of neuroscience methods*, vol. 148, no. 1, pp. 49–59, 2005.

[188] R. F. Lyon, A. G. Katsiamis, and E. M. Drakakis, "History and future of auditory filter models," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010.

[189] S. Strahl and A. Mertins, "Sparse gammatone signal model optimized for english speech does not match the human auditory filters," *Brain research*, vol. 1220, pp. 224–233, 2008.

[190] S. Strahl and A. Mertins, "Analysis and design of gammatone signal models," *The Journal of the Acoustical Society of America*, vol. 126, no. 5, pp. 2379–2389, 2009.

[191] B. C. Moore, R. W. Peters, and B. R. Glasberg, "Auditory filter shapes at low center frequencies," *The Journal of the Acoustical Society of America*, vol. 88, no. 1, pp. 132–140, 1990.

[192] MathWorks. MATLAB Chrip Swept Frequency Cosine chirp. Last Accessed: 2015-08-13. [Online]. Available: http://uk.mathworks.com/help/signal/ref/chirp.html

[193] S. Charleston-Villalobos, R. González-Camarena, G. Chi-Lem, and T. Aljama-Corrales, "Crackle sounds analysis by eprclmode decomposition," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 26, no. 1, pp. 40–47, 2007.

[194] D. Donoho, I. Drori, V. Stodden, Y. Tsaig, A. Maleki, and M. Shahram. SparseLab 2.1 MATLAB Toolbox. Last Accessed: 2015-08-13. [Online]. Available: https://sparselab.stanford.edu/

[195] T. Blumensath. Sparsify MATLAB toolbox. Last Accessed: 2015-08-13. [Online]. Available: http://www.personal.soton.ac.uk/tb1m08/sparsify/sparsify.html

[196] MathWorks. Mathworks System Objects. Last Accessed: 2015-08-13. [Online]. Available: http://uk.mathworks.com/help/comm/gs/what-are-system-objects.html

[197] MathWorks. Mathworks Class Constructor Methods. Last Accessed: 2015-08-13. [Online]. Available: http://uk.mathworks.com/help/matlab/matlab_oop/class-constructor-methods.html

[198] F. Abrard, Y. Deville, and P. White, "A new source separation approach for instantaneous mixtures based on time-frequency analysis," *Proc. ECM2S*, pp. 259–267, 2001.

[199] S. Arberet, R. Gribonval, and F. Bimbot, "A robust method to count and locate audio sources in a stereophonic linear anechoic mixture," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 3, 2007, pp. 745–748.

[200] S. Arberet, R. Gribonval, and F. Bimbot, "A robust method to count and locate audio sources in a multichannel underdetermined mixture," *Signal Processing, IEEE Transactions on*, vol. 58, no. 1, pp. 121–133, 2010.

[201] E. Vincent, "Musical source separation using time-frequency source priors," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 1, pp. 91–98, 2006.

[202] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[203] P. Sestini, E. Renzoni, M. Rossi, V. Beltrami, and M. Vagliasindi, "Multimedia presentation of lung sounds as a learning aid for medical students," *European Respiratory Journal*, vol. 8, no. 5, pp. 783–788, 1995.

[204] A. Sovijarvi, F. Dalmasso, J. Vanderschoot, L. Malmberg, G. Righini, and S. Stoneman, "Definition of terms for applications of respiratory sounds," *European Respiratory Review*, vol. 10, no. 77, pp. 597–610, 2000.

[205] M. Bahoura and X. Lu, "Separation of crackles from vesicular sounds using wavelet packet transform," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, 2006.

[206] Y. A. Tolias, L. J. Hadjileontiadis, and S. M. Panas, "A fuzzy rule-based system for real-time separation of crackles from vesicular sounds," in *Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE*, 1997.

[207] L. Vannuccini, M. Rossi, and G. Pasquali, "A new method to detect crackles in respiratory sounds," *Technology and Health Care*, vol. 6, no. 1, pp. 75–79, 1998.

[208] M. Yeginer and Y. Kahya, "Modeling of pulmonary crackles using wavelet networks," in *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, 2006.

[209] L. J. Hadjileontiadis and I. T. Rekanos, "Detection of explosive lung and bowel sounds by means of fractal dimension," *Signal Processing Letters, IEEE*, vol. 10, no. 10, pp. 311–314, 2003.

[210] P. Piirila and A. Sovijarvi, "Crackles: recording, analysis and clinical significance," *European Respiratory Journal*, vol. 8, no. 12, pp. 2139–2148, 1995.

[211] S. S. Kraman, G. R. Wodicka, Y. Oh, and H. Pasterkamp, "Measurement of respiratory acoustic signals: effect of microphone air cavity width, shape, and venting," *CHEST Journal*, vol. 108, no. 4, pp. 1004–1008, 1995.

[212] D. Fitzgerald, "Harmonic/percussive separation using median filtering," *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, no. 1, pp. 10–13, 2010.

[213] C. Fevotte, R. Gribonval, and E. Vincent. BSS EVAL Toolbox User Guide. Last Accessed: 2015-08-13. [Online]. Available: http://www.irisa.fr/metiss/bsseval/

[214] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 1462–1469, 2006.

[215] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, "Subjective and objective quality assessment of audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 7, pp. 2046–2057, 2011.

[216] E. Vincent, "Complex nonconvex l p norm minimization for underdetermined source separation," in *Independent Component Analysis and Signal Separation*, 2007.