Wind Noise Reduction in Single Channel Speech Signals

Kristian Timm Andersen

Kongens Lyngby 2008

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Abstract

In this thesis a number of wind noise reduction techniques have been reviewed, implemented and evaluated. The focus is on reducing wind noise from speech in single channel signals. More specifically a generalized version of a Spectral Subtraction method is implemented along with a Non-Stationary version that can estimate the noise even while speech is present. Also a Non-Negative Matrix Factorization method is implemented. The PESQ measure, different variations of the SNR and Noise Residual measure, and a subjective MUSHRA test is used to evaluate the performance of the methods. The overall conclusion is that the Non-Negative Matrix Factorization algorithm provides the best noise reduction of the investigated methods. This is based on both the perceptual and energy-based evaluation. An advantage of this method is that it does not need a Voice Activity Detector (VAD) and only assumes a-priori information about the wind noise. In fact, the method can be viewed solely as an advanced noise estimator. The downside of the algorithm is that it has a relatively high computational complexity. The Generalized Spectral Subtraction method is shown to improve the speech quality, when used together with the Non-Negative Matric Factorization.

Preface

This thesis was prepared at Informatics and Mathematical Modeling, Technical University of Denmark in partial fulfillment of the requirements for acquiring the M.Sc. degree in engineering. The workload for this project is 35ECTS points and has been carried out over a period of 6 months from September 3rd 2007 to February 29th 2008.

The thesis deals with different speaker-independent models for reducing wind noise from speech in mono signals. The main focus is on extensions of Spectral Subtraction and Non-Negative Matrix Factorization and evaluation of these methods.

I would like to thank my supervisor Associate Professor Jan Larsen for guidance and help during this thesis. I also wish to thank Ph.D. Student Mikkel N. Schmidt for help with providing articles and mathematical discussion concerning the Qualcomm and Non-Negative Matrix Factorization algorithm. Finally I would like to thank my family and Inge Hansen for all their love and support during the writing of this thesis.

Lyngby, February 2008

Kristian Timm Andersen

Contents

Chapter 1

Introduction

Noise reduction algorithms have been used for many decades to suppress undesired components of a signal, but so far little attention has been directed towards specializing these applications to wind noise reduction. The standard approach to deal with this problem, has been to either use a general noise suppression algorithm or to cover up the microphone with a hood to prevent the wind from exciting the membrane of the microphone. This solution, however, is not very eloquent and it is expected that more powerful signal processing techniques can yield better results. Also with the rapid development of small high-technological consumer products like headsets, mobiles, video cameras and hearing aids, it becomes very impractical to implement a cover for the microphone given the size of the units.

The use of a communication device in stormy weather is an every-day experience for people around the world, but removal of the noise is often not so easy because the issues are manifold. A basic characteristic of wind noise is that it is highly non-stationary in time, sometimes even resembling transient noise. This makes it very hard for an algorithm to estimate the noise from a noisy speech signal and recently methods that incorporates premodeled estimates of the noise has become more popular in general noise reduction schemes. These methods often outperform methods that only estimate the noise based on the noisy signal. Another issue is that of speaker- and signal-independence. Different speakers have different pitch and timbre and an algorithm that is optimized for one speaker might not perform adequately for another speaker. If the method has to be signal independent (i.e. not assume anything about the desired signal) the estimation is even harder.

For this project it has been decided to focus on filtering wind noise from a noisy speech signal. To keep the potential applications as general as possible the dissertation focuses on methods that does not modeling individual speakers and only operates on mono signals.

1.1 Noise Reduction Algorithms

Many different noise reduction algorithms exist, but a lot of them are not directly applicable to speaker independent noise reduction in single channel signals. This section contains an overview of the potential methods for this problem along with a discussion of their relevance and advantages. The methods implemented for this dissertation will be reviewed in greater detail in the following chapters.

Classic filtering algorithms like Wiener filtering [44] and Spectral Subtraction [5] have been used for decades for general noise reduction. The Wiener filter is an optimal filter in the least squares sense to remove noise from a signal, given that the signal and noise are independent, stationary processes. It also assumes that the second order statistics of the signal and noise processes are known and works by attenuating frequencies where the noise is expected to be the most dominant. The biggest problem with this method is that it assumes stationary signals, which is obviously not a good approximation for speech and wind noise. The Spectral Subtraction method subtracts an estimate of the noise magnitude spectrum from the noisy speech magnitude spectrum and transforms it back to the time domain using the phase of the original noisy speech signal. Often the noise estimate is obtained during speech pauses, using a Vocal Activity Detector (VAD). As the method is unable to obtain new noise estimates during speech, this method also assumes that the noise is stationary at least for as long as the person is talking. For both methods adaptive versions have been developed that relaxes the assumption of stationarity a bit [3] [41]. The advantages of these methods are that they are robust, easy to implement and that they have been thoroughly studied and generalized through several decades. For this project the stationary Spectral Subtraction algorithm is implemented and it is shown that it can be generalized to the Wiener filter. Also a non-stationary version where the noise can be estimated during speech is implemented.

With more microphones available, correlation between the desired signals in the different microphones can be used to filter out the noise. This has been used in methods like Independent Component Analysis [30] and directivity based applications like beamforming [19] and sometimes even combined [33]. As only one channel is assumed known for this thesis however, these methods are not applicable here.

More recent methods involve modeling the sources in the noisy signal independently and then using these models to find the best estimate of the speech- and noise-signal. The individual signals can then be separated by for instance refiltering the noisy signal or using binary masking [18]. Many different models have been proposed, for instance Hidden Markov Models [36], Gaussian Mixture Models [11], Vector Quantization [8] and Non-negative Sparse Coding [31]. The problem with this approach is that they often model an individual speaker and therefore are not speaker independent.

The general formulation of the wind reduction problem is what makes it hard. The more information that can be put into the method about the sources, the better the resulting separation is expected to be and in the speaker independent single channel case, the only information available is the expected wind noise and speaker independent models of speech.

For this thesis a modified Non-Negative Matrix Factorization algorithm is suggested as a good way to filter wind noise. The algorithm factorizes the magnitude spectrogram of the noisy signal into a dictionary matrix and a code matrix that contains activations of the dictionary in the respective positions of the noisy magnitude spectrogram. In the modified version, wind noise spectrums are trained and put into the dictionary matrix beforehand. First of all this incorporates wind noise information into the method and subsequently leads to an expected better factorization, but it also makes it possible to determine which part of the code- and dictionary matrix belongs to the estimated clean filtered signal. Based on this factorization, the clean signal can be resynthesized.

Evaluations of noise reduction methods are usually only based on Signal-to-Noise (SNR) measures, like in [21]. This measure, however, does not evaluate how a person perceives sound and a better way to compare methods, would be to implement measures that takes that into consideration. For this purpose the PESQ [16] measure is implemented.

All method implementations, figures and data analysis for this thesis has been done in Matlab.

1.2 Overview of Thesis

The thesis is divided up into the following chapters:

Chapter 1 is the current chapter and forms the introduction to the thesis. In this chapter the problem is defined and a discussion of possible solutions to the problem is given.

Chapter 2 contains the theory behind the Generalized Spectral Subtraction algorithm. This is a basic noise reduction algorithm that is implemented for comparison and as a backend addition to the other two methods.

Chapter 3 contains the theory behind the Non-Stationary Spectral Subtraction algorithm, which is an adaptive version of the normal Spectral Subtraction algorithm. This method allows noise to be estimated, while speech is present, by introducing speech and noise codebooks.

Chapter 4 contains the theory behind the Non-Negative Matrix Factorization algorithm.

Chapter 5 reviews objective, subjective and perceptual-objective measures to evaluate the performance of the noise reduction algorithms.

Chapter 6 describes the data that is used to evaluate the noise reduction algorithms. A part of the sound data has been recorded specifically for this project and this chapter describes how it is obtained and processed.

Chapter 7 is the experimental analysis part of the thesis. In this chapter the parameters of the noise reduction algorithms are optimized to filter wind noise from speech.

Chapter 8 contains the results of the thesis, which is based on the analysis and application of the theory in the previous chapters.

Chapter 9 contains the conclusion of the thesis along with future work.

Chapter 2

Generalized Spectral Subtraction

One of the most widely used methods to attenuate noise from a signal is Spectral Subtraction. In its basic form it is a simple method that operates in the frequency domain to obtain a magnitude spectrum estimate of the noise and then use that estimate to filter the noisy signal. Due to its popularity, many different variations of it have been developed, which will also be reviewed in this section. First the basic version of the method will be given, followed by noise estimation considerations and generalizations to the algorithm. Finally other known methods will be compared to the generalized Spectral Subtraction method.

The basic assumptions of the Spectral Subtraction algorithm are that:

- The noise is additive. This means that the noisy signal consists of a sum of the desired- and noise-signal: $y(n) = s(n) + w(n)$, where $y(n)$ is the noisy signal, $s(n)$ is the desired signal and $w(n)$ is the noise signal.
- The human hearing is insensitive to small phase distortions.

The first assumption means that the noisy signal can be filtered, by simply subtracting the noise estimate from the noisy signal. In the frequency domain this equation becomes: $Y(\omega) = S(\omega) + W(\omega)$. As the phase is very hard to estimate however, usually only the magnitude spectrum of the noise can be obtained. This leads to the following filtering equation:

$$\hat{S}(\omega) = (|Y(\omega)| - |\hat{W}(\omega)|) \cdot e^{j \cdot \phi_Y(\omega)} = \left(1 - \frac{|\hat{W}(\omega)|}{|Y(\omega)|}\right) \cdot Y(\omega) = H(\omega) \cdot Y(\omega) \quad (2.1)$$

with

$$H(\omega) = 1 - \frac{|\hat{W}(\omega)|}{|Y(\omega)|}$$

$\phi_Y(\omega)$ is the phase of $Y(\omega)$. In (2.1) the phase of the noise is approximated by the phase of the noisy signal. This is what the second assumption means. Any negative values in the filter $H(\omega)$ is due to estimation errors in $\hat{W}(\omega)$, where the noise is estimated to be larger than the noisy signal and should be set to zero. Finally the speech estimate $\hat{s}(n)$ is obtained by transforming the spectrum $\hat{S}(\omega)$ back to the time domain with an inverse Fourier transform.

To ensure that the signal is stationary, these calculations are performed on

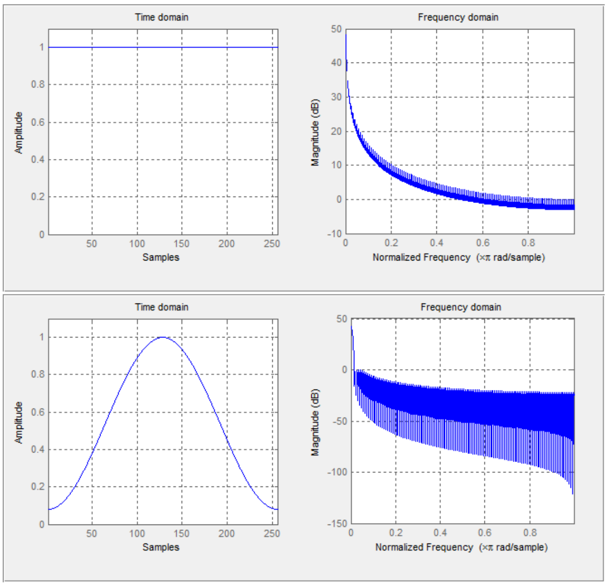


Figure 2.1: *Top: Rectangular window. Bottom: Hamming window.*

small overlapping frames of $y(n)$ (<100ms). Extracting a frame from a signal equates to multiplying it with a rectangular window, but as a rectangular window has very bad spectral properties, the frame is usually instead multiplied with a more smooth window, for instance a Hamming window. This approach causes less spectral smearing of the signal [35], as can be seen in figure 2.1. Then the FFT of the frame is taken and combining all frames into a matrix yields the spectrogram. This procedure is known as Short Time Fourier Transform (STFT). In the following $S(\omega_i, m)$ will mean the spectrogram of a signal $s(n)$, where i is the frequency point and m is the frame number.

There are basically two ways in which a spectral subtraction algorithm can vary: How to estimate the magnitude spectrum of the noise and different generalizations to the filtering operation in (2.1). The two variations will be explained in separate subsections.

2.1 Noise Estimation

A proper noise estimation technique is essential for achieving good results. A popular way to acquire the noise estimate from the noisy signal is during speech pauses [5]:

$$|\hat{W}(\omega_i, m)| = E(|Y(\omega_i, m)|) \qquad \text{, during speech pauses}$$

where $E()$ is the expectation operator. In practice it can implemented as a window of length K over an STFT, where all frames within the window are averaged to obtain the noise estimate:

$$E(|Y(\omega_i, m)|) \approx \frac{1}{K} \sum_{k=l}^{l+K-1} |Y(\omega_i, k)| \qquad \text{, during speech pauses}$$

In order to know when speech is present, a Voice Activity Detector (VAD) is needed and a lot of effort has been devoted towards developing stable VADs, e.g. [39]. This, however, will not be pursued any further in this dissertation and it will be assumed that it is known beforehand when the speech is present.

This way of estimating the noise has a serious drawback. As long as the speech is present, the noise estimate cannot be updated and with very non-stationary signals like wind, the noise estimate will not be a very good estimate over a long time period. Instead of estimating the noise when no speech is present, precomputed codebooks can be used to find the best fit to the current frame. The best fit in the codebook is then the estimated wind noise for that frame. This estimation is also possible while speech is present and will be pursued further in chapter 3.

2.2 Generalizations of Spectral Subtraction

The basic Spectral Subtraction filter given in (2.1) is sufficient if perfect estimations of the noise can be made. This, however, is not possible, as explained in the previous section, which leads to errors in the magnitude of the speech estimate $|\hat{S}(\omega)|$. There can be two kinds of errors in $|\hat{S}(\omega)|$: residual noise and speech distortion. If the noise estimate at a certain frequency is too high, the subtraction algorithm will remove too much from the noisy signal at that frequency point and some of the speech will also be removed. This will be heard as speech distortion. If on the other hand the noise estimate is lower than the actual noise level, there will be some residual noise left in the speech estimate. If this residual noise occupies a large area in the spectrum it will be heard as broadband noise, but if narrow spectral peaks occur in the residual noise, it will be heard as musical tones and is often called 'musical noise' [28]. Musical noise is very annoying to listen to and should be minimized if possible.

2.2.1 Overestimation and Spectral Floor

It can be shown that overestimating the noise like in [28] reduces the musical noise. The idea consists of multiplying a signal dependant constant $\alpha(\omega_i, m)$ onto the noise estimate and putting a lower bound β on the filter. This changes formula (2.1) into:

$$H(\omega_i, m) = \max(1 - \alpha(\omega_i, m) \cdot \frac{|\hat{W}(\omega_i, m)|}{|Y(\omega_i, m)|}, \beta) \qquad , 1 \le \alpha(\omega_i, m), 0 \le \beta \ll 1$$

$\alpha(\omega_i, m)$ is usually calculated based on a relation like this:

$$\alpha(\omega_i, m) = \alpha_0 - \mathit{slope} \cdot 10 \log \left(\frac{|Y(\omega_i, m)|}{|\hat{W}(\omega_i, m)|} \right) \qquad , \alpha_{min} \le \alpha(\omega_i, m) \le \alpha_{max}$$

The formula can be recognized as a linear relationship between the Signal-to-Noise ratio (SNR) and $\alpha(\omega_i, m)$. Other formulas are also possible, for instance a sigmoid function. For the values $\alpha_{min} = 1.25, \alpha_{max} = 3.125, \alpha_0 = 3.125, \mathit{slope} = (\alpha_{max} - \alpha_{min})/20$, the relation between $10 \log \left(\frac{|Y(\omega_i,m)|}{|\hat{W}(\omega_i,m)|} \right)$ and $\alpha(\omega_i, m)$ can be seen in figure 2.2. When the noise is relatively high, the fraction $\frac{|Y(\omega_i,m)|}{|\hat{W}(\omega_i,m)|}$ is low and the filter will subtract more from the noisy signal. This reduces both musical- and broadband-noise. The downside is that if $\alpha(\omega_i, m)$ is too high, the speech will distort. This is an important tradeoff. Another tradeoff is that of choosing β. When there are large peaks in the noise estimate that are further enhanced by $\alpha(\omega_i, m)$, a lot of speech distortion can happen. By introducing

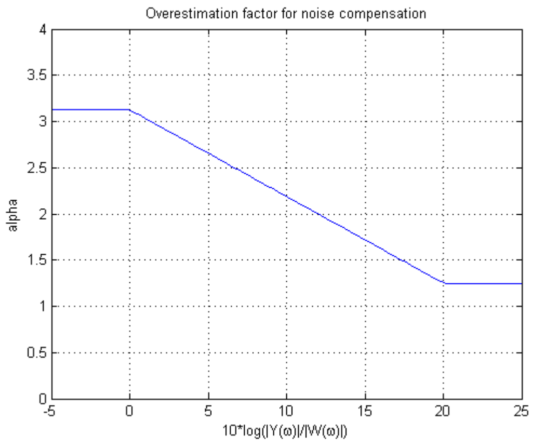


Figure 2.2: *Overestimation as a function of SNR.*

a non-zero spectral floor, these peak excursions can be limited, reducing the speech distortion. This, however, also introduces broadband noise, though it is important to note that the filter $H(\omega_i, m)$ is multiplied onto $Y(\omega_i, m)$ and so this broadband noise will be relative to $Y(\omega_i, m)$ (ie. if β is a lot smaller than 1, it will always be much lower than the speech).

2.2.2 Filter Smoothing

By smoothing the filter $H(\omega_i, m)$ in both the time and frequency domain, large peaks will be reduced, which will reduce the musical noise. This smoothing is inspired by articles like [6] and [27]. In those articles it is suggested that the smoothing over time takes the form of a first order auto-regressive filter to obtain a smoothed filter $H_s(\omega_i, m)$:

$$H_s(\omega_i, m) = \lambda_H \cdot H_s(\omega_i, m-1) + (1-\lambda_H) \cdot H(\omega_i, m) \qquad , 0 \leq \lambda_H \leq 1$$

This filter is also known as a first order lowpass filter. When the smoothed filter has low values, the noisy signal $Y(\omega_i, m)$ is being heavily filtered because there are a large amount of noise. It is therefore expected to be better to smooth a lot in this area and less in areas with a low amount of noise. A new smoothing filter $H_{s2}(\omega_i, m)$ can be calculated as:

$$H_{s2}(\omega_i, m) = H(\omega_i, m) \cdot H_s(\omega_i, m) + H_s(\omega_i, m) \cdot (1 - H_s(\omega_i, m))$$

This is a weighting between $H(\omega_i, m)$ and $H_s(\omega_i, m)$ where $H_s(\omega_i, m)$ is also being used as a weight. In areas with prominent speech $H_s(\omega_i, m)$ will be large and not a lot of smoothing is happening, but when the noise is prominent $(1 - H_s(\omega_i, m))$ is large and the filter will be heavily smoothed.

By smoothing the filter in the frequency domain as well, large peaks in the filter's spectrum will be further reduced and can be done with a simple sliding-window filter of length $2L - 1$:

$$H_{s3}(\omega_i, m) = \frac{1}{2L-1} \sum_{l=-L}^{L} H_{s2}(\omega_{i+l}, m)$$

2.2.3 Exponent Choosing

The Spectral Subtraction method reviewed so far, is also known as Magnitude Spectral Subtraction, because it is the magnitude of the noise spectrum that is being subtracted from the noisy signal. Another kind of Spectral Subtraction is Power Spectral Subtraction, where it is assumed that $|Y(\omega)|^2 = |S(\omega)|^2 + |W(\omega)|^2$. This equation can be attained by squaring the basic assumption for Magnitude Spectral Subtraction:

$$|Y(\omega)|^2 = (|S(\omega)| + |W(\omega)|)^2 = |S(\omega)|^2 + |W(\omega)|^2 + 2|S(\omega)||W(\omega)|$$

By assuming that noise and speech is uncorrelated the last term will vanish on average, ie. approximate $|S(\omega)||W(\omega)|$ with $E[|S(\omega)||W(\omega)|]$.

The basic filter operation will then look like this:

$$|\hat{S}(\omega)|^2 = (|Y(\omega)|^2 - |\hat{W}(\omega)|^2) = \left(1 - \frac{|\hat{W}(\omega)|^2}{|Y(\omega)|^2}\right) \cdot |Y(\omega)|^2 = H_{\text{power}}(\omega) \cdot |Y(\omega)|^2$$

The full complex speech estimate becomes:

$$\hat{S}(\omega) = (H_{\text{power}}(\omega) \cdot |Y(\omega)|^2)^{1/2} \cdot e^{j \cdot \arg(\phi_Y(\omega)} = H_{\text{power}}^{1/2}(\omega) \cdot Y(\omega)$$

To generalize this, an arbitrary constant can be chosen as exponent:

$$\hat{S}(\omega) = \left(1 - \frac{|\hat{W}(\omega)|^\gamma}{|Y(\omega)|^\gamma}\right)^{1/\gamma} \cdot Y(\omega)$$

A suggestion for a different exponent, could be $\gamma = 0.67$, which would equal Steven's Power Law[1] for loudness. To generalize this even further, the exponent to the filter $1/\gamma$ can be set to another constant ρ/γ. Increasing the value of ρ

[1]Steven's Power Law is a proposed relationship between the magnitude of a physical stimulus and its perceived intensity or strength.

yields a stronger filtering, leading to cleaner silence portions at the expense of stronger distortion of the low-energy speech portions:

$$\hat{S}(\omega) = \left(1 - \frac{|\hat{W}(\omega)|^\gamma}{|Y(\omega)|^\gamma}\right)^{\rho/\gamma} \cdot Y(\omega)$$

The use of ρ, can be seen as a kind of overestimation and as in the case with $\alpha(\omega_i, m)$, it might improve the filtering to introduce a lower bound β_{filter}:

$$|\hat{S}(\omega)|^\gamma = \max\left(\left(1 - \frac{|\hat{W}(\omega)|^\gamma}{|Y(\omega)|^\gamma}\right)^{\rho} \cdot |Y(\omega)|^\gamma, \beta_{\text{filter}} \cdot |\hat{W}(\omega)|^\gamma\right) \qquad , 0 \leq \beta_{\text{filter}} \ll 1$$

The lower bound is introduced as a proportion of the noise estimate, to make sure it is not present when there is only speech or silence.

2.2.4 Generalized Filter

Combining all these generalizations into one filtering operation yields:

$$\hat{S}(\omega) = \max\left(\max\left(1 - \alpha(\omega_i, m) \cdot \frac{|\hat{W}(\omega)|^\gamma}{|Y(\omega)|^\gamma}, \beta\right)^{\rho} \cdot |Y(\omega)|^\gamma, \beta_{\text{filter}} \cdot |\hat{W}(\omega)|^\gamma\right)^{1/\gamma} \cdot e^{j \cdot \phi_Y(\omega)}$$

with $H(\omega) = \max\left(1 - \alpha(\omega_i, m) \cdot \frac{|\hat{W}(\omega)|^\gamma}{|Y(\omega)|^\gamma}, \beta\right)^{\rho}$ being smoothed as detailed in section 2.2.2.

Using the values:

- $\alpha(\omega_i, m) = 1$
- $\beta = 0$
- $\gamma = 1$
- $\rho = 1$
- $\beta_{\text{filter}} = 0$
- $\lambda_H = 0$
- $L = 1$

would equal the basic magnitude Spectral Subtraction method.

2.3 References to Known Algorithms

There are other known noise reduction algorithms that are contained within the methods that have been developed in the preceding sections. This section contains a description of those.

2.3.1 Wiener Filter

The Wiener filter is another popular filtering method. In the following it will be shown how the wiener filter relates to the Spectral Subtraction filter given in section 2.2.4.

Generally the Wiener filter can be shown to have the frequency response [25]:

$$H(\omega) = \frac{P_S(\omega)}{P_S(\omega) + P_W(\omega)}$$

with $P(\omega)$ being the power density spectra. The problem with this expression is that the signals are assumed stationary and $P_S(\omega)$ known. Instead the Wiener filter can be approximated with an expected frequency response:

$$H(\omega) = \frac{E[|S(\omega)|^2]}{E[|S(\omega)|^2] + E[|W(\omega)|^2]}$$

Further by assuming $|Y(\omega)|^2 = |S(\omega)|^2 + |W(\omega)|^2$ the expression can be rewritten:

$$H(\omega) = \frac{E[|S(\omega)|^2 + E[|W(\omega)|^2]}{E[|S(\omega)|^2] + E[|W(\omega)|^2]} - \frac{E[|W(\omega)|^2]}{E[|S(\omega)|^2] + E[|W(\omega)|^2]} =$$

$$1 - \frac{E[|W(\omega)|^2]}{E[|Y(\omega)|^2]} = 1 - \frac{|\hat{W}(\omega)|^2}{|Y(\omega)|^2}$$

where $E[|W(\omega)|^2] = |\hat{W}(\omega)|^2$ means that the noise is estimated using some method. It is seen that the Wiener filter has a close relation to the Power Spectral Subtraction method.

2.3.2 Qualcomm

The Qualcomm algorithm is a method that was proposed for the 7th International Conference on Spoken Language Processing in 2002 [1]. It is a complete frontend for an Automatic Speech Recognition (ASR) system and has outperformed the current ETSI Advanced Feature Standard on a number of different testsets. It includes a noise reduction scheme that is contained within the generalized filter given in section 2.2.4. It can be used by setting the different variables to [9]:

- $\alpha(\omega_i, m)$ should be set to the values used in figure 2.2
- $\beta = 0.01$
- $\gamma = 2$
- $\rho = 1$
- $\beta_{\text{filter}} = 0.001$
- $\lambda_H = 0.9$
- $L = 10$

Chapter 3

Non-Stationary Spectral Subtraction

The method given in this section is an advanced non-stationary version of the Spectral Subtraction method described in the previous chapter. A fundamental assumption in the noise estimation for the Spectral Subtraction algorithm in the previous section is that the noise is stationary. This is because the noise estimate cannot be updated while speech is present. Also the method requires a VAD that might not work very well under very noisy conditions.

The fundamental advantage of the non-stationery Spectral Subtraction method is that it estimates the noise and speech in each time frame and thus can adapt to varying levels of noise even while speech is present. This is done by using a-priori information about the spectrum of speech and noise to compute code-books. In [3] and [12] it is argued that it is beneficial to perform the spectral subtraction, as a filtering operation based on the Auto-Regressive (AR) spectral shape of the noise and speech estimate in each frame. This results in smooth frequency spectrums and thus reduces musical noise. The actual filtering operation is basically the same and can be generalized in the same way as for normal Spectral Subtraction.

First the theory behind the method is described followed by a description of the structure and parameters of the model.

3.1 Spectral Subtraction

The basic Magnitude Spectral Subtraction filter of this method is the same as in the previous section, except that the clean noise and speech magnitude spectrum now is estimated in every timeframe and the resulting magnitude spectrum is given as:

$$\hat{S}(\omega) = (|Y(\omega)| - |\hat{W}(\omega)|)e^{j\cdot\phi_Y(\omega)} = \Big(1 - \frac{|\hat{W}(\omega)|}{|Y(\omega)|}\Big)Y(\omega) \approx$$

$$\Big(1 - \frac{|\hat{W}_{AR}(\omega)|}{|\hat{S}_{AR}(\omega)| + |\hat{W}_{AR}(\omega)|}\Big)Y(\omega) = H(\omega)Y(\omega)$$

with

$$H(\omega) = 1 - \frac{|\hat{W}_{AR}(\omega)|}{|\hat{S}_{AR}(\omega)| + |\hat{W}_{AR}(\omega)|} \tag{3.1}$$

where the subscript AR indicates that the estimations are based on an Auto-Regressive model and $|Y(\omega)|$ is being approximated by the sum of the noise and speech AR-estimate.

3.2 Noise and Speech Estimation

The idea behind the estimation of noise and speech is to use smoothed spectrums to approximate the noisy signal with AR models. This has already been used in papers like [24] to model degraded speech. This sections contains a brief review of AR modeling in relation to signal estimation and a review of how to estimate the speech and noise.

3.2.1 AR Modeling

An AR model of $x(n)$ is a linear prediction model that, given a number of parameters N, predicts the next value of $x(n)$ based on the previous N values of $x(n)$. It is defined as:

$$x(n) = -a_1x(n-1) - a_2x(n-2) - ... - a_Nx(n-N) + \varepsilon(n)$$

where $\varepsilon(n)$ is white noise with variance σ_ε^2 and mean zero and $(a_1, a_2, ..., a_N)$ are the parameters of the process. As can be expected, the model gives good predictions for data with high correlation between data points spaced less than or equal to N points apart.

Transforming the expression to the frequency domain yields:

$$\begin{aligned} X(\omega) &= -X(\omega)(a_1 e^{-j\omega} + a_2 e^{-2j\omega} + ... + a_N e^{-Nj\omega}) + \varepsilon(\omega) \Leftrightarrow \\ X(\omega)(1 + a_1 e^{-j\omega} &+ a_2 e^{-2j\omega} + ... + a_N e^{-Nj\omega}) = \varepsilon(\omega) \Leftrightarrow \\ X(\omega) &= \frac{\varepsilon(\omega)}{1 + a_1 e^{-j\omega} + a_2 e^{-2j\omega} + ... + a_N e^{-Nj\omega}} \end{aligned}$$

From this equation the AR-process can be recognized as an All-pole model or an Infinite Impulse Response (IIR) filter [35]. The power spectrum of such a signal can be estimated as the expected value of $|X(\omega)|^2$:

$$P_{xx} = E[|X(\omega)|^2] = \frac{E[|\varepsilon(\omega)|^2]}{|1 + a_1 e^{-j\omega} + a_2 e^{-2j\omega} + ... + a_N e^{-Nj\omega}|^2} = \frac{\sigma_\varepsilon^2}{|a_x(\omega)|^2} \tag{3.2}$$

where σ_ε^2 is the excitation variance, which is equal to the power of $\varepsilon(n)$ since it is white noise with zero mean and $a_x(\omega) = 1 + \sum_{k=1}^{N} a_k e^{-kj\omega}$.

The parameters $(a_1, a_2, ..., a_N)$ and σ_ε^2 are solved by the Yule-Walker equations [22]:

$$\begin{bmatrix} \gamma_0 & \gamma_{-1} & \cdots & \gamma_{-N+1} \\ \gamma_1 & \gamma_0 & \cdots & \gamma_{-N+2} \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_{N-2} & \gamma_{N-3} & \cdots & \gamma_{-1} \\ \gamma_{N-1} & \gamma_{N-2} & \cdots & \gamma_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_{N-1} \\ a_N \end{bmatrix} = \begin{bmatrix} -\gamma_1 \\ -\gamma_2 \\ \cdots \\ -\gamma_{N-1} \\ -\gamma_N \end{bmatrix} \tag{3.3}$$

$$\sigma_\varepsilon^2 = \gamma_0 + a_1\gamma_1 + a_2\gamma_2 + ... + a_N\gamma_N$$

where $\gamma_0, \gamma_1, ..., \gamma_N$ are the autocorrelation estimates with the index being the lag. These equations are solved efficiently by means of the Levinson-Durbin recursion, which is implemented in the Matlab function *aryule*.

The number of parameters N also governs how smooth the spectrum is, as can be seen in figure 3.1. The plot shows the Periodogram of a 512 sample point speech signal and two corresponding AR spectrums with a different number of parameters.

3.2.2 Minimization-Problem

To estimate the noise and speech in each time-frame, a minimization-problem is solved [41], which is described below. Let P_{yy} be the estimated AR-Power spectrum of the observed noisy signal and let $\hat{P}_{yy}$ be the corresponding power

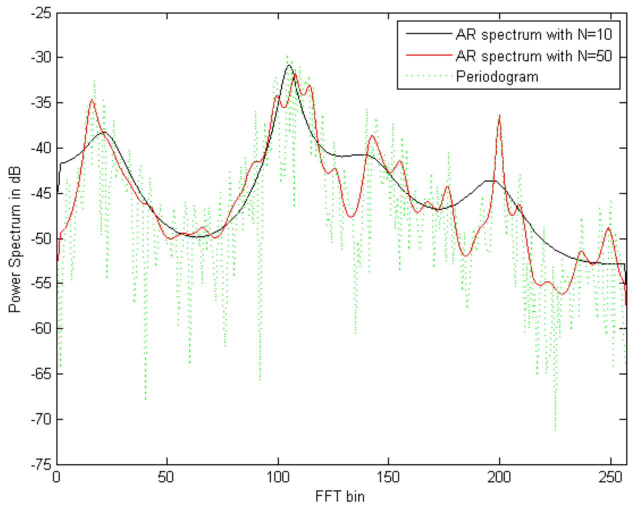


Figure 3.1: *Comparison between power spectrum estimates of a speech signal. Lower order AR-models gives more smooth spectrums.*

spectrum of the modeled signal equal to $\hat{P}_{ss} + \hat{P}_{ww}$, where the spectrum of the noise and speech signals can be evaluated from the AR-coefficients $\theta_s = \{\sigma_s^2, a_1, a_2, ..., a_N\}$ and $\theta_w = \{\sigma_w^2, b_1, b_2, ...b_N\}$. Furthermore define a measure of the difference between the 2 spectra $\mathrm{d}(P_{yy}, \hat{P}_{yy})$. The problem of estimating the speech and noise spectrum used in the spectral subtraction filtering, can then be formulated as:

$$(\hat{\theta}_s, \hat{\theta}_w) = \arg\min_{\theta_s, \theta_w} \mathrm{d}(P_{yy}, \hat{P}_{yy})$$

To solve this, the log-spectral distortion between the sum of the estimated noise and speech power spectrums and the observed noisy spectrum is minimized:

$$\mathrm{d}_{LS} = \frac{1}{2\pi} \int \left| \ln\left(\frac{\sigma_s^2}{|a_s(\omega)|^2} + \frac{\sigma_w^2}{|a_w(\omega)|^2}\right) - \ln\left(\frac{\sigma_y^2}{|a_y(\omega)|^2}\right) \right|^2 \mathrm{d}\omega \tag{3.4}$$

The solution to this minimization problem does not have a unique solution and a global search through all possible combinations would be computationally unfeasible. Instead a codebook that contains the AR-coefficients of the noise and speech spectra that are expected to be found in the noisy signal is introduced. For each combination of speech and noise AR-coefficients, the log-spectral distortion must be evaluated and the set that has the lowest measure is used for the spectral subtraction.

3.2.3 Estimating the Variance

There is an issue with the AR representation power spectrum that has to be addressed before the log spectral distortion can be evaluated. Looking at (3.2) it is seen that the power spectrum consists of the variance of the white noise and the AR-coefficients. The variance is just a constant that shifts the power spectrum up or down in the spectrum, while the shape of the spectrum is governed by the AR-coefficients. Noise can have many different spectral shapes, but it can also have many different levels of energy and therefore it is not a good idea to save the variance of the white noise in the codebook along with the AR-coefficients, as it would necessitate many more entries in the codebook, to obtain a good representation of the noise. Instead, only the AR-coefficients are saved in the codebook and the variance is estimated for each combination of noise and speech spectrum.

It has not been possible to find an explicit derivation of the variance in any paper that mentions this method and therefore it is derived explicitly here.

To estimate the variance, the log spectral distortion is minimized for each set of AR-coefficients. The minimum is found by differentiating the measure with respect to the 2 variances and then setting it equal to zero. First the measure is simplified to make sure that the resulting equations are linear:

$$\begin{aligned}
\mathrm{d}_{LS} &= \frac{1}{2\pi}\int \left|\ln\Big(\frac{\sigma_s^2}{|a_s(\omega)|^2}+\frac{\sigma_w^2}{|a_w(\omega)|^2}\Big)-\ln\Big(\frac{\sigma_y^2}{|a_y(\omega)|^2}\Big)\right|^2 \mathrm{d}\omega = \\
&\frac{1}{2\pi}\int \left|\ln\Big(\frac{|a_y(\omega)|^2}{\sigma_y^2}\Big(\frac{\sigma_s^2}{|a_s(\omega)|^2}+\frac{\sigma_w^2}{|a_w(\omega)|^2}\Big)\Big)\right|^2 \mathrm{d}\omega = \\
&\frac{1}{2\pi}\int \left|\ln\Big(1+\frac{|a_y(\omega)|^2}{\sigma_y^2}\Big(\frac{\sigma_s^2}{|a_s(\omega)|^2}+\frac{\sigma_w^2}{|a_w(\omega)|^2}\Big)-1\Big)\right|^2 \mathrm{d}\omega \approx \\
&\frac{1}{2\pi}\int \left|\frac{|a_y(\omega)|^2}{\sigma_y^2}\Big(\frac{\sigma_s^2}{|a_s(\omega)|^2}+\frac{\sigma_w^2}{|a_w(\omega)|^2}\Big)-1\right|^2 \mathrm{d}\omega = \\
&\frac{1}{2\pi}\int \frac{|a_y(\omega)|^4}{\sigma_y^4}\Big(\frac{\sigma_s^4}{|a_s(\omega)|^4}+\frac{\sigma_w^4}{|a_w(\omega)|^4}+\frac{2\sigma_s^2\sigma_w^2}{|a_s(\omega)|^2|a_w(\omega)|^2}\Big)+ \\
&1-\frac{2|a_y(\omega)|^2}{\sigma_y^2}\Big(\frac{\sigma_s^2}{|a_s(\omega)|^2}+\frac{\sigma_w^2}{|a_w(\omega)|^2}\Big)\mathrm{d}\omega
\end{aligned}$$

where it is used that $\ln(1+z)\approx z$, for small z, i.e. small modeling errors, which is illustrated in figure 3.2. Partial differentiating d_{LS} with respect to σ_s^2 and σ_w^2 and setting it equal to zero yields:

$$\frac{\partial \mathrm{d}_{LS}}{\partial \sigma_s^2} = \frac{1}{2\pi}\int \frac{2\sigma_s^2|a_y(\omega)|^4}{\sigma_y^4|a_s(\omega)|^4}+\frac{2\sigma_w^2|a_y(\omega)|^4}{\sigma_y^4|a_s(\omega)|^2|a_w(\omega)|^2}-\frac{2|a_y(\omega)|^2}{\sigma_y^2|a_s(\omega)|^2}\mathrm{d}\omega = 0 \quad (3.5)$$

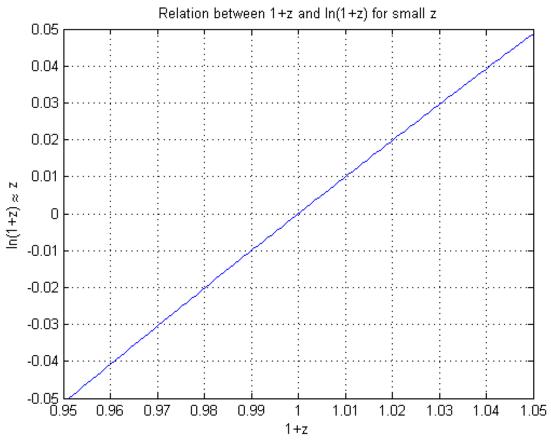


Figure 3.2: *Approximation to* $ln(1+z)$.

$$\frac{\partial \mathrm{d}_{LS}}{\partial \sigma_w^2} = \frac{1}{2\pi}\int \frac{2\sigma_w^2|a_y(\omega)|^4}{\sigma_y^4|a_w(\omega)|^4} + \frac{2\sigma_s^2|a_y(\omega)|^4}{\sigma_y^4|a_s(\omega)|^2|a_w(\omega)|^2} - \frac{2|a_y(\omega)|^2}{\sigma_y^2|a_w(\omega)|^2}\mathrm{d}\omega = 0$$

This set of equations can be rewritten in matrix form:

$$\begin{bmatrix} \int \frac{|a_y(\omega)|^4}{\sigma_y^2|a_s(\omega)|^4}\mathrm{d}\omega & \int \frac{|a_y(\omega)|^4}{\sigma_y^2|a_s(\omega)|^2|a_w(\omega)|^2}\mathrm{d}\omega \\ \int \frac{|a_y(\omega)|^4}{\sigma_y^2|a_s(\omega)|^2|a_w(\omega)|^2}\mathrm{d}\omega & \int \frac{|a_y(\omega)|^4}{\sigma_y^2|a_w(\omega)|^4}\mathrm{d}\omega \end{bmatrix} \begin{bmatrix} \sigma_s^2 \\ \sigma_w^2 \end{bmatrix} = \begin{bmatrix} \int \frac{|a_y(\omega)|^2}{|a_s(\omega)|^2}\mathrm{d}\omega \\ \int \frac{|a_y(\omega)|^2}{|a_w(\omega)|^2}\mathrm{d}\omega \end{bmatrix}$$

The variance of the speech and noise can now be estimated by isolating σ_w^2 and σ_s^2:

$$\begin{bmatrix} \sigma_s^2 \\ \sigma_w^2 \end{bmatrix} = \begin{bmatrix} \int \frac{|a_y(\omega)|^4}{\sigma_y^2|a_s(\omega)|^4}\mathrm{d}\omega & \int \frac{|a_y(\omega)|^4}{\sigma_y^2|a_s(\omega)|^2|a_w(\omega)|^2}\mathrm{d}\omega \\ \int \frac{|a_y(\omega)|^4}{\sigma_y^2|a_s(\omega)|^2|a_w(\omega)|^2}\mathrm{d}\omega & \int \frac{|a_y(\omega)|^4}{\sigma_y^2|a_w(\omega)|^4}\mathrm{d}\omega \end{bmatrix}^{-1} \begin{bmatrix} \int \frac{|a_y(\omega)|^2}{|a_s(\omega)|^2}\mathrm{d}\omega \\ \int \frac{|a_y(\omega)|^2}{|a_w(\omega)|^2}\mathrm{d}\omega \end{bmatrix} \tag{3.6}$$

Negative variances that arise from estimation errors are set to zero.

3.2.4 Calculating Integrals

The system of equations in (3.6) that estimates the excitation variances contains a number of integrals with AR-coefficients. These equations are solved by regarding the expression in the integrals as filters with filter-coefficients equal to the AR-coefficients. The frequency response of each filter is then evaluated in N number of points spaced evenly along the unit circle and the numerical integral is calculated, as illustrated on the second integral:

$$\int \frac{|a_y(\omega)|^4}{\sigma_y^2 |a_s(\omega)|^2 |a_w(\omega)|^2} \mathrm{d}\omega = \frac{1}{\sigma_y^2} \int |H(\omega)|^2 \mathrm{d}\omega \approx \frac{2\pi}{N\sigma_y^2} \sum_{k=1}^{N} |H(\frac{2\pi k}{N})|^2 \qquad (3.7)$$

where $H(k) = \frac{a_y(k)a_y(k)}{a_s(k)a_w(k)}$. The frequency response of $H(k)$ can be evaluated with the Matlab function *freqz*.

3.3 Description of Method

An illustration of the algorithm can be seen in fig. 3.3, where it is broken down into a number of steps:

The signal is divided up into overlapping timeframes of constant width and is

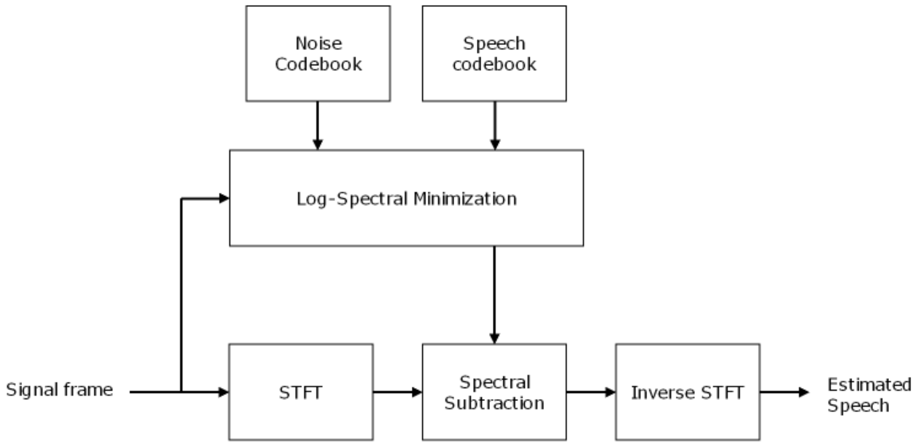


Figure 3.3: *Flowchart of Non-Stationary Spectral Subtraction.*

then routed through 2 different paths; one for estimating the noise and speech part of the signal and the other for holding the original signal in the spectral subtraction part. Each time-frame is then handled individually.

1.1: Before the noise and speech part of the signal is estimated from the signal, the AR-parameters and excitation variance of the original signal frame must be estimated.

1.2: For each pair of noise and speech AR-signals in the codebooks, the excitation variances are calculated by (3.6) and negative variances arising from modeling errors are set equal to zero. The log-spectral distortion is then evaluated by (3.4) and the pair yielding the lowest measure is selected as the one that provides the best spectral fit to the signal frame.

2.1: Along the other path, the signal frame is windowed by a Hamming window of equal length and then an FFT is performed. This is equivalent to performing STFT (Short Time Fourier Transform).

2.2: The Fourier transformed signal is now filtered with the filter (3.1), containing the AR-spectral shapes found in step 1.2. This filtering can be generalized in the same way as the Magnitude Spectral Subtraction algorithm in chapter 2.

2.3: Inverse STFT is then performed to transform the time-overlapping spectral frames into a time signal.

The code for this method has been implemented in Matlab and can be seen in Appendix ??.

3.4 Codebooks

The codebooks used to estimate the speech and noise are generated from separate databases, representing the signals that are expected to be found in the noisy signal. There are two codebooks; one for noise and one for speech each consisting of a matrix. These matrices, who are not necessarily of the same size, contain AR-coefficients representing the shape of the signals they were derived from. Each row in the matrix is a set of AR-coefficients and the number of columns denotes the number of coefficients used to represent the training set:

$$CC_{speech} = \begin{bmatrix} 1 & a_1 & a_2 & \dots & a_N \\ 1 & b_1 & b_2 & \dots & b_N \\ 1 & c_1 & c_2 & \dots & c_N \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \qquad CC_{noise} = \begin{bmatrix} 1 & d_1 & d_2 & \dots & d_M \\ 1 & e_1 & e_2 & \dots & e_M \\ 1 & f_1 & f_2 & \dots & f_M \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

Segments of the same length and overlap as the input to the Non-Stationary Spectral Subtraction algorithm are sampled from the speech and noise part of the training set and used to generate the two codebooks. For each segment the corresponding AR-model is estimated using Yule-Walkers equations (3.3) and stored in the codebooks. Because the variance is estimated for each timeframe, only the AR-coefficients are saved in the codebook.

These codebooks, however, contains a lot of redundant data, since many instances of the same noise and speech is present and will only increase the amount of time needed to search through the codebooks. Therefore a method to decrease the size of the codebooks is needed. When performing this reduction, it is important that as much of the representation in the training set is kept and for that the k-means algorithm [20] is used. This algorithm clusters the AR-coefficients into k M-dimensional cluster centers (where M is the number of AR-coefficients), by an iterative procedure. An artificial example of a k-means clustering can be seen in figure 3.4. In the example 2-dimensional data has been clustered by k=3 cluster centers. The red dots are cluster centers and the numbers are data points, where the numbers indicate which cluster center the point belongs to. In the k-means algorithm a point is assigned to the cluster with the shortest Euclidean distance. It can be shown that the k-means clustering is actually a special case of the gaussian mixture model [4]. After the k-means algorithm has been applied, the new codebooks only contain the cluster centers found by the iterative scheme ($k \ll$ number of original spectrum-entries).

In the non-stationary Spectral Subtraction algorithm, the codebooks are only

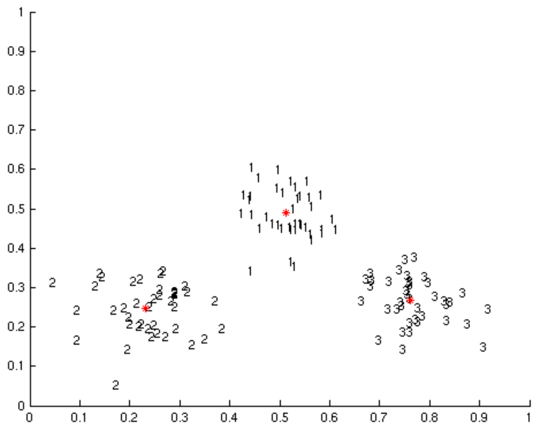


Figure 3.4: *K-means clustering on 2-dimensional data with k=3.*

used to evaluate powerspectrums in the integrals in (3.7) and the spectral distortion in (3.4). Therefore computation time can be saved, by precomputing the spectrums from the codebook using the freqz command in Matlab as mentioned in section 3.2.4. This will, however, increase the amount of memory needed to store the codebooks.

3.4.1 Searching the Codebooks

The use of codebooks enables the estimation of noise even while speech is present, but it also increases the computation time needed as the method searches through the codebook for the best fit. To keep the computation cost of the method at a certain level, it might therefore be necessary to limit the size of the codebook, thereby possibly reducing the performance of the noise estimation. This motivates the implementation of more intelligent searching strategies in the codebook.

The most intuitive way to search the codebook is to compare each spectrum in the speech codebook with each spectrum in the noise codebook and find the pair with the lowest spectral distortion according to (3.4). This brute-force method, however, is very computationally inefficient with an upper-bound of $O(K_s \cdot K_w)$, where K_s and K_w is the number of speech and noise spectrums in the codebooks respectively.

An alternative searching scheme can instead be implemented that reduces the computation complexity significantly. For each time frame, a noise estimate must be obtained using any noise estimation technique, for instance the one in chapter 2.1. Based on this noise estimate, the entire speech codebook is searched to find the best fit that minimizes the spectral distortion (3.4). Then using this speech entry in the speech codebook, the entire noise codebook is searched to find the best fit according to the spectral distortion. Again the speech codebook is searched using the new noise estimate and this procedure is repeated until the spectral distortion has converged. The obtained noise and speech shapes are then used to filter that noisy frame. The upper-bound using this approach is $O(K_s + K_w)$ and in practice it is found that it is only necessary to search through each codebook about 2-4 times.

Chapter 4

Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (sometimes called Non-Negative Matrix Approximation) is a relatively new method that has many potential application areas, for instance Image Processing, Text Analysis and Blind Source Separation, see [29] and [40] for a general review of the method with applications. The method first became popular in 1999 with the article [23] by Lee and Seung. The idea behind the method is to factorize a matrix Λ into a product of two matrices D and C. The usual factorization is interpreted as a dictionary matrix D that contains the different possible activations that occur in Λ in each column and C is a code matrix that contains information about where in Λ the activations occur:

$$\Lambda = D \cdot C$$

A simple example of a factorization is:

$$\begin{bmatrix} 1 & 1 & 2 & 0 & 11 & 2 & 3 \\ 2 & 1 & 3 & 0 & 8 & 4 & 2 \\ 3 & 1 & 4 & 0 & 5 & 6 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3 \\ 2 & 1 & 2 \\ 3 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 1 \end{bmatrix}$$

Λ contains different combinations of the 3 basis vectors (3 columns) in D and C contains information about where in Λ the different basisvectors are. As the name implies only non-negative numbers are allowed in the matrices, which can be interpreted as if D contains magnitudes that can only be added together (because C is non-negative) to get Λ. In the context of wind noise filtering, Λ is the magnitude spectrogram, D contains spectral magnitude shapes that belongs to either speech or noise and C contains information about where the respective spectral shapes in D occur.

A number of observations can be made about the factorization example:

- The factorization of Λ into D and C is not unique. For example the first column of D can be divided by 2 if the first row of C is multiplied by 2.
- Column 2 in Λ could be represented by a sum of all basisvectors in D, but to keep the interpretation of the factorization simple, C should be as sparse as possible (contain as many zeros as possible).
- Highly trivial and undesirable factorizations can be found, for instance the factorization where D is a unity matrix and C is equal to Λ.

The first problem can be avoided by making sure D is normalized:

$$D = D/||D||$$

where $||\cdot||$ is the Euclidean norm. The 2 other problems are related, because by keeping C sparse, as much information will be put into D as possible and undesirable factorizations will hopefully be avoided. By putting as much information as possible into D, the interpretation of D as a basis matrix is also strengthened. The problem of keeping C sparse will be dealt with later.

4.1 Defining a Cost Function

Due to the recent popularity of this method, many different suggestions for obtaining a factorization of Λ exists. Most of the methods, however, minimize a least squares costfunction, but without mentioning the explicit assumptions made about this costfunction. In the following a derivation of the cost function that is inspired by [38] is followed. It assumes that the reader has some knowledge of probability theory.

4.1.1 Maximum Likelihood Estimate

The problem of finding a factorization can be stated as:

$$V = \Lambda + \varepsilon = D \cdot C + \varepsilon$$

where $\varepsilon \in \mathbb{R}^{K \times L}$ is residual noise, $V \in \mathbb{R}_{+}^{K \times L}$ is the data matrix and $\Lambda \in \mathbb{R}_{+}^{K \times L}$ is the factorized approximation to V.

The Maximum Likelihood (ML) estimate of D and C is equal to the minimum of the negative log-likelihood:

$$(\hat{D}, \hat{C}) = \underset{D,C>0}{\arg\min}\, \mathcal{L}_{V|D,C}(D, C)$$

where $\mathcal{L}_{V|D,C}(D, C)$ is the negative log-likelihood of D and C. The likelihood depends on the residual noise ε. If the noise is assumed to be independent identically distributed (i.i.d) Gaussian noise with variance σ_{ε}^2, the likelihood can be written as:

$$p(V|D, C) = \frac{1}{(\sqrt{2\pi}\sigma_{\varepsilon})^{KL}} \exp(-\frac{||V - D \cdot C||^2}{2\sigma_{\varepsilon}^2})$$

which is basically as gaussian distribution over the noise. From this it is seen that the negative log-likelihood is:

$$\mathcal{L}_{V|D,C}(D, C) \propto \frac{1}{2}||V - D \cdot C||^2$$

This expression is known as a least squares function and a factorization of Λ can be found by using it as a costfunction that should be minimized:

$$CC_{LS} = \frac{1}{2}||V - D \cdot C||^2 = \frac{1}{2} \sum_i \sum_j (V_{i,j} - \sum_k D_{i,k} \cdot C_{k,j})^2 \qquad (4.1)$$

where the indices denotes elements in the matrices. Other kinds of noise assumptions, leads to other cost functions, for instance would a Poisson noise assumption lead to the following cost function [32]:

$$CC_{KL} = \sum_i \sum_j V_{i,j} \cdot \log\Big(\frac{V_{i,j}}{\sum_k D_{i,k} \cdot C_{k,j}}\Big) - V_{i,j} + \sum_k D_{i,k} \cdot C_{k,j} \qquad (4.2)$$

which is known as the Kullback-Leibler divergence.

4.1.2 Enforcing Sparsity

The costfunctions derived so far has no sparsity build into them. As long as Λ is a good approximation of V, it does not take into consideration if the found factorization is meaningful. A way to implement this is to include prior knowledge about the code matrix C into the estimation using maximum a posteriori (MAP) estimates. Using Bayes rule, the posterior is given by:

$$p(D, C|V) = \frac{p(V|D, C) \cdot p(D, C)}{p(V)}$$

Given V the numerator is constant and the minimum of the negative logarithm of the posterior $p(D,C|V)$ is seen to be proportional to a sum of the negative log-likelihood (the ML estimate) and the negative logarithm of a prior term $p(D,C)$ that can be used to penalize solutions that are undesired:

$$\mathcal{L}_{D,C|V}(D,C) \propto \mathcal{L}_{V|D,C}(D,C) + \mathcal{L}_{D,C}(D,C) \tag{4.3}$$

A way to impose a sparse representation on C would then be to introduce an exponential prior over C:

$$p(D,C) = \prod_{i,j} \lambda \cdot \exp(-\lambda C_{i,j})$$

A plot for the exponential prior over 1 dimension and for $\lambda = 0.2$ can be seen in figure 4.1. The negative log-likelihood of the exponential prior is:

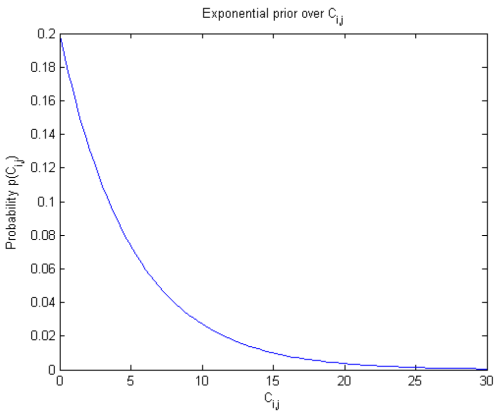


Figure 4.1: *Exponential prior for one element in* C *with* $\lambda = 0.2$. *As can be seen the prior favors small values of* C.

$$\mathcal{L}_{D,C}(D,C) \propto -\log\left(\prod_{i,j} \exp(-\lambda C_{i,j})\right) = \lambda \sum_{i,j} C_{i,j}$$

According to equation (4.3), this term can be added to the negative log-likelihood to give a posterior estimate that enforces sparsity. Adding this term to the cost functions in equation (4.1) and (4.2) gives:

$$CC_{LS} = \frac{1}{2}\sum_i \sum_j (V_{i,j} - \sum_k D_{i,k} \cdot C_{k,j})^2 + \lambda \cdot C_{i,j} \tag{4.4}$$

$$CC_{KL} = \sum_i \sum_j V_{i,j} \cdot \log\left(\frac{V_{i,j}}{\sum_k D_{i,k} \cdot C_{k,j}}\right) - V_{i,j} + \sum_k D_{i,k} \cdot C_{k,j} + \lambda \cdot C_{i,j} \tag{4.5}$$

The regularization parameter λ determines how much large values in C should be penalized.

4.2 Minimizing the Costfunction

The method of minimizing the sparse costfunction given in this section is using multiplicative update rules that are derived from the gradient descent method. This method iteratively updates the estimates of D and C to arrive at a solution that minimizes the costfunction. It is inspired by articles like [37], but is also using sparseness constraints. Updating rules for sparse costfunctions like equation (4.4) can be found in [10] and [13], but to the knowledge of the author no paper exists that actually derive the update rules for sparse costfunctions. Therefore they will be derived here, but only for equation (4.4) as the approach for (4.5) is exactly the same.

Looking at equation (4.4) it is seen that there is a potential pitfall in the minimization process, because of the added sparsity constraint. By dividing C with a constant larger than one and multiplying the same constant onto D, the sparsity term will decrease while the first ML term will remain the same. This means that any minimization algorithm can decrease the costfunction by letting C go towards zero and proportionately increasing D. This numerical instability can be avoided by normalizing D before each new update and introducing a normalization of D in the costfunction:

$$CC_{LS} = \frac{1}{2}\sum_i \sum_j (V_{i,j} - \sum_k \frac{D_{i,k}}{||D_k||} \cdot C_{k,j})^2 + \lambda \cdot C_{i,j} \tag{4.6}$$

$D_{i,k}$ is normalized with the Euclidean norm of the corresponding row and now the cost function is invariant to the scaling of D. In the derivation of the update steps to C, some intermediate derivatives are needed:

$$\frac{\partial \sum_k \frac{D_{i,k}}{||D_k||} \cdot C_{k,j}}{\partial C_{l,d}} = \begin{cases} \frac{D_{i,l}}{||D_l||} = D_{i,l} & \text{if } j = d \\ 0 & \text{if } j \neq d \end{cases}$$

$$\frac{\partial CC_{LS}}{\partial C_{l,d}} = \frac{\partial \frac{1}{2} \sum_i \sum_j (V_{i,j} - \sum_k \frac{D_{i,k}}{||D_k||} \cdot C_{k,j})^2 + \lambda \cdot C_{i,j}}{\partial C_{l,d}} =$$

$$-\sum_i \left[(V_{i,d} - \sum_k D_{i,k} \cdot C_{k,d}) \cdot D_{i,l} \right] + \lambda$$

where it is used that $||D_k|| = 1$ because D has just been normalized before each update. Now the derivative of the costfunction with respect to an element in D is found, with a few intermediate derivatives:

$$\frac{\partial ||D_k||}{\partial D_{l,d}} = \frac{\partial \sqrt{D_{1,k}^2 + D_{2,k}^2 + ...D_{l,k}^2 + ...}}{\partial D_{l,d}} = \begin{cases} \frac{2D_{l,d}}{2||D_d||} = D_{l,d} & \text{if } k = d \\ 0 & \text{if } k \neq d \end{cases}$$

$$\frac{\partial \sum_k \frac{D_{i,k}}{||D_k||} \cdot C_{k,j}}{\partial D_{l,d}} = \begin{cases} \frac{||D_d|| - D_{l,d}^2}{||D_d||^2} \cdot C_{d,j} = (1 - D_{l,d}^2) \cdot C_{d,j} & \text{if } i = l \\ \frac{-D_{i,d} \cdot D_{l,d}}{||D_d||^2} \cdot C_{d,j} = -D_{i,d} \cdot D_{l,d} \cdot C_{d,j} & \text{if } i \neq l \end{cases}$$

$$\frac{\partial CC_{LS}}{\partial D_{l,d}} = \frac{\partial \frac{1}{2} \sum_i \sum_j (V_{i,j} - \sum_k \frac{D_{i,k}}{||D_k||} \cdot C_{k,j})^2 + \lambda \cdot C_{i,j}}{\partial D_{l,d}} =$$

$$-\sum_j \Big[(V_{l,j} - \sum_k D_{l,k} \cdot C_{k,j}) \cdot C_{d,j}\Big] + \sum_i \sum_j \Big[(V_{i,j} - \sum_k D_{i,k} \cdot C_{k,j}) \cdot D_{i,d} \cdot D_{l,d} \cdot C_{d,j}\Big]$$

The second term in $\frac{\partial CC_{LS}}{\partial D_{l,d}}$ would vanish if D was not normalized with the Euclidean norm in the cost function and can be considered a correction term for the numerical stability. For the rest of the derivation a simpler notation can be used, by rewriting the two derivatives of the costfunction into matrix-form:

$$\frac{\partial CC_{LS}}{\partial \boldsymbol{C}} = -\boldsymbol{D}^T(\boldsymbol{V} - \boldsymbol{\Lambda}) + \lambda$$

$$\frac{\partial CC_{LS}}{\partial \boldsymbol{D}} = -(\boldsymbol{V} - \boldsymbol{\Lambda})\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot ((\boldsymbol{V} - \boldsymbol{\Lambda})\boldsymbol{C}^T))$$

where $\boldsymbol{\Lambda} = \boldsymbol{DC}$, 1 is a square matrix of ones of suitable size, $\odot$ is the elementwise multiplication operator like $.*$ in Matlab and ^{T} is the transpose operator. All matrix divisions in this chapter are elementwise divisions, like the ./ operator in Matlab.

Now the update steps can be found by gradient descent. This method consists of initializing $\boldsymbol{C}$ and $\boldsymbol{D}$ to some value and then iteratively updating them, by taking steps in the direction of the negative gradient of the costfunction:

$$\boldsymbol{C} \leftarrow \boldsymbol{C} - \eta_C \odot \frac{\partial CC_{LS}}{\partial \boldsymbol{C}}$$

$$\boldsymbol{D} \leftarrow \boldsymbol{D} - \eta_D \odot \frac{\partial CC_{LS}}{\partial \boldsymbol{D}}$$

As the gradient points in the direction of steepest ascent, following the negative gradient guarantees that the costfunction will decrease. η is the stepsize, which should be chosen with great care. A too small step size will make the algorithm very slow in reaching a minimum and too large stepsizes will make the algorithm overshoot the minimum and possibly end up in a new position with higher cost than before. To obtain the multiplicative update rules, η_C is chosen so that the two terms in the gradient descent can be contracted:

$$\eta_C = \frac{\boldsymbol{C}}{\boldsymbol{D}^T\boldsymbol{\Lambda} + \lambda}$$

$$\boldsymbol{C} - \eta_C \odot \frac{\partial CC_{LS}}{\partial \boldsymbol{C}} = \boldsymbol{C} - \boldsymbol{C} \odot \frac{-\boldsymbol{D}^T(\boldsymbol{V} - \boldsymbol{\Lambda}) + \lambda}{\boldsymbol{D}^T\boldsymbol{\Lambda} + \lambda} =$$

$$\boldsymbol{C} \odot \left(1 - \frac{\boldsymbol{D}^T\boldsymbol{\Lambda} + \lambda}{\boldsymbol{D}^T\boldsymbol{\Lambda} + \lambda} + \frac{\boldsymbol{D}^T\boldsymbol{V}}{\boldsymbol{D}^T\boldsymbol{\Lambda} + \lambda}\right) =$$

$$\boldsymbol{C} \odot \frac{\boldsymbol{D}^T\boldsymbol{V}}{\boldsymbol{D}^T\boldsymbol{\Lambda} + \lambda}$$

The same strategy is applied for η_D:

$$\eta_D = \frac{\boldsymbol{D}}{\boldsymbol{\Lambda}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{V}\boldsymbol{C}^T)))}$$

$$\boldsymbol{D} - \eta_D \odot \frac{\partial CC_{LS}}{\partial \boldsymbol{D}} = \boldsymbol{D} - \boldsymbol{D} \odot \frac{-(\boldsymbol{V} - \boldsymbol{\Lambda})\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot ((\boldsymbol{V} - \boldsymbol{\Lambda})\boldsymbol{C}^T))}{\boldsymbol{\Lambda}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{V}\boldsymbol{C}^T)))} =$$

$$\boldsymbol{D} \odot \left(1 - \frac{\boldsymbol{\Lambda}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{C} \odot (\boldsymbol{V}\boldsymbol{C}^T)))}{\boldsymbol{\Lambda}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{V}\boldsymbol{C}^T)))} + \frac{\boldsymbol{V}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{\Lambda}\boldsymbol{C}^T)))}{\boldsymbol{\Lambda}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{V}\boldsymbol{C}^T)))}\right) =$$

$$\boldsymbol{D} \odot \frac{\boldsymbol{V}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{\Lambda}\boldsymbol{C}^T)))}{\boldsymbol{\Lambda}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{V}\boldsymbol{C}^T)))}$$

4.3 Complete NNMF Algorithm

The complete algorithm for performing Sparse Non-Negative Matrix Factorization (NNMF) using a least squares costfunction and multiplicative update rules is:

Least Squares NNMF Sparse Multiplicative Update Rules

1. Initialize C and D to be random strictly positive matrices

2. Normalize D to unity

$$D_{i,j} = \frac{D_{i,j}}{||D_j||}$$

3. Update C according to

$$\boldsymbol{C} \leftarrow \boldsymbol{C} \odot \frac{\boldsymbol{D}^T\boldsymbol{V}}{(\boldsymbol{D}^T\boldsymbol{D})\boldsymbol{C} + \lambda}$$

4. Update D according to

$$\boldsymbol{D} \leftarrow \boldsymbol{D} \odot \frac{\boldsymbol{V}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{D}(\boldsymbol{C}\boldsymbol{C}^T))))}{\boldsymbol{D}(\boldsymbol{C}\boldsymbol{C}^T) + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\boldsymbol{V}\boldsymbol{C}^T)))}$$

5. Check for convergence and maximum number of iterations, otherwise return to 2

Here $\boldsymbol{\Lambda}$ has been replaced with $\boldsymbol{DC}$ and parentheses have been put in to show that the multiplications $\boldsymbol{D}^T\boldsymbol{D}$ and $\boldsymbol{C}\boldsymbol{C}^T$ should be performed first to save computationtime. This is because the number of columns in D (the number of basisvectors), which is equal to the number of rows in C, is expected to be a lot smaller than the other dimension of D and C. The update rules using the Kullbach-Leibler divergence as costfunction can be derived in exactly the same way as done with the least squares costfunction and are:

Kullbach-Leibler NNMF Sparse Multiplicative Update Rules

1. Initialize C and D to be random strictly positive matrices
2. Normalize D to unity

$$D_{i,j} = \frac{D_{i,j}}{||D_j||}$$

3. Update C according to

$$\boldsymbol{C} \leftarrow \boldsymbol{C} \odot \frac{\boldsymbol{D}^T \frac{\boldsymbol{V}}{\boldsymbol{DC}}}{\boldsymbol{D}^T 1 + \lambda}$$

4. Update D according to

$$\boldsymbol{D} \leftarrow \boldsymbol{D} \odot \frac{\frac{\boldsymbol{V}}{\boldsymbol{DC}}\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (1\boldsymbol{C}^T)))}{1\boldsymbol{C}^T + \boldsymbol{D} \odot (1(\boldsymbol{D} \odot (\frac{\boldsymbol{V}}{\boldsymbol{DC}}\boldsymbol{C}^T)))}$$

5. Check for convergence and maximum number of iterations, otherwise return to 2

There is a strong resemblance between the two update algorithms. A small dither should be added to all numerators to avoid division by zero. A significant advantage of the multiplicative update equations is that as long as C and D is initialized as positive matrices, they will remain positive, because the matrices are only multiplied with a positive constant in the update equations. This ensures that the factorization is always non-negative. Also the method is relatively easy to implement and has a simple formulation. Other methods that uses second order information (the Hessian matrix) might have faster convergence time, for instance Alternating least squares projected gradient [26], Quasi-Newton optimization [45], Fast Newton-type [7] or SMART [2] methods.

4.3.1 Convergence

According to step 5 in the NNMF multiplicative update algorithm, convergence should be checked for each iteration. A way to do this is to check, if the relative change in the cost function in relation to the last iteration is smaller than a certain threshold value ϵ:

$$\epsilon > \frac{CC^n - CC^{n-1}}{CC^n}$$

where n is the iteration number. According to [29] the multiplicative update equations guarantees convergence to a stationary point that is either a (local) minimum or a saddle point. Saddle points are usually undesirable, but the equations have been shown to give good factorizations in just a single run [31].

4.3.2 Accelerated Convergence

To speed up the convergence, the multiplicative term can be exponentiated with an acceleration parameter δ larger than one. For each iteration the costfunction is evaluated and if the cost is smaller than in the previous iteration, δ is increased and if it is larger, then δ is decreased. Illustrated on the update equations for the least squares C expression:

1. Initialize δ to some value larger than 1
2. Update C according to
$$\boldsymbol{C} \leftarrow \boldsymbol{C} \odot \left(\frac{\boldsymbol{D}^T\boldsymbol{V}}{(\boldsymbol{D}^T\boldsymbol{D})\boldsymbol{C} + \lambda}\right)^{\delta}$$
3. if $CC_{LS}^{n} \leq CC_{LS}^{n-1}$
   then $\delta \leftarrow \delta^2$
   else $\delta \leftarrow \max(\frac{\delta}{2}, 1)$ and return to step 2
4. Check for convergence and maximum number of iterations, otherwise return to 2

Individual acceleration parameters can be used for C and D. When the algorithm has converged, δ should be set to one and the update should be run one more time to make sure that the algorithm did not overshoot a minimum because of a too large stepsize.

4.4 A-weighted Costfunction

To include some perceptual knowledge about how the ear perceives sound, the costfunction can be A-weighted [34]. This is a weighting over frequency according to how sensitive the ear is to that corresponding frequency and is related to perceptual loudness.The frequency response of the A-weighting filter is given by:
$$Aw(s) = \frac{7.39705 \cdot 10^9 \cdot s^4}{(s + 129.4)^2(s + 676.7)(s + 4636)(s + 76655)^2}$$
A magnitude plot of the A-weighting curve can be seen in figure 4.2. It is seen that the Aw filter gives high weight to frequencies from 1000-6000Hz and very low weight to low frequency signals. It should be noted that the A-weighting curve is a very simple model of the frequency-dependent sensitivity of the ear and is really only valid for relatively quiet pure tone signals. Wind noise contains a lot of low-frequency content, as will be shown in chapter 6, which can be trivially removed from speech using a highpass filter. Therefore it could potentially improve the factorization by A-weighting the cost function:
$$CC_{LS} = \frac{1}{2}\sum_i\sum_j Aw_i \cdot (V_{i,j} - \sum_k \frac{D_{i,k}}{||D_k||} \cdot C_{k,j})^2 + \lambda \cdot C_{i,j}$$

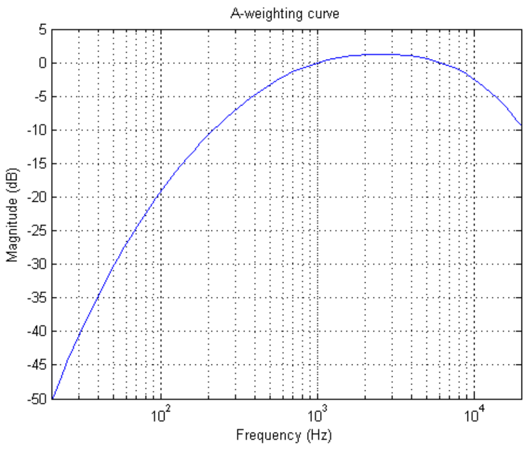


Figure 4.2: *A-weighting filter as a function of frequency.*

where Aw_i is the corresponding A-weighting for that particular frequency. This can also be done in a similar way for the Kullbach Leibler divergence costfunction. The derivative of the least squares costfunction with respect to $C_{l,d}$ and $D_{l,d}$ is:

$$\frac{\partial CC_{LS}}{\partial C_{l,d}} = \frac{\partial \frac{1}{2}\sum_i \sum_j Aw_i \cdot (V_{i,j} - \sum_k \frac{D_{i,k}}{||D_k||} \cdot C_{k,j})^2 + \lambda \cdot C_{i,j}}{\partial C_{l,d}} =$$

$$-\sum_i Aw_i \cdot \Big[(V_{i,d} - \sum_k D_{i,k} \cdot C_{k,d}) \cdot D_{i,l}\Big] + \lambda$$

$$\frac{\partial CC_{LS}}{\partial D_{l,d}} = \frac{\partial \frac{1}{2}\sum_i \sum_j Aw_i \cdot (V_{i,j} - \sum_k \frac{D_{i,k}}{||D_k||} \cdot C_{k,j})^2 + \lambda \cdot C_{i,j}}{\partial D_{l,d}} =$$

$$-\sum_j \Big[Aw_l \cdot (V_{l,j} - \sum_k D_{l,k} \cdot C_{k,j}) \cdot C_{d,j}\Big] + \sum_i \sum_j Aw_i \cdot \Big[(V_{i,j} - \sum_k D_{i,k} \cdot C_{k,j}) \cdot D_{i,d} \cdot D_{l,d} \cdot C_{d,j}\Big]$$

From this it is seen that the multiplicative update rules can be A-weighted by multiplying $V_{i,j}$ and $\Lambda_{i,j} = \sum_k D_{i,k} \cdot C_{k,j}$ with Aw_i. The A-weighting has a numerical problem, because the weight for the lowest frequencies approaches zero. This can be avoided by setting a lower bound on the weight, for instance 0.25.

4.5 Sparse NNMF in Wind Noise Reduction

When using Sparse NNMF to filter noise it is assumed that speech and noise is additive:

$$\boldsymbol{V} = \boldsymbol{S} + \boldsymbol{W} = \boldsymbol{DC} = \begin{bmatrix} \boldsymbol{D_s} & \boldsymbol{D_w} \end{bmatrix} \begin{bmatrix} \boldsymbol{C_s} \\ \boldsymbol{C_w} \end{bmatrix}$$

where $\boldsymbol{V}$ is the magnitude spectrogram of the noisy signal, $\boldsymbol{S}$ is the magnitude spectrogram of the speech, $\boldsymbol{W}$ is the magnitude spectrogram of the wind noise, $\boldsymbol{D}$ and $\boldsymbol{C}$ is the factorization result of the NNMF algorithm and the indices s and w indicate the part of the factorization that belongs to the speech and wind noise respectively.

To filter the noise away, $\boldsymbol{V}$ is given as input to the NNMF algorithm, which then estimates $\boldsymbol{D}$ and $\boldsymbol{C}$. The estimated speech is then obtained by multiplying together $\boldsymbol{D_s}$ and $\boldsymbol{C_s}$:

$$\hat{\boldsymbol{S}} = \boldsymbol{D_s C_s}$$

Then the phase of the noisy signal is used to convert the speech estimate back to the time domain. A potentially better estimate can be obtained by estimating the noise by multiplying $\boldsymbol{D_w}$ and $\boldsymbol{C_w}$ and then filter $\boldsymbol{V}$ using the generalized Spectral Subtraction algorithm described in chapter 2. In this case NNMF is used as an advanced noise estimator. Like in the Generalized Spectral Subtraction algorithm, the magnitude spectrogram can also be exponentiated before the factorization is calculated:

$$\boldsymbol{V}^\gamma = \boldsymbol{S}^\gamma + \boldsymbol{W}^\gamma = \boldsymbol{D}^\gamma \boldsymbol{C}^\gamma = \begin{bmatrix} \boldsymbol{D_s}^\gamma & \boldsymbol{D_w}^\gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{C_s}^\gamma \\ \boldsymbol{C_w}^\gamma \end{bmatrix}$$

This, however, breaks the additive assumption of $\boldsymbol{S}$ and $\boldsymbol{W}$. A remaining problem that has not been mentioned is how to figure out, which part of the factorization belongs to the speech and which part belongs to the noise. This will be addressed in the following section.

4.6 Noise Codebooks

In order to estimate the speech and noise part of an NNMF it must be known which part of the factorization matrices $\boldsymbol{D}$ and $\boldsymbol{C}$ belongs to speech and which part to wind noise. Any 2 columns in $\boldsymbol{D}$ can be switched around as long as the corresponding rows in $\boldsymbol{C}$ are switched correspondingly. This permutation ambiguity makes it impossible to know which part of the NNMF belongs to speech and noise without making some sort of inspection of them. A way around this problem is to precompute the noise part of $\boldsymbol{D}$ before NNMF is run. $\boldsymbol{D_w}$ can be estimated by running NNMF on pure noise that is expected to be found in the noisy signal $\boldsymbol{V}$. If a number of wind noise files are used for computing the wind noise dictionary $\boldsymbol{D_w}$, then these files can be concatenated into one file and then given as input to the NNMF algorithm using another sparsity parameter λ_{pre}. The resulting basismatrix is then the noise codebook $\boldsymbol{D_w}$ that can be put into $\boldsymbol{D}$ before running the NNMF on the noisy signal $\boldsymbol{V}$.

As $\boldsymbol{D_w}$ is now kept fixed, the update equations for the least squares costfunction can be separated into:

$$\boldsymbol{C_s} \leftarrow \boldsymbol{C_s} \odot \frac{\boldsymbol{D_s}^T\boldsymbol{V}}{(\boldsymbol{D_s}^T\boldsymbol{D})\boldsymbol{C} + \lambda_s}, \quad \boldsymbol{C_w} \leftarrow \boldsymbol{C_w} \odot \frac{\boldsymbol{D_w}^T\boldsymbol{V}}{(\boldsymbol{D_w}^T\boldsymbol{D})\boldsymbol{C} + \lambda_w}$$

$$\boldsymbol{D_s} \leftarrow \boldsymbol{D_s} \odot \frac{\boldsymbol{V}\boldsymbol{C_s}^T + \boldsymbol{D_s} \odot (1(\boldsymbol{D_s} \odot (\boldsymbol{D}(\boldsymbol{C}\boldsymbol{C_s}^T))))}{\boldsymbol{D}(\boldsymbol{C}\boldsymbol{C_s}^T) + \boldsymbol{D_s} \odot (1(\boldsymbol{D_s} \odot (\boldsymbol{V}\boldsymbol{C_s}^T)))}$$

and similarly for the Kullbach-Leibler divergence. Different sparsity parameters λ_s and λ_w is introduced as it might improve the factorization.

As in the Non-Stationary Spectral Subtraction (NSS) algorithm in chapter 3, speech could also be precomputed and put into the basismatrix. This, however, might not be a good idea idea. The NSS algorithm uses smooth spectrum-estimates in the codebooks to estimate the noise and speech, while the NNMF algorithm uses exact shapes of the signals. So while the NSS algorithm filters the signal using smooth filters, the NNMF algorithm might not be able to find a good match of the current signal in the codebook, especially if the noisy signal contains other kinds of signals than just speech and noise.

4.7 Real-Time NNMF

The method reviewed so far estimates the factorization over an entire magnitude spectrogram and thus assumes that the noisy signal is known for all times. This is not possible in a real-time implementation. Instead the method can be changed so it uses a sliding window of size L_f over the frames of the magnitude spectrogram. Each time the sliding window is moved one frame, the factorization must be recalculated. Based on this new factorization, the newest arriving frame in the window can be filtered. This, however, is a very costly procedure and can be improved by considering that all frames in the window except for the newest frame already has been factorized. Therefore the old factorization can be used as a starting guess for the new factorization, which will lead to a much faster convergence of the NNMF algorithm.

CHAPTER 5

Evaluation Methods

When evaluating how the methods perform it is important to use relevant measures. Often papers give Signal-to-Noise ratios (SNR) of how their method performs, but rarely has the perceptual comprehension of the output of the method been subjected to an objective evaluation. These two concepts does not have a simple relationship and in [31] it was even found that the method that improved the speech recognition the most in a noise reduction comparison between 3 methods, actually decreased the SNR.

In this chapter different ways to evaluate the output of a noise reduction algorithm will be reviewed. The chapter covers different objective evaluations, subjective evaluations and an objective perceptual evaluation, which is an automated methodology for capturing the perceptual quality in the subjective evaluations.

5.1 Objective Evaluation

The most commonly used objective measure for evaluating noise reduction algorithms is the Signal-to-Noise measure. Different variations of this measure will be reviewed in this section.

5.1.1 Signal-to-Noise Ratio

The input Signal-to-Noise ratio is defined as:

$$SNR_i = 10\log\frac{\sum_n s(n)^2}{\sum_n w(n)^2}$$

where $s(n)$ is the clean speech and $w(n)$ the wind noise. The SNR is basically a ratio between the energy in the speech and the energy in the wind noise.

The output of a noise reduction algorithm is the speech estimate $\hat{s}(n)$. The output speech error is then given as:

$$s_o(n) = s(n) - \hat{s}(n)$$

$s_o(n)$ contains the wind noise that has not been filtered, but it also contains the part of s that has been removed by estimation errors in the algorithm. The output SNR is then:

$$SNR_o = 10\log\frac{\sum_n s(n)^2}{\sum_n s_o(n)^2}$$

It should be noted that the only difference between SNR_i and SNR_o is the noise, because the desired signal is the same in both cases.

Another measure, the SNR_{seg} can be found by calculating the SNR in small frames and then averaging over all frames:

$$SNR_{seg} = \frac{1}{M}\sum_{m=1}^{M} 10\log\frac{\sum_n s(n,m)^2}{\sum_n s_o(n,m)^2}$$

where m indicates the frame number and n is the sample number in frame m. Before averaging, all frames with a speech or noise level below a certain threshold ϵ_t should be left out.

A similar measure, the Noise Residual NR, can be measured as:

$$NR_o = 10\log\frac{\sum_n w(n)^2}{\sum_n w_o(n)^2}$$

where $w_o = w(n) - \hat{s}(n)$ is the difference between the true wind noise and the output. This is a measure of how much the output resembles the wind noise. A good noise reduction methods needs to have both a high SNR_o and a low NR_o. This measure can also be modified to a segmented version.

To include some perceptual knowledge about how the ear perceives sound, the energy in the speech and noise can be A-weighted [34]. This is a weighting over frequency according to how sensitive the ear is to that corresponding frequency and is related to perceptual loudness. A magnitude plot of the A-weighting curve can be seen in figure 5.1. It is seen that the Aw filter gives high weight to frequencies from 1000-6000Hz and very low weight to low frequency signals. Wind noise contains a lot of low-frequency content, as will be shown in chapter

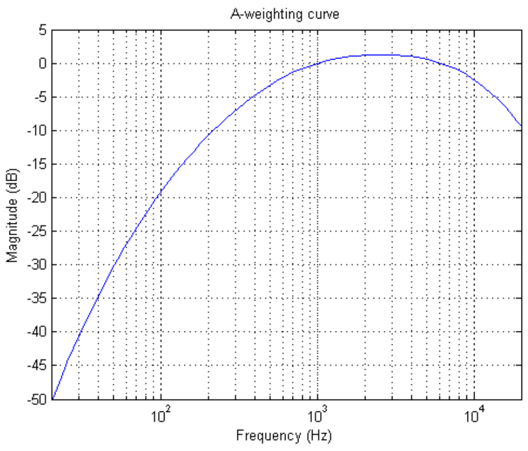


Figure 5.1: *A-weighting filter as a function of frequency.*

6, which can be trivially removed from speech using a highpass filter. Using A-weighted measures to evaluate the output of a noise reduction method, makes it possible to put more emphasis on the middle frequency range where most of the speech content exists. To A-weight the SNR measure, a frequency representation of the energy must be obtained. This can be done by estimating the power spectral density using the Welch method [35]. This method divides the signal up into small overlapping frames, windows each frame, calculates the Periodogram and then averages over all frames. It can be shown that using this strategy rather than just calculating the Periodogram over the entire signal, significantly decreases the variance of the estimate.

An A-weighted SNR estimate is then given by:

$$SNR_{Aw} = 10 \log \frac{\sum_{\omega} P_s(\omega) \cdot Aw(\omega)}{\sum_{\omega} P_{w_o}(\omega) \cdot Aw(\omega)}$$

where $P_s(\omega)$ and $P_{w_o}(\omega)$ is the power spectral density estimate using the Welch method of the speech and output noise respectively.

A similar segmented SNR estimate can be obtained:

$$SNR_{Aw,seg} = \frac{1}{M}\sum_{m=1}^{M} 10\log\frac{\sum_{\omega} P_s(\omega,m)\cdot Aw(\omega)}{\sum_{\omega} P_{w_o}(\omega,m)\cdot Aw(\omega)}$$

The NR measure can be extended in a similar way.

5.1.2 Binary Masking

Based on a speech and noise spectrogram, a binary mask [18] can be calculated. This binary mask consists of a time-frequency matrix of the same size as the speech and noise spectrogram. The mask contains only 0's and 1's, where a time-frequency bin is 0 if the noise spectrogram is larger than the speech for that bin and otherwise 1. By multiplying the binary mask with the noisy spectrogram, a speech estimate is obtained. If the noise and speech spectrogram used to calculate the binary mask is the original spectrograms used to create the noisy signal, then the binary mask is called the "Ideal Binary Mask" and the resulting speech estimate represents an optimal separation that can be compared to other methods.

5.2 Subjective Evaluation

Using test subjects to evaluate the sound output is the best way to grade noise reduction algorithms that should be used by real people, but it is also the most time consuming. Many different subjective tests exist and in the following, a number of these will be briefly reviewed. As the MUSHRA test is the most relevant for this thesis, it will be reviewed in greater detail.

5.2.1 Preference Test

In the preference test the test subject is presented to two different sounds and the test subject must then choose which one has the most pleasing overall sound quality. The test subject can only choose one of the two sounds or select no preference. This is a relatively simple test that is well suited for comparing two alternatives.

5.2.2 ITU-T Recommendation P.800

MOS [17], *Mean Opinion Score*, provides a numerical indication of the perceived quality of a sound after it has been processed in some way and is specified in the ITU-T recommendation P.800. MOS is expressed as a number between 1.0 and 5.0, where 5.0 corresponds to the best quality. A disadvantage of this methodology is that there is no build-in scaling, so one test subject might grade sounds with minor artifacts as having very bad quality, while another test subject might grade it as having good quality. This means that a lot of test subjects are needed to achieve a statistically significant result.

5.2.3 ITU-R Recommendation BS.1116-1

Methods for the Subjective Assessment of Small Impairments in Audio Systems Including Multichannel Sound Systems [15] uses a "double-blind triple-stimulus with hidden reference" testing method. The test subject has three signals available (A,B,C). A known reference signal (the clean speech signal) is always available as stimulus A. A hidden reference and the sound to be graded are simultaneously available but are "randomly" assigned to B and C. The subject is asked to assess the impairments on B compared to A, and C compared to A, according to the continuous five grade impairment scale that is also used in the MOS test. This makes it possible to determine very small impairments in the degraded signal, but it is not very usable for comparing different processing techniques because only one degraded signal is available at a time. Also because of the small degradation in the sound, expert listeners who are used to listening to sound in a critical way is needed.

5.2.4 ITU-R Recommendation BS.1534-1

MUSHRA [14], *Multiple Stimuli with Hidden Reference and Anchor*, is a methodology to evaluate the perceived quality of the output from lossy audio compression algorithms. It is defined by ITU-R recommendation BS.1534-1 and is intended for medium and large impairments in the sound.

It is a "double-blind multi-stimulus" test method with hidden references and anchors. In one test trial there are a number of test sounds and a reference sound that contains the original clean speech signal. The test subject can listen to and compare each sound in any order, any number of times. The test subject is required to score the test samples according to how pleasing the sound is in relation to the reference. The grading scale is based on a continuous quality scale of integers from 0 to 100, which is divided into five equal categories as shown in figure 5.2. The test subject should not necessarily give a grade in

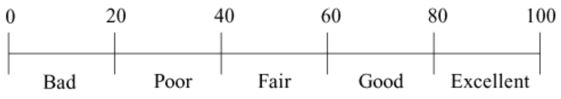


Figure 5.2: *Grading scale for MUSHRA.*

the "bad" category to the sound with the lowest quality in the test, but one or more sounds must be given a grade of 100 because the given reference signal is also hidden as one of the test sounds to be graded. A number of anchors are also hidden among the test signals to be graded. According to the standard one of these must be a 3.5kHz lowpass filtered version of the reference signal. Any other anchors are optional, but they should contain artifacts similar to the other test signals.

The test subjects should have some experience in listening critically to sound, although this is not as important as in ITU-R Recommendation BS.1116-1, because the impairments are expected to be relatively high and therefore easy to detect. Before the test is started, it is mandatory to expose the test subjects to the full range and nature of the impairments and all the test signals that will be experiences during the test. The listening level can be adjusted before a test run to within $\pm$4dB relative to the reference level defined in ITU-R Recommendation BS.1116-1, but should be kept constant during a test run. Also only the use of either headphones or speakers are allowed.

An advantage of MUSHRA is that the use of a hidden reference and anchors avoids the problems with MOS, because they provide an absolute scale that the test signals can be compared to. Also because the different test sounds are available at the same time, the test subjects can better judge the difference between them and a comparison is easier to make. This also makes it possible to perform a finer distinction between the test sounds, which is reflected in the 0 to 100 quality scale, as opposed to the 1.0 to 5.0 scale used in MOS.

5.2.4.1 Statistical Analysis

The mean of a test sound can be found as the average over all test subjects:

$$\overline{\mu}_j = \frac{1}{N}\sum_{i=1}^{N}\mu_{i,j}$$

where N is the number of test subjects and $\mu_{i,j}$ is the score for test subject i and algorithm j.

A 95% confidence interval for the mean can be found as:

$$[\mu_j - \delta_j, \mu_j + \delta_j]$$

where

$$\delta_j = 1.96\sqrt{\frac{\sum_{i=1}^{N}(\overline{\mu}_j - \mu_{i,j})^2}{N(N-1)}}$$

For multiple test runs with different reference sounds, a similar mean and confidence interval can be found over all test runs.

5.3 Objective Perceptual Evaluation

Subjective test evaluations are very time consuming and can therefore not be used during the optimization of a noise reduction algorithm. For this purpose some objective evaluation is needed that gives predictions that are close to those of a subjective listening test. In [43] it is shown that PESQ is an appropriate measure for the development and optimization of noise reduction schemes and has a high correlation with subjective listening tests.

5.3.1 ITU-T Recommendation P.862

PESQ [42] [16], *Perceptual Evaluation of Speech Quality*[1], is an objective method for automated assessment of the speech quality in a narrow-band (about 300-3400Hz) telephone system or speech codec. It is standardized by ITU-T in recommendation P.862 (02/01). A wideband extension (about 50-7000Hz) has been added in recommendation P.862.2.

The PESQ measure compares an original clean speech signal $s(n)$ with a degraded signal $\hat{s}(n)$ and outputs a prediction of the perceived quality that would be given to $\hat{s}(n)$ by subjects in a subjective listening test. In the context of noise reduction schemes, $\hat{s}(n)$ would be the output of the noise reduction algorithm. The ITU-T reference mentions that in 22 benchmark tests the correlation between the PESQ score and a listening test with live subjects was 0.935. The output of the PESQ algorithm is given as a MOS-like score.

[1]A reference implementation of the PESQ algorithm can be downloaded from http://www.itu.int/rec/T-REC-P.862/en

In the following a brief overview of the calculation of the PESQ score will be given. PESQ basically compares $s(n)$ and $\hat{s}(n)$ using a perceptual model. First individual utterances in the signals are delay-compensated to make sure the two signals are properly aligned. Then they are level aligned corresponding to 79dB SPL at the ear reference point. The STFT is then taken and the frequency representation is mapped into a psychophysical representation. This representation consists of a time-frequency mapping into the perceptual Bark frequency and Sone loudness level followed by a frequency warping and a compressive loudness scaling. Based on this, different Disturbance values are calculated and a linear combination of these makes up the final PESQ score. The details of this calculation can be found in [16] and its references. The usual range of the PESQ score is from 1.0 to 4.5, but in extreme cases it can go down to 0.5.

5.3.1.1 Choice of Input Sound

The PESQ standard recommends that natural speech recordings are used when evaluation is performed, although artificial speech signals and concatenated real speech signals can be used, but only if they represent the temporal structure (including silent intervals) and phonetic structure of real speech signals. Test signals should be representative of both male and female speakers and for natural speech recordings it is recommended that at least two male talkers and two female talkers are used for each testing condition. They should include speech bursts of about 1 to 3 seconds separated by silence and speech should be active for about 40 to 80 percent of the time. Most of the signal used in calibrating and validating PESQ contained pairs of sentences separated by silence, totalling about 8 seconds in duration.

5.3.1.2 Comparison between objective and subjective scores

How well the PESQ measure predicts the subjective scores may be measured by calculating the correlation coefficient between PESQ measure results and subjective listening test results from the same data set. The correlation coefficient is calculated using Pearson's formula:

$$r = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2 \sum_j (y_j - \overline{y})^2}}$$

where x_i, y_i, $\overline{x}$ and $\overline{y}$ is the result from the PESQ measure and listening test on the ith datapoint and the averages of these respectively.

CHAPTER 6

Experimental Data

This section contains information about the sound data used to train, optimize and evaluate the models. The first part contains a description of how the sound data is obtained followed a description of the different data sets used in the thesis. All datasets are disjoint.

6.1 Obtaining the Data

Different datasets are used for training, optimizing, and evaluating the models. For the training part individual speech and wind noise sound files are used. For optimizing and evaluating the methods speech and noise is mixed together. Ideally, the mixed sound files should be recorded by a microphone while both speech and wind noise is present. The problem with this approach is that the clean speech and noise signals are not known and so it is not possible to objectively measure how much the method improves the speech. Instead the wind noise and speech will be obtained separately and mixed together on a computer for the optimization and some of the evaluation. Some noisy speech signals that has been recorded while wind noise is present will, however, also be used as a part of the subjective evaluation.

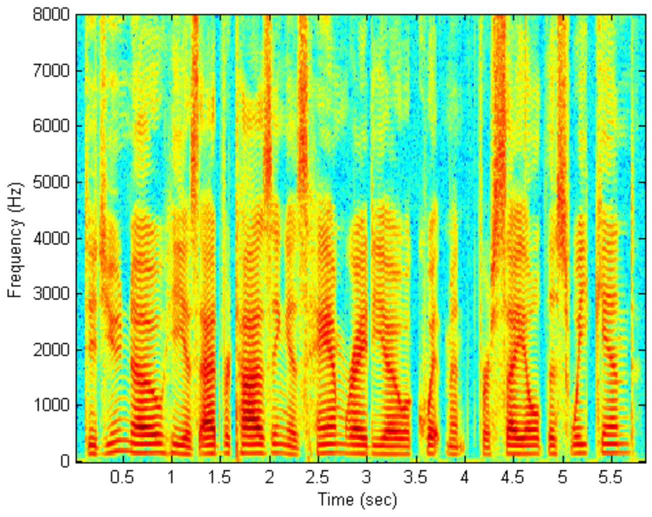


Figure 6.1: *Spectrogram of speech from the TIMIT database. The amplitude is measured in dB.*

The speech data used for this thesis consists of speech samples from the TIMIT-database. In total 60 male and 60 female randomly chosen speech sentences has been extracted. Each sentence has a duration of 2-4 seconds corresponding to about 6 minutes of speech. The speech data is available as 16bit signals sampled at 16kHz. A typical speech recording can be seen in figure 6.1. Speech has most of its energy in the frequency range from 125-4000Hz. The wind data is recorded around the Copenhagen area using a single hearing aid-type microphone. The wind is recorded while holding the microphone at different angles relative to the wind and is sampled at 16kHz. The wind speed is measured while recording using a handheld anemometer. A spectrogram of a wind noise sample recorded at 5-10m/s wind speed is seen in figure 6.2. Wind noise has a lot of low frequency content with a few spikes higher op in the frequency spectrum.

6.2 Dataset 1

This dataset is used for training the codebooks in the Non-Stationary Spectral Subtraction Algorithm and the Non-Negative Matrix Factorization algorithm. It consists of 5 male and 5 female sentences from the TIMIT database and 10 wind noise recordings of about 5 seconds each. The recorded wind velocity is

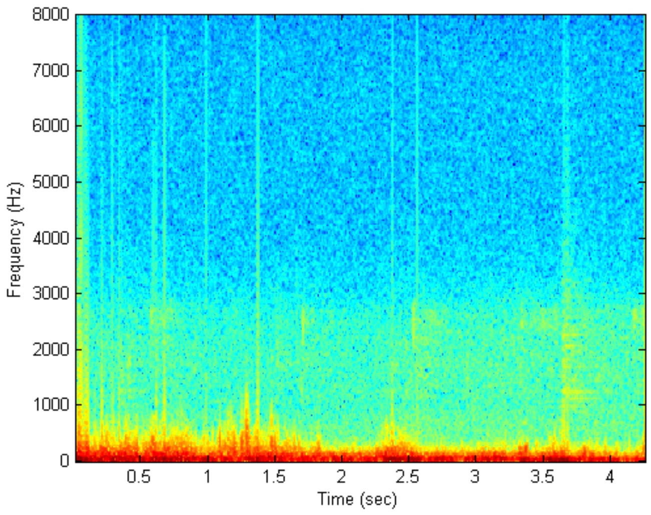


Figure 6.2: *Spectrogram of a 5-10m/s wind noise sample in dB.*

about 5-10m/s with a few wind gusts of up to 15 m/s.

6.3 Dataset 2

This dataset is used for optimizing the methods to filter wind noise from speech using the PESQ measure. It consists of 5 male and 5 female sentences from the TIMIT database and 10 wind noise recordings of about 5 seconds each. The 10 speech files are mixed with the 10 wind noise files at 0dB A-weighted SNR to produce 10 noisy speech signals. They are mixed so there is one second of clean wind noise in the beginning of each file. The recorded wind velocity is about 5-10m/s with a few wind gusts of up to 15 m/s.

6.4 Dataset 3

This dataset is used for evaluating the methods using the objective measures including PESQ. It consists of 44 male and 44 female sentences taken from the TIMIT database and 44 wind noise recordings of about 5 seconds each. 22 male and 22 female sentences are mixed with the 44 wind noise recordings at 0dB A-weighted SNR. 0dB A-weighted SNR corresponds to about -20dB normal SNR and while this number seems quite small, most of the energy is concentrated in the lowest frequency range, where it does not overlap with the speech. The remaining sentences are mixed with the same 44 wind noise recordings but at -11dB A-weighted SNR. This results in 44 noisy speech signals mixed at 0dB A-weighted SNR and 44 noisy speech signals mixed at -11dB A-weighted SNR. The recorded wind velocity is about 5-10m/s with a few wind gusts of up to 15 m/s.

6.5 Dataset 4

This dataset is used for evaluating the methods using the subjective MUSHRA test. It consists of 3 male and 3 female sentences taken from the TIMIT database. Each sentence is played back on an ADAM A7 professional studio monitor at a moderate volume, and recorded by a microphone while wind noise is present. This results in 6 speech recordings with wind noise. Because the speech and wind is not mixed in a computer, the clean wind noise is not known. The original speech sentence from the TIMIT database is used a an estimate of the clean speech sentence. The recorded wind velocity is about 5-10m/s with a few wind gusts of up to 15m/s. A spectrogram of one of the recordings can be seen in figure 6.3.

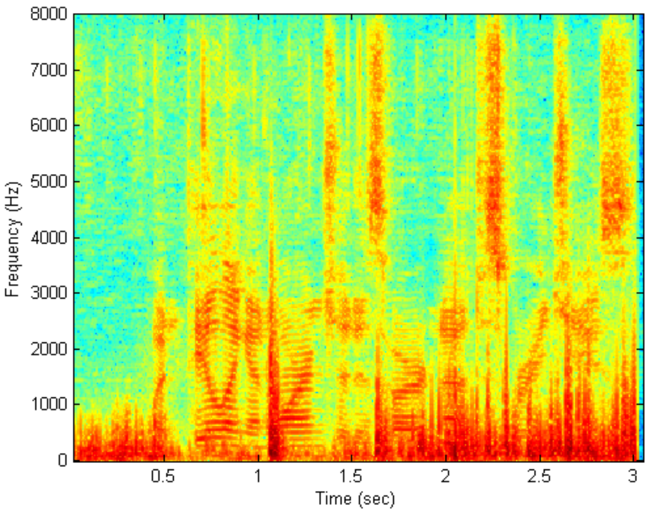


Figure 6.3: *Spectrogram of a recorded speech sample during wind noise in dB.*

6.6 Dataset 5

This dataset is used for evaluating the Non-Negative Matrix Factorization algorithm in heavy wind conditions. It consists of 3 male and 3 female sentences from the TIMIT database and is processed in the same way as dataset 4, but with a wind velocity of about 10-20m/s.

6.7 Dataset 6

This dataset is used for training the codebook in the Non-Negative Matrix Factorization algorithm for heavy wind conditions (10-20m/s). It consists of 10 wind noise recordings of about 5 seconds each. The recorded wind velocity is about 10-20m/s.

Chapter 7

Experimental Analysis

In this chapter the different methods reviewed so far, will be optimized to reduce wind noise from mono speech signals. First a general optimization scheme is described followed by an optimization procedure that uses the PESQ measure given in section 5.3.1.

All frequency domain analysis in this chapter is using 32ms Hanning windows with 75% overlap between each window and the wideband extension of the PESQ measure has been used for calculating the PESQ values.

7.1 General Optimization Scheme

The three different noise reduction algorithms reviewed in this thesis will be optimized to filter wind noise from speech. Because this is done in a similar way for all algorithms, the approach will be described here.

For each parameter in the algorithm that is being optimized, good initial values are found by listening to the output of the algorithm when run on the noisy speech in dataset 2. Based on this, a meaningful range of values is chosen. In this interval the PESQ measure from section 5.3.1 is evaluated on the output of the algorithm from the same files, and a maximum of the PESQ measure is found by manually adjusting the parameters. This is done in the following way:

1. For some given parameters, the noisy speech signals from dataset 2 are given as input to the algorithm.
2. The ten output files of about 4-5s. each are concatenated two and two, to form five 8-10s. sound files. This is done in order to obtain files of the same length and type as the PESQ method has been verified on.
3. The PESQ measure is calculated on the five concatenated files. This will give different values than calculating the PESQ measure on each of the non-concatenated files and then averaging the result, because of the nonlinear averaging process in the PESQ method.
4. The mean of the the PESQ results from the five concatenated sound files are found.
5. This procedure is repeated for some other given parameters until a maximum mean PESQ values has been found.

In each parameter interval, at least 5 values are selected and for each possible combination of parameters, the PESQ measure is calculated. The parameters corresponding to the largest PESQ value is then used as a starting point for further investigation. Next the parameters are varied one by one within the interval while keeping the other values fixed to the values found previously. In this way, the sensitivity and importance of each individual parameter can be evaluated. The results from the optimization procedure of each method will be presented in the following sections.

7.2 Optimization of Generalized Spectral Subtraction

This section contains the optimization procedure for the Generalized Spectral Subtraction algorithm reviewed in chapter 2.

The wind noise estimate for this method is obtained as an averaging window over the initial silence in each sound file. The length of the window used was 225ms long, but it was found that the exact size of the window was not critical. A relevant range of parameter values to optimize the model over is found to be:

- $\lambda_H \in [0.00, 0.875]$, The amount of time smoothing
- $L \in [0, 20]$, The sliding-window size of the frequency smoother
- $\gamma \in [0.25, 2.0]$, The exponent to the spectrogram
- $\rho \in [0.25, 2.5]$, The exponent to the filter H
- $\beta \in [0.0, 0.20]$, The lower bound on the filter H
- $\alpha_{max} \in [1.25, 4.75]$, The maximum overestimation factor
- $\beta_{filter} \in [0.0, 0.10]$, The lower bound on the speech estimate

7.2.1 Importance of Individual Parameters

The parameters are varied one by one within the interval while keeping the other values fixed, and selected values for each parameter can be seen in figure 7.1-7.7. The blue circle is the mean, the red line is the median, the limit of the solid blue box corresponds to the lower and higher quartile, and the dotted line signifies the range of the data. A point is considered an outlier (red cross) if it is more than $1.5 \cdot IQR$ away from the upper or lower quartile, where IQR is the difference between the lower and upper quartile. Because the PESQ measure gets 8-10s. files as input, there is a lot of averaging going on inside the PESQ algorithm that is not captured in the boxplot. Therefore the boxplot is not an accurate indication of the variance of the data and can only be used to see the difference between the five concatenated files.

7.2.1.1 Optimal Parameters

Overall it is found that a lot of the parameters does not have a very large influence on the quality of the filtering. The most significant parameters are seen to be an exponentiation of the spectrogram and a high degree of smoothing of the filter. When removing the weighted smoother $H_{s2}(\omega_i, m)$ from section 2.2.2, the PESQ score slightly decreased although it was hard to hear a difference between them. The choice of parameters that maximizes the PESQ measure for the Generalized Spectral Subtraction method is:

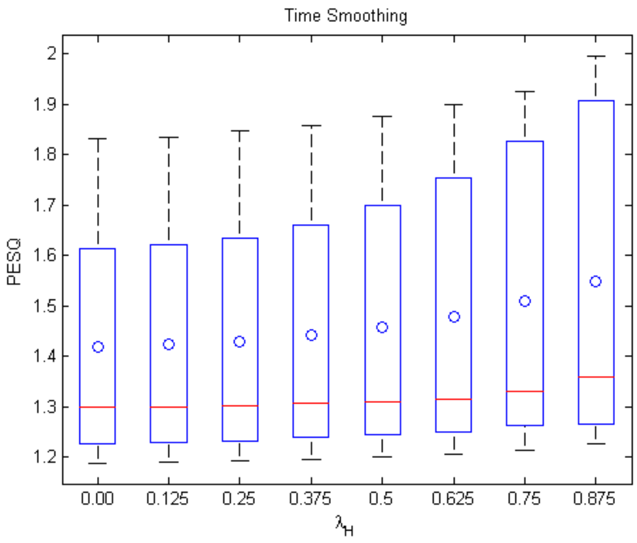


Figure 7.1: *The amount of time smoothing as a function of PESQ. More smoothing yields higher PESQ values.*

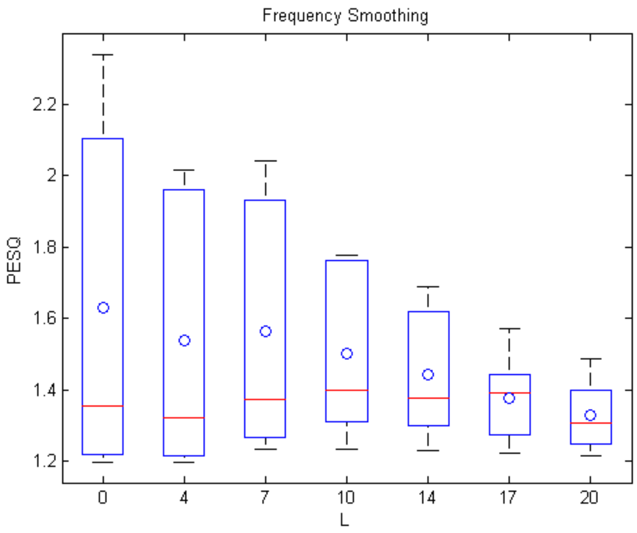


Figure 7.2: *The sliding-window size of the frequency smoother as a function of PESQ. The highest mean is found at L=0, but the difference between the concatenated files is lower at L=7 while having almost the same mean.*

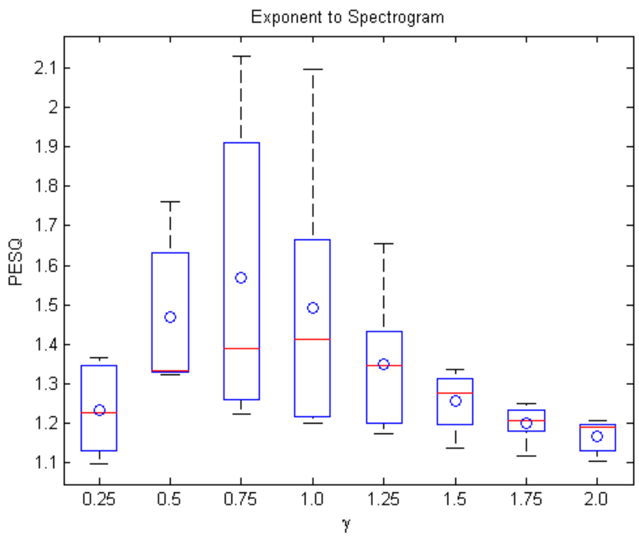


Figure 7.3: *The exponent to the Spectrogram as a function of PESQ. The method is very sensitive to the this parameter and values around* $\gamma = 0.75$ *gives the highest PESQ values.*

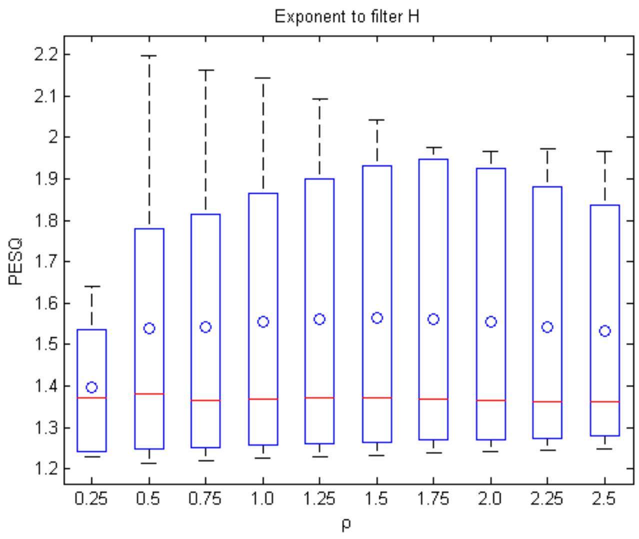


Figure 7.4: *The exponent to the filter H as a function of PESQ. The method is not particular sensitive to this parameter as long as it is higher than 0.5. A good value seems to be around the area of 1.5.*

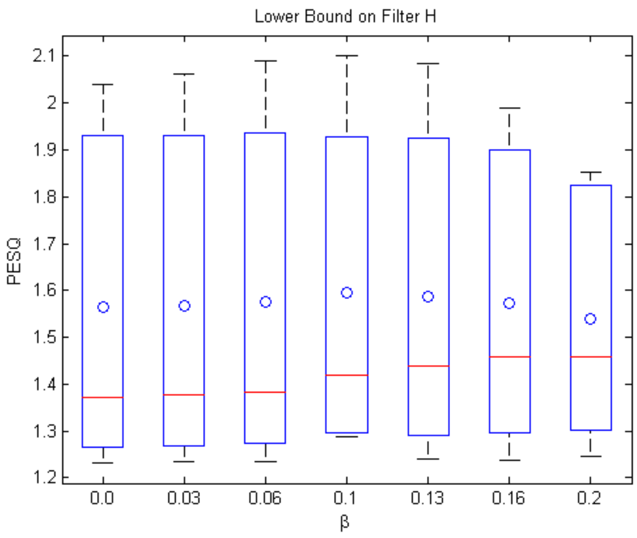


Figure 7.5: *The lower bound on the filter H as a function of PESQ. The method is not particularly sensitive to this parameter as long as it is kept below 0.16.*

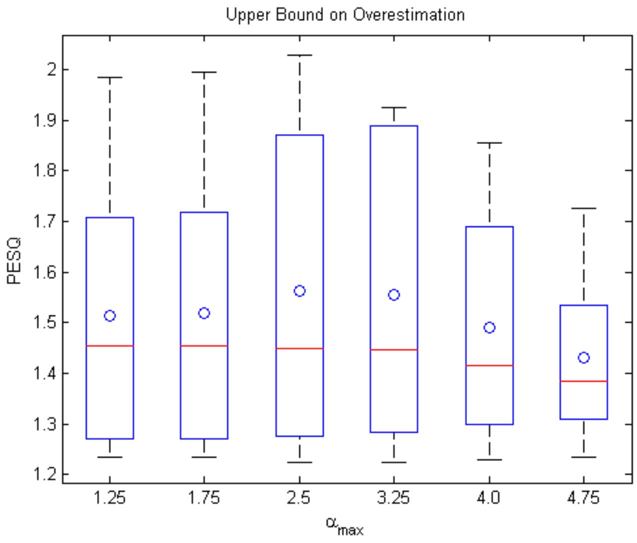


Figure 7.6: *The upper bound on the overestimation factor as a function of PESQ. A maximum is seen around 2.5-3.25.*

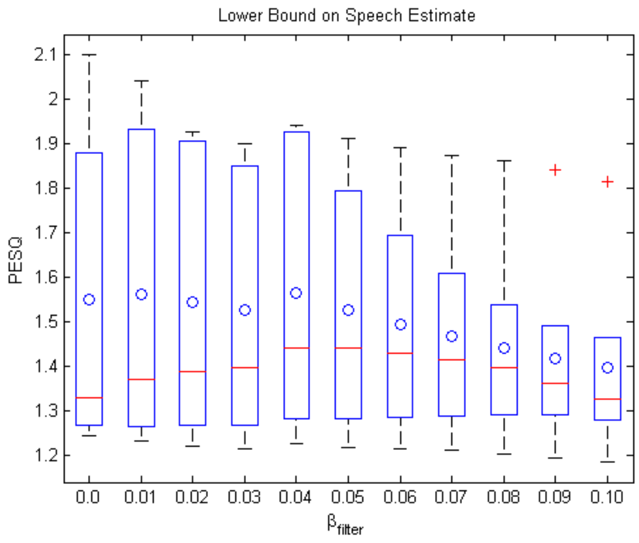


Figure 7.7: *The lower bound on the speech estimate as a function of PESQ. The value should not be set higher than 0.4.*

Parameter	Description
$\lambda_H = 0.9$	The smoothing factor seems important and higher degrees of smoothing yields better noise reduction
$L = 0$	Smoothing in the frequency domain does not give a better result
$\gamma = 0.7$	This parameter is highly significant and has a maximum around 0.7. The Power Spectral Subtraction assumption ($\gamma = 2$) is seen to be a bad choice of parameter. 0.7 is very close to the proposed relation in Steven's Power Law, between a sounds intensity and its perceived strength
$\rho = 1.5$	This parameter is not very significant, but a maximum can be found around 1.5
$\beta = 0.1$	This parameter is also not very significant, but a small maximum is found around 0.1
$\alpha_{max} = 3.125$	Other parameters could have been varied for the overestimation factor, but for simplicity it was chosen to use the formula given in [9], which can be seen in figure 2.2. Only the upper bound on the overestimation is varied and the optimal value is found at 3.125
$\beta_{filter} = 0.4$	This value is not very significant as long as it is kept at or below 0.4

This choice of parameters, give an average PESQ value of 1.54. By listening to the filtered noisy speech signals, it is informally verified that this choice of parameters indeed does gives significant noise reduction, although there is still intervals of several seconds where wind noise can clearly be heard.

7.3 Optimization of Non-Stationary Spectral Subtraction

This section contains the optimization procedure for the Non-Stationary Spectral Subtraction algorithm reviewed in chapter 3.

The noise estimation is performed by using codebooks and searching through them intelligently in each timeframe, the way it is described in section 3.4.1. Dataset 1 is used for calculating the codebooks. The initial noise estimate that is needed for this approach, is obtained by running the k-means clustering algorithm from section 3.4 on the noise codebook with only one clustercenter. This clustercenter, that can be precomputed, is then used as the initial noise estimate. Furthermore, to keep the computation time of the algorithm reasonable, it has been decided to limit the codebooks to 20 spectrums for each codebook.

The initial range of values is found to be:

- $N \in [64, 4092]$, The number of points used in the integration
- $AR_y \in [1, 25]$, The number of AR coefficients used to model the noisy speech signal y
- $K_w \in [10, 20]$, The number of spectrums in the noise codebook has been limited due to computation time
- $AR_w \in [5, 40]$. The number of AR coefficients in the noise codebook
- $K_s \in [10, 20]$, The number of spectrums in the speech codebook has been limited due to computation time
- $AR_s \in [5, 40]$. The number of AR coefficients in the speech codebook

It might also improve the filtering to exponentiate the noisy speech signal and the codebooks before the algorithm is run, but this has not been done here.

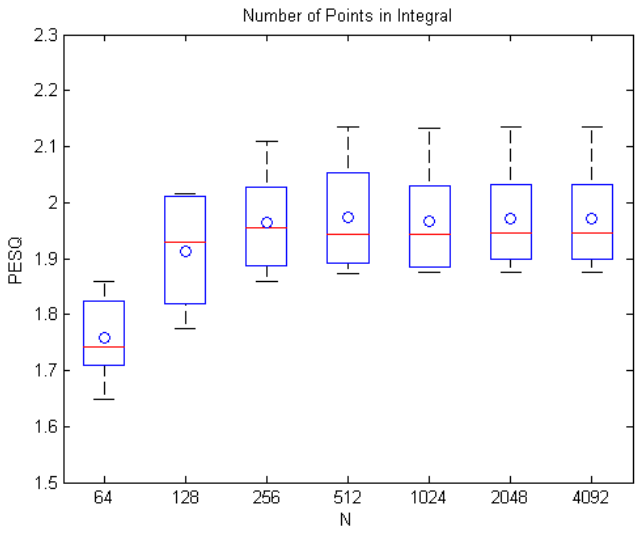


Figure 7.8: *The number of points used in the integration as a function of PESQ. N should at least be 512.*

7.3.1 Importance of Individual Parameters

The parameters are varied one by one within the interval while keeping the other values fixed and selected values for each parameter can be seen in figure 7.8-7.11.

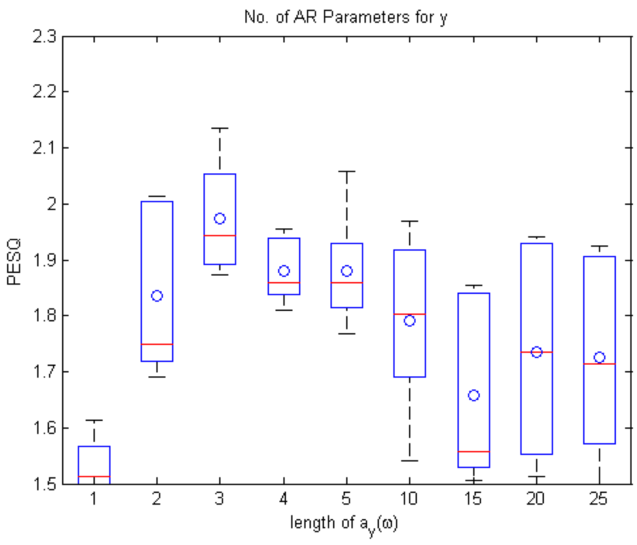


Figure 7.9: *The number of AR coefficients used to model y as a function of PESQ. This is an important parameter. The maximum is found at 3.*

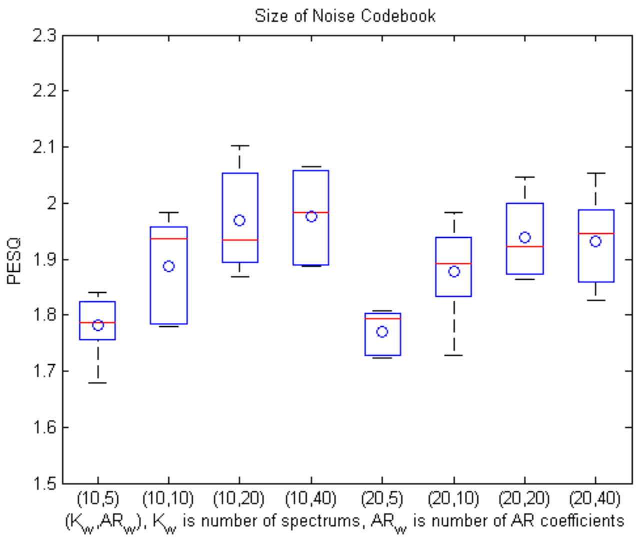


Figure 7.10: *The size of the noise codebook as a function of PESQ. There is no advantage to using 20 spectrums instead of 10 and the number of AR coefficients should be 20 or 40.*

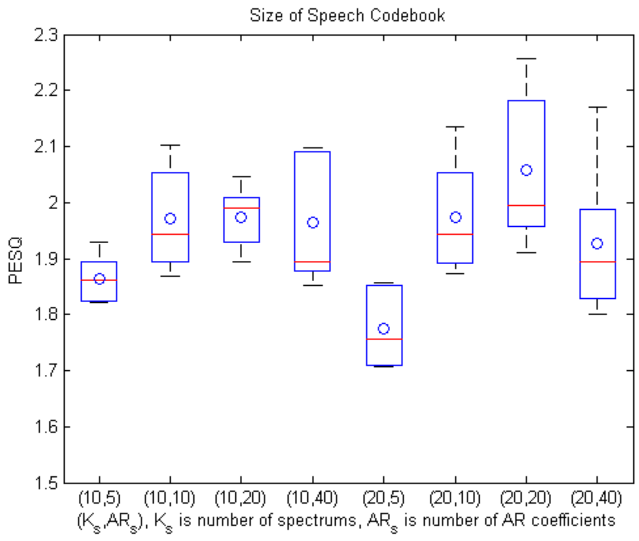


Figure 7.11: *The size of the speech codebook as a function of PESQ. 20 AR coefficients should be used and an increase in PESQ by about 0.1 can be achieved by increasing the number of spectrums from 10 to 20.*

7.3.1.1 Optimal Parameters

All parameters are seen to have a significant influence on the Non-Stationary Spectral Subtraction algorithm and a good combination of parameters is found to be:

Parameter	Description
$N = 512$	The number of points used in the integration should be at least 512, but there is no advantage to using more points
$AR_y = 3$	The number of coefficients used to model the noisy speech signal y should be 3
$K_w = 10$	The number of spectrums in noise codebook should be set to 10. It should not be increased to 20 spectrums, as the algorithm actually performs worse then. This could be due to overfitting
$AR_w = 20$	The number of AR coefficients used in noise codebook can be set to 20 or a bit higher to gain a marginal increase in PESQ. To keep the computational complexity as low as possible it is set to 20 here
$K_s = 20$	The number of spectrums in speech codebook has been limited to 20, due to the computational complexity
$AR_s = 20$	The number of AR coefficients used in speech codebook should be set to 20

This choice of parameters gives an average PESQ value of 2.04. By listening to the filtered noisy speech signals, it is informally verified that this choice of parameters indeed does yield good noise reduction without musical noise.

7.3.1.2 Non-Stationary Generalized Spectral Subtraction

To further improve the filtering, the noise and speech magnitude estimates found by the Non-Stationary Spectral Subtraction algorithm, can be used in the Generalized Spectral Subtraction filter:

$$H(\omega) = \max\left(1 - \alpha(\omega_i, m) \cdot \frac{|\hat{W}_{AR}(\omega)|^{\gamma}}{(|\hat{W}_{AR}(\omega)| + |\hat{S}_{AR}(\omega)|)^{\gamma}}, \beta\right)^{\rho}$$

where $|\hat{W}_{AR}(\omega)|$ and $|\hat{S}_{AR}(\omega)|$ is the smoothed noise and speech magnitude estimates found from the codebooks in the Non-Stationary Spectral Subtraction method respectively. It is important here to distinguish between $|\hat{S}(\omega)|$

and $|\hat{S}_{AR}(\omega)|$, where the former is the speech estimate output and the latter is the smoothed speech estimate from the codebook that is used in the filtering operation in equation 3.1. This filter is then used in the generalized Spectral Subtraction algorithm as detailed in section 2.2.4.

The optimal parameters for the Non-Stationary Spectral Subtraction algorithm are used to find the noise and speech spectrums in the codebooks. For the Generalized Spectral Subtraction algorithm, the optimization procedure is repeated as in section 7.2 and the parameters that improve the filtering are:

Parameter	Description
$\lambda_H = 0.9$	The smoothing factor is an important parameter and higher degrees of smoothing yields better noise reduction
$\gamma = 0.7$	This parameter is highly significant and should be set to 0.7
$\rho = 0.8$	That this value is below 1, means that it helps to increase the filter values in H. This indicates that the Non-Stationary Spectral Subtraction algorithm has a tendency to overestimate the noise
$\beta = 0.005$	This parameter is not very significant, but a small maximum is found around 0.005
$\beta_{filter} = 0.01$	This value is not very significant, but a small maximum is found around 0.01

This gives an average PESQ value of 2.46, which is an improvement of 0.92 and 0.44 when compared to the two other methods respectively. When listening to the filtered speech estimate, the speech estimate is slightly better than without the Generalized Spectral Subtraction method, but it still has the same overall sound.

7.4 Optimization of Non-Negative Matrix Factorization

This section contains the optimization procedure for the Non-Negative Matrix Factorization algorithm reviewed in chapter 4.

The wind noise files in Dataset 1 are used for precomputing the noise dictionary as explained in section 4.6 and the parameters of the model are optimized using Dataset 2 with the procedure from section 7.2. The factorization is calculated over the entire magnitude spectrogram of a sound file. Because $\boldsymbol{D}$ and $\boldsymbol{C}$ are initialized as random positive matrices, the gradient descent optimization scheme will produce different factorizations, each time it is run. To investigate how different these factorizations are, the algorithm is run 20 times on the same data file with the same parameters. The PESQ value is then calculated for each file. The variance of the PESQ value between the 20 files was 0.062 and the largest deviation from the mean was 0.130. Audibly there was a very small or no difference between the files, and throughout this optimization scheme it is assumed that the NNMF method produces reliable results in just a single run. The maximum number of iterations is set to 500 and the convergence criteria to 10^{-4}.

The initial range of values is found to be:

- $\gamma \in [0.4, 1.5]$, The exponentiation of the magnitude spectrogram
- $\lambda_s \in [0, .5]$, The sparsity parameter for the speech
- $\lambda_w \in [0, 0.1]$, The sparsity parameter for the noise
- $\lambda_{pre} \in [0, 0.3]$. The sparsity parameter for precomputing the noise codebook
- $ds \in [16, 128]$, The number speech basis vectors in D
- $dw \in [16, 128]$. The number of precomputed noise basis vectors in D

Also the costfunction should be evaluated.

7.4.1 Importance of Individual Parameters

The parameters are varied one by one within the interval while keeping the other values fixed and selected values for each parameter can be seen in figure 7.12-7.17.

7.4.1.1 Optimal Parameters

The parameters that maximizes the PESQ measure are:

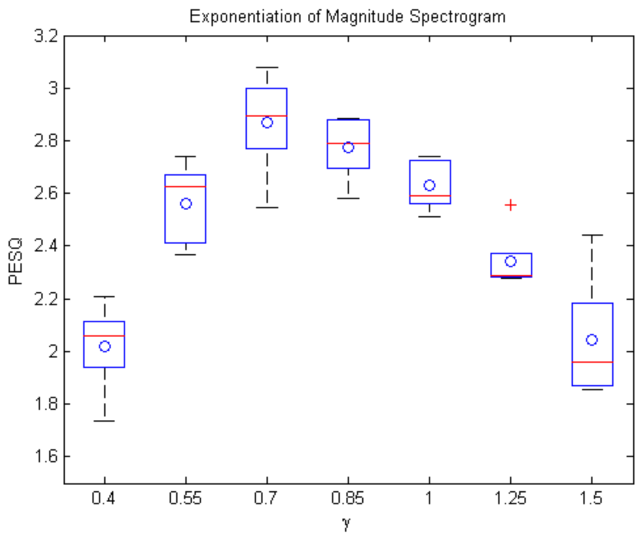


Figure 7.12: *The exponentiation of the magnitude spectrogram as a function of PESQ. This parameter is highly significant and a maximum is seen at 0.7.*

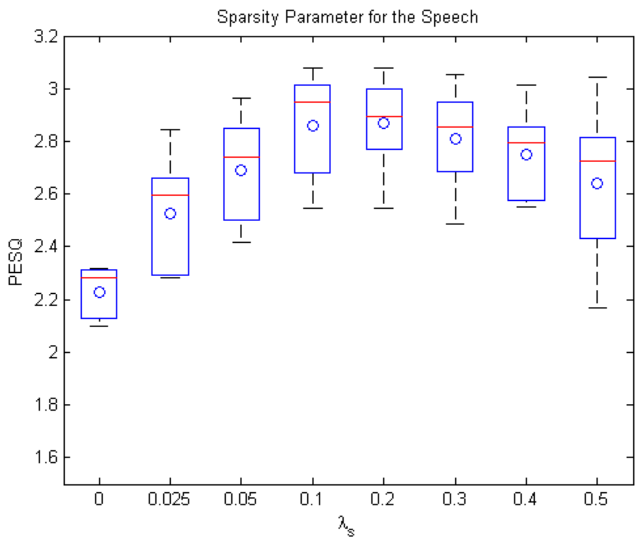


Figure 7.13: *The sparsity parameter for the speech as a function of PESQ. This parameter has a maximum around 0.2.*

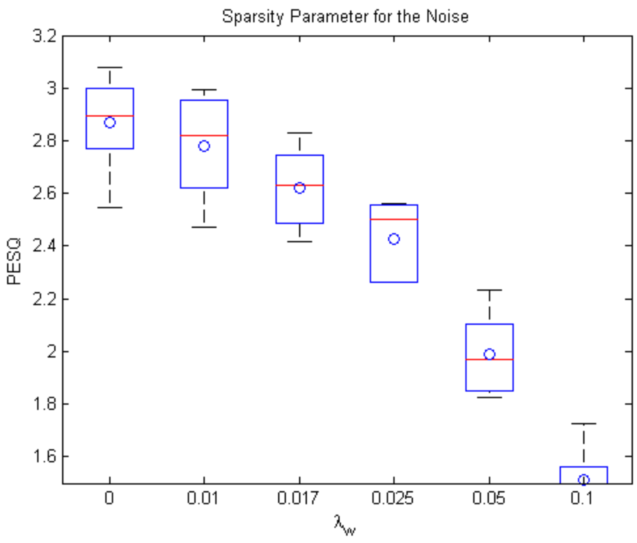


Figure 7.14: *The sparsity parameter for the noise as a function of PESQ. It is best not to use any sparsity for the noise.*

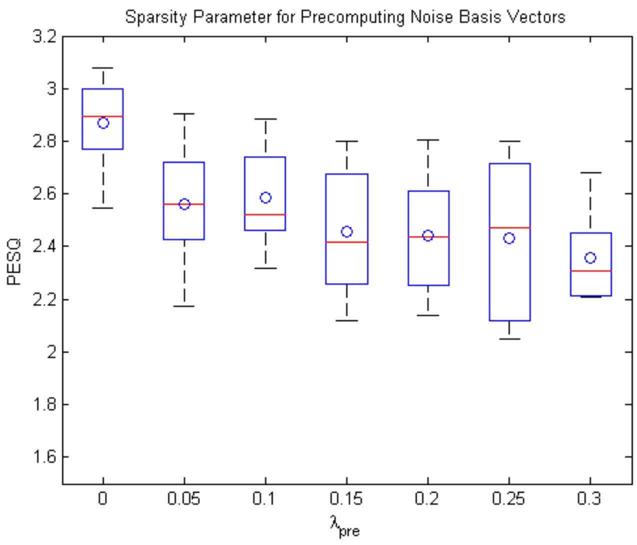


Figure 7.15: *The sparsity parameter for precomputing the noise dictionary as a function of PESQ. It is best not to use any sparsity for precomputing the noise codebook.*

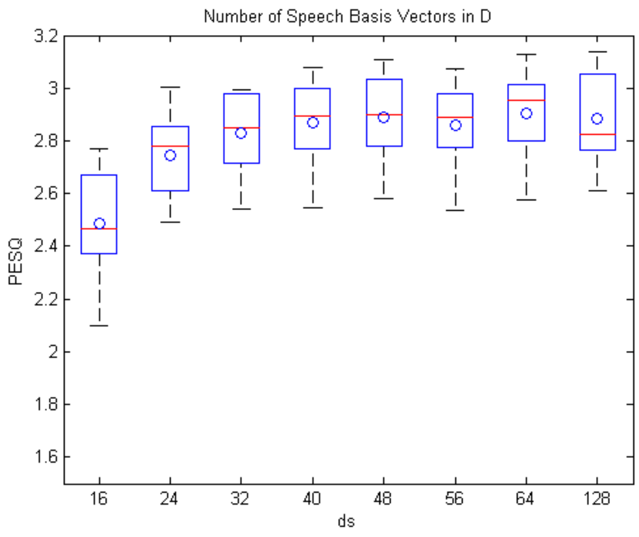


Figure 7.16: *The number of speech basis vectors as a function of PESQ. At least 40 speech basis vectors should be used.*

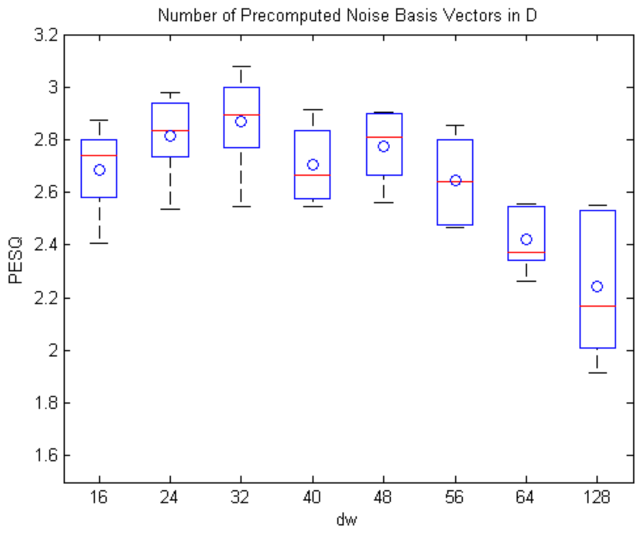


Figure 7.17: *The number of precomputed basisvectors in D as a function of PESQ. A maximum is seen at 32 basis vectors.*

Parameter	Description
$\gamma = 0.7$	The exponentiation of the magnitude spectrogram is a highly significant parameter and should be set to 0.7, which equals Steven's Power Law
$\lambda_s = 0.2$	The sparsity parameter for the speech has a maximum at 0.2
$\lambda_w = 0$	The sparsity parameter for the noise should be set to zero
$\lambda_{pre} = 0$	The sparsity parameter for precomputing the noise codebook should be set to zero
$ds = 40$	The number speech basis vectors in D should at least be set to 40. More basis vectors increases computation time
$dw = 32$	The number of precomputed noise basis vectors in D should be set to 32. Too many noise basis vectors, actually decreases performance
$CC =$ Kullbach Leibler	The minimization of the Kullbach Leibler costfunction has the best performance.

Generally, it appears that sparsity should be used for speech, but not for wind noise and that an exponentiation equal to Steven's Power Law gives good results. The Least Squares and Kullbach Leibler costfunction results can be seen in table 7.1. The values given are the mean of the PESQ scores when run five times on Dataset 2. The Kullbach Leibler costfunction gives a significantly higher PESQ

Least Squares	Kullbach Leibler
2.43	2.87

Table 7.1: *The performance for the two costfunctions.*

score. The result for the Kullbach Leibler costfunction with and without A-weighting can be seen in table 7.2. This is calculated in a similar way. It does not improve the PESQ score to introduce A-weighting of the costfunction. This choice of parameters gives an average PESQ value of 2.87.

no weighting	A-weighted
2.87	2.72

Table 7.2: *The performance for the two A-weighted costfunction.*

7.4.1.2 Generalized Spectral Subtraction in NNMF

The noise part of the factorization can be used as an advanced noise estimator as mentioned in section 4.5. This noise estimation can then be used in the Generalized Spectral Subtraction Algorithm filter given in section 7.2. Using this procedure and repeating the optimization process, it is found that a smoothing of the filter by $\lambda_H = 0.7$ improves the average PESQ score from 2.87 to 3.14. None of the other parameters improves the filtering, although it should be mentioned that the exponentiation of the magnitude spectrogram already is implemented in the NNMF algorithm.

7.5 Computational Complexity

This section contains an investigation of the computational complexity of each method. The simulations of the computational complexity is done on a computer with an AMD Athlon 64 3700+ processor with 2.22GHz. The complexity is only evaluated based on how many seconds it takes to run the Matlab implementation on a sound file of 4 seconds and can not be used to evaluate the real-time performance. The algorithms are run on Dataset 2.

Because the Non-Negative Matrix Factorization algorithm is initialized with

Method	Computation Time (sec.)
NNMF	88
NNMF acc. Conv.	54
NSS	117
NSS int. CB	21
SPS	0.51

Table 7.3: *Computation time for each method in seconds. NNMF: Non-Negative Matrix Factorization, NNMF acc. Conv: NNMF with accelerated convergence (see section 4.3.2), NSS: Non-Stationary Spectral Subtraction, NSS int. CB: NSS with intelligent searching strategy for the codebooks (see section 3.4.1), SPS: Generalized Spectral Subtraction*

random matrices, the convergence time changes for each new factorization by a few seconds. Therefore, the method is started 60 times and an average over the computation time is found. For the two other methods, an average over 10 runs is calculated. For all methods, it is assumed that the magnitude spectrogram of the noisy signal is available before it is run. The parameters are set to those found in the optimization section for each method. The result can be seen in table 7.3. It is seen that the SPS method has a very low computation time. For the NNMF and NSS method, the original NNMF algorithm is faster than the NSS algorithm, but intelligently searching the codebooks in the NSS algorithm reduces the computation time to only 21 seconds. The accelerated convergence of the NNMF algorithm increase the computation time by 34 seconds to 54 seconds.

7.6 Learning Codebooks

During the optimization of the parameters, the entire Dataset 1 has been used to compute the codebooks for the Non-Stationary Spectral Subtraction and Non-Negative Matrix Factorization algorithm. When training the methods for new kinds of noise it is useful to know approximately how much data is needed to obtain good codebooks. If only small samples of data is needed it might even be possible to obtain the codebook on-the-fly when a new noise type is encountered. To investigate how the methods perform with small training sets, wind and speech data of varying length has been extracted from Dataset 1 and used to train the codebooks for the two methods. For the extracted datasets that are smaller than half of the original Dataset 1, the same size of data has been extracted several times from different places in the dataset. The codebooks are then trained on the data and used to filter wind noise from speech using Dataset 2. The PESQ results of this data analysis can be seen in figure 7.18 and 7.19. The NNMF curve seems to drop around 7.7 seconds while the NSS curve begins to fall between 11.5 and 23 seconds. For the rest of the thesis, the entire training set has been used for calculating the codebooks.

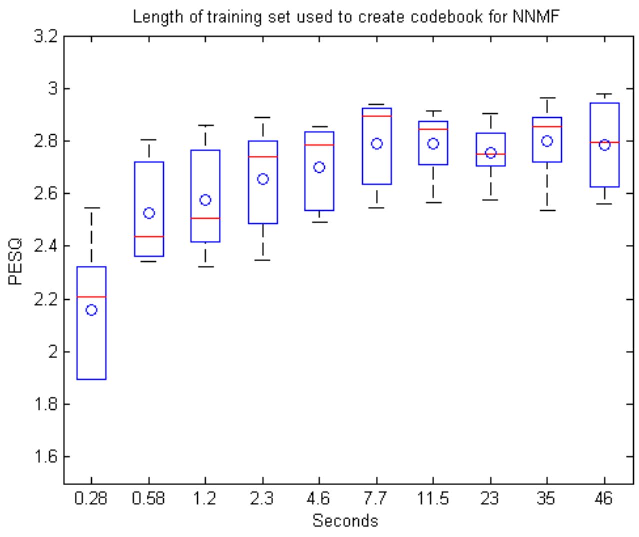


Figure 7.18: *The size of the training set as a function of PESQ for the NNMF algorithm.*

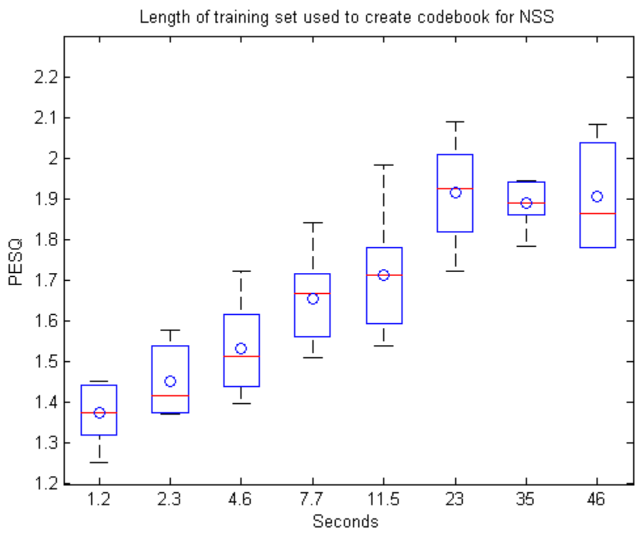


Figure 7.19: *The size of the training set as a function of PESQ for the NNS algorithm.*

CHAPTER 8

Results

This chapter contains the evaluation of the noise reduction algorithm reviewed in chapter 2, 3 and 4. The noise reduction algorithms are evaluated based on the measures in chapter 5 using the data from chapter 6 and the optimized parameters from chapter 7.

8.1 Objective Evaluation

This section contains the part of the evaluation that is based on the objective measures in chapter 5 including the PESQ measure. The three methods with the optimized parameters from chapter 7 has been run on dataset 3. The resulting PESQ values can be seen in figure 8.1 and 8.2. 'SPS' is the Spectral Subtraction algorithm, 'NNMFSPS' is the Non-Negative Matrix Factorization algorithm with the Generalized Spectral Subtraction addon that is optimized in section 7.4.1.2, 'NSSSPS' is the Non-Stationary Spectral Subtraction algorithm with the Generalized Spectral Subtraction algorithm optimized in section 7.3.1.2, 'IDEAL MASK' is the ideal binary mask calculated from the original speech and noise files, and 'NNMF MASK' is the binary mask calculated from the NNMF factorization. The blue circle is the mean, the red line is the median, the limit of the solid box corresponds to the lower and higher quartile, and the dotted line signifies the range of the data. A point is considered an outlier (red cross) if it is more than $1.5 \cdot IQR$ away from the upper or lower quartile, where IQR is the difference between the lower and upper quartile. The outputs of the methods have been concatenated two and two like in chapter 7, resulting in 22 speech estimates for each method.

The PESQ mean of each method is also shown in table 8.1. The NNMF-

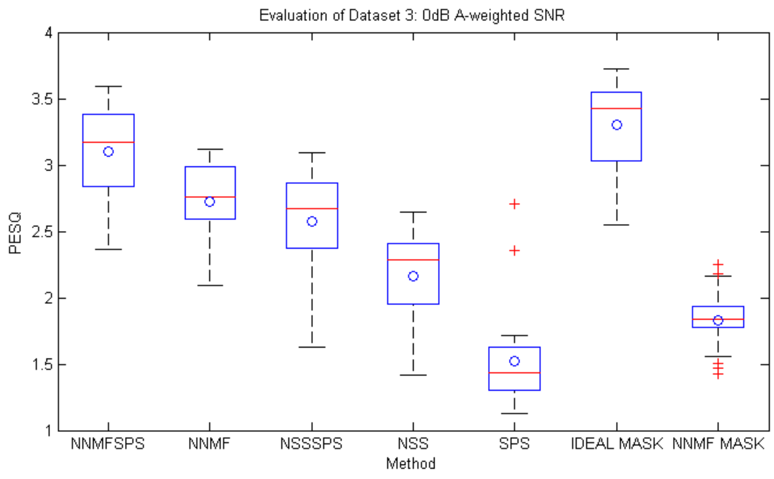


Figure 8.1: *The PESQ value for the 5 different methods and 2 binary mask separations on the 0dB A-weighted dataset 3. The mean of the NNMFSPS PESQ value is 3.10, which is 0.20 below the ideal mask.*

NNMFSPS	NNMF	NSSSPS	NSS	SPS	Id. Mask	NNMF Mask
3.10	2.72	2.57	2.16	1.52	3.30	1.83
2.46	2.19	1.67	1.61	1.17	2.95	1.53

Table 8.1: *The mean PESQ value for the 5 methods plus masks. The first row is the 0dB A-weighted SNR dataset 3 and the second row is the -11dB A-weighted SNR dataset 3.*

SPS is seen to have the highest PESQ mean value, while SPS performs the worst and NSS is in between. For both the NNMF and NSS case, it improves the PESQ measure to use SPS filtering. It should be noted that an exponentiation of the magnitude spectrogram is implemented in the NNMF algorithm, which means that the difference between the NNMF and NNMFSPS algorithm is only a first order auto-regressive smoother on the filter in the time domain. For the 0dB A-weighted SNR dataset, the NNMFSPS algorithm performs almost as well as the the ideal binary mask, which represents the theoretical limit

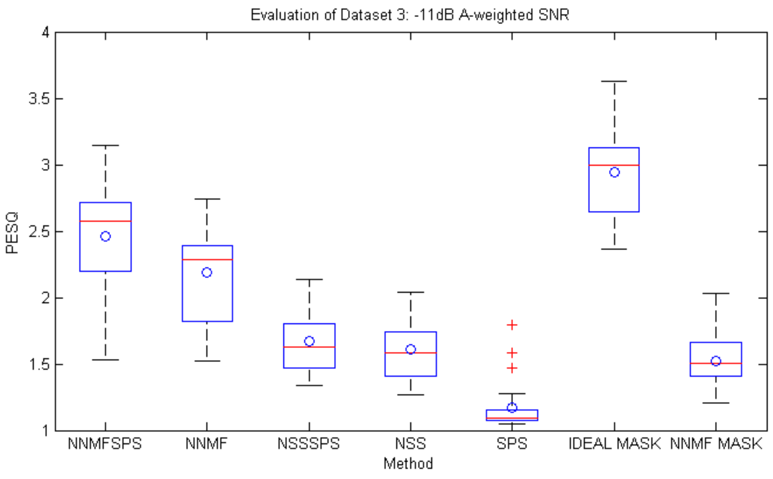


Figure 8.2: *The PESQ value for the 5 different methods and 2 binary separations on the -11dB A-weighted dataset 3. The mean of the NNMFSPS PESQ value is 2.46, which is 0.49 below the ideal mask.*

of noise reduction using binary masking. SPS has some PESQ scores (the red crosses) that are significantly higher, than the rest of the scores. Listening to the original noisy sound data, it is observed that files that score very high PESQ values using the SPS algorithm has very stationary wind noise. This is likely to be because of how the noise is estimated. As the noise estimate can not be updated during speech, the performance depends on how well the noise during non speech intervals resembles the noise during speech activity. If the wind noise is very stationary, then the estimation will be accurate. Finally it is seen that for the -11dB dataset, the SPS PESQ values have been squeezed against the lower bound of the PESQ measure and illustrates a weakness of the PESQ measure; That the predetermined range of possible PESQ values, makes the measure unsuitable for applications that lies outside this range.

For reference, the mean PESQ values from the optimization process in chapter 7 is shown in table 8.2. Overall there is little difference between the mean

NNMFSPS	NNMF	NSSSPS	NSS	SPS
3.14	2.87	2.46	2.04	1.54

Table 8.2: *The mean PESQ value for the 5 methods evaluated on the 0dB A-weighted SNR dataset 2.*

PESQ values of the two datasets. This is an encouraging result, because the

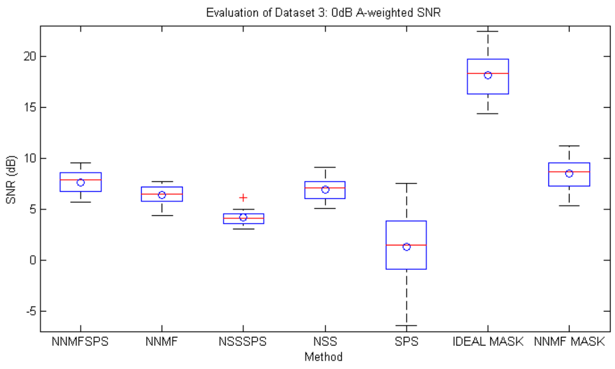


Figure 8.3: *The SNR results on all methods for the 0dB A-weighted SNR dataset 3.*

parameters were specifically optimized for the small dataset 2, while dataset 3 is a much larger dataset that is completely independent from dataset 2.

Dataset 3 has also been evaluated using the objective measures in section 8.1. However, because the results for a lot of the methods are similar, only a selection of them will be shown here. All the objective measure results can be seen in Appendix A. The SNR measure results for the 0dB A-weighted dataset 3 can be seen in figure 8.3. It is seen the the SPS algorithm performs the worst, while the NNMF and NSS algorithm has similar performance. The SPS addition increases the performance for the NNMF algorithm, while it decreases the performance for the NSS algorithm. A reason for is that the SNR measure is a simple energy measure, and the SPS addition adds broadband noise to the NSS output to help reduce musical noise. Also the parameter $\rho = 0.8$ (see section 7.3.1.2) increases the filter values in H, which means that the noisy signal has less filtering. The SPS addition to the NNMF algorithm only smoothes the filter in the timedomain. The NNMF Mask actually performs slightly better than the 5 original methods. The A-weighted SNR results for the 0dB A-weighted dataset 3 can be seen in figure 8.4. The most significant difference from the unweighted measures is that the SPS algorithm performs as well as the NNMF and NSS algorithm for the A-weighted measure results. This can be explained by noticing that the output of the SPS algorithm has large low-frequency artifacts from the wind noise during speech activity. Because the A-weighting puts very little emphasis on the low-frequency content, the SPS algorithm performs relatively better using A-weighted estimates. The unweighted and A-weighted NR results from the 0dB A-weighted dataset can be seen in figure 8.5 and 8.6. For the unweighted NR results all methods are close to 0dB except the SPS method. This is because of the high level of low-frequency wind content that is in both

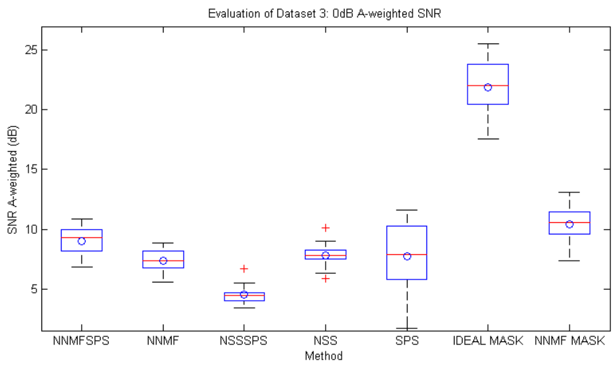


Figure 8.4: *The SNR A-weighted results on all methods for the 0dB A-weighted SNR dataset 3.*

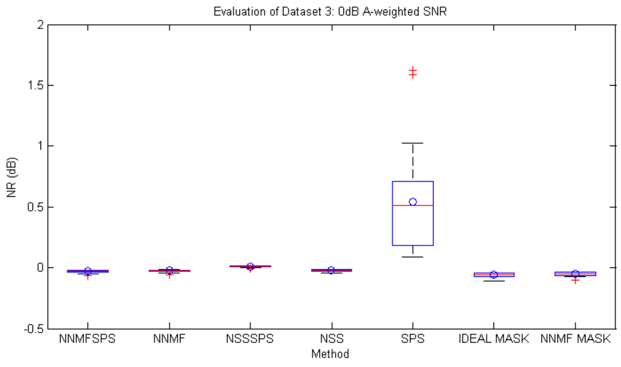


Figure 8.5: *The NR results on all methods for the 0dB A-weighted SNR dataset 3.*

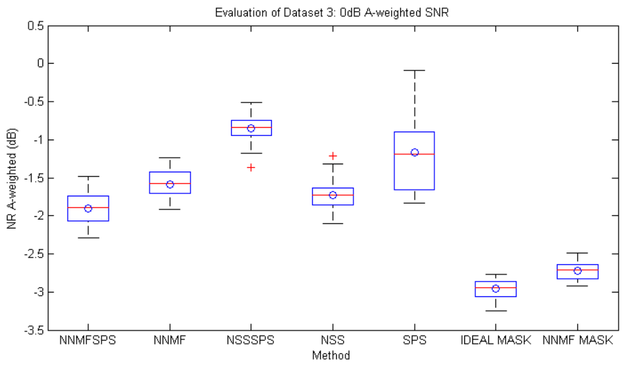


Figure 8.6: *The NR A-weighted results on all methods for the 0dB A-weighted SNR dataset 3.*

the nominator and denominator of the NR measure. This figure reinforces the assertion that the SPS output has a lot of noise artifacts compared to the other methods. For the A-weighted NR results, it is seen that the NSSSPS method has the highest value. Like before this is likely caused by the broadband noise that the SPS method adds to the output.

Looking at the figures, it seems there is a lot of redundancy in the measures. To investigate this, the correlation between all measures has been calculated and can be seen in table 8.3. The calculation is based on the 44 sound files from dataset 3 evaluated using the 7 different noise reduction methods resulting in 308 values for each measure. They have been sorted according to their correlation with PESQ. A graphical representation of the correlation coefficients can

	1	2	3	4	5	6	7	8	9
1 $PESQ$	1.00	0.60	0.57	0.48	0.47	-0.41	-0.42	-0.49	-0.52
2 SNR	0.60	1.00	0.95	0.83	0.80	-0.66	-0.66	-0.80	-0.86
3 SNR_{seg}	0.57	0.95	1.00	0.92	0.92	-0.55	-0.53	-0.71	-0.83
4 $SNRA$	0.48	0.83	0.92	1.00	0.97	-0.24	-0.22	-0.46	-0.68
5 $SNRA_{seg}$	0.47	0.80	0.92	0.97	1.00	-0.20	-0.18	-0.42	-0.66
6 NR_{seg}	-0.41	-0.66	-0.55	-0.24	-0.20	1.00	0.98	0.93	0.74
7 NR	-0.42	-0.66	-0.53	-0.22	-0.18	0.98	1.00	0.93	0.71
8 NRA	-0.49	-0.80	-0.71	-0.46	-0.42	0.93	0.93	1.00	0.84
9 NRA_{seg}	-0.52	-0.86	-0.83	-0.68	-0.66	0.74	0.71	0.84	1.00

Table 8.3: *The correlation coefficients between all objective measures evaluated on dataset 3. "A" means that the measure has been A-weighted, e.g. SNRA is the SNR A-weighted measure*

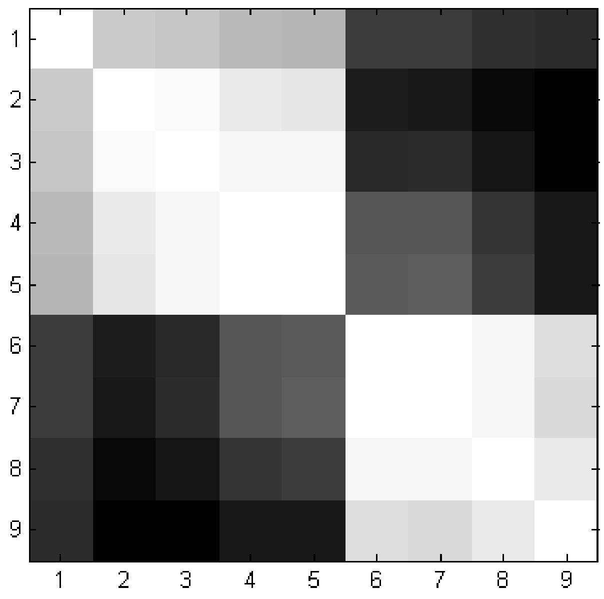


Figure 8.7: *The correlation coefficients between all objective measures evaluated on dataset 3. The values have been converted to grayscale, so "white" is 1 and "black" is -1.*

be seen in figure 8.7. It is seen that the only measure that is not highly correlated with another measure is PESQ. Especially segmented measures are highly correlated with the corresponding not-segmented measure, e.g. SNR is highly correlated with SNR_{seg}. PESQ is most correlated with SNR at 0.60. Also all SNR measures are correlated with each other, just as all NR measures are. Some negative correlation is also seen between the SNR and NR measures. Overall it is not clear which energy measure is best suited for evaluating the performance of the noise reduction methods. Even though the A-weighted measures could be expected to capture more of the perceptual quality of the speech estimate, this can not be concluded from the analysis of this data.

8.2 Subjective Evaluation

This section contains the evaluation of the noise reduction methods that is based on the subjective measures from section 5.2. A webpage[1] has been created with a listening test. By following the instructions on the webpage, test subjects can listen to and evaluate the outputs of the noise reduction algorithms. The answers are saved in a MySQL database running on the server and can be extracted for analysis. This way of setting up a listening test is very practical for the test subjects, because it can be done anywhere that has a computer and a set of speakers. The disadvantage is that the test environment is not very controlled and a lot of unknown factors can skew the results. There is way to control what quality of speakers the test subject is using or how the test subject fills out the test, and the results should therefore be taken with some reservation.

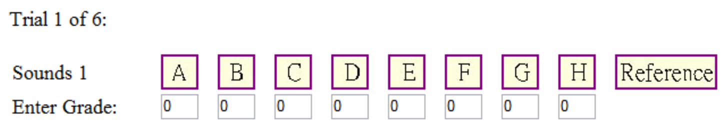
At the beginning of the test, the test subjects are instructed to listen through the available files and adjust the sound volume of the speakers to a comfortable level, and keep it constant for the duration of the test. The listening test is divided up into two individual tests - a MUSHRA test and a preference test. Before submitting an answer on the webpage, the test subject must fill out some personal information and disclose whether he/she has experience in listening to sound in a critical way.

In total 22 people took the listening test. Out of those, 2 were discarded, because of missing data and for the MUSHRA test, another 2 were discarded, because they graded more than 40% of the files to 0. Out of the 20 complete submissions, 12 indicated that they had experience listening to sound in a critical way.

8.2.1 MUSHRA Test

This test is intended to subjectively evaluate the differences between the methods reviewed in chapter 2, 3 and 4. It uses the data from Dataset 4 as input to the methods and consists of 6 trials - one for each noisy speech signal in Dataset 4. This data consists of speech from the TIMIT database played back on a speaker and recorded by a microphone, while wind noise of about 5-10m/s is present. The test subjects are instructed in completing the test in a similar, but shorter, way to the description in section 5.2.4. The test subjects are told to listen through all sound files before scoring them. Each of the 6 trials that should be answered, looks like figure 8.8. As mentioned in the previous section,

[1]The webpage for the listening test can be found at: http://www.kristian.timm-danmark.dk


Trial 1 of 6:

Sounds 1 A B C D E F G H Reference

Enter Grade: 0 0 0 0 0 0 0 0

Figure 8.8: *The window that is presented to the test subjects for each trial. By clicking on a letter, the corresponding file will be played. The grade is given by entering a number between 0 and 100 in the box below the letter.*

18 usable submissions has been acquired for the MUSHRA test and out of those 10 indicated that they had experience listening to sound in a critical way. The noise reduction methods that have to be scored in the MUSHRA test are:

NNMFSPS: This is the Non-Negative Matrix Factorization algorithm with the Generalized Spectral Subtraction algorithm used as backend.

NNMF: This is the Non-Negative Matrix Factorization algorithm.

NSSSPS: This is the Non-Stationary Spectral Subtraction algorithm with the Generalized Spectral Subtraction algorithm used as backend.

SPS: This is the Generalized Spectral Subtraction algorithm.

Ideal Mask: This is an ideal binary mask calculated from the original speech file from the TIMIT database and a random wind noise file from Dataset 6.

NNMF Mask: This is the Non-Negative Matrix Factorization algorithm, where the factorization has been used to create a binary mask.

Lowpass: This is the original speech file from the TIMIT database that has been lowpass filtered at 3.5kHz as specified in the MUSHRA standard (see section 5.2.4).

Reference: This is the original speech file from the TIMIT database.

It should be noted that Ideal Mask, Lowpass, and Reference uses the original TIMIT database speech files while all other methods uses the speech file recorded as in the description of Dataset 4 (see section 6.5).

The results from the 6 trials of the MUSHRA test are combined and the total result can be seen in figure 8.9. Because of the uncertainty of the test environment, all outliers have been disregarded in the calculation of the mean. The mean value of the results can also be seen in table 8.4. It is seen that the NNMF

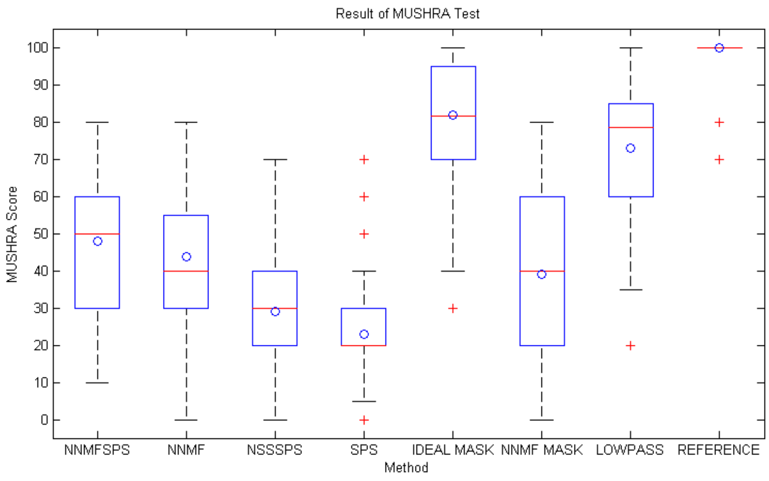
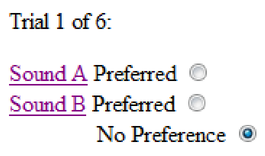


Figure 8.9: *The MUSHRA test result. The outliers (red crosses) are disregarded in the calculation of the mean (blue circle).*

NNMFSPS	NNMF	NSSSPS	SPS	Ideal M.	NNMF M.	Lowp.	Ref.
48	44	29	23	82	39	73	100

Table 8.4: *The mean of the MUSHRA test.*

outperforms the NSSSPS and SPS algorithm. The use of the SPS algorithm as a backend addition to the NNMF algorithm increases the performance, while the binary masking performs worse than just the NNMF algorithm. Generally though, the methods that relies on the original TIMIT database file have much better performance, which is not surprising. The interpretation and overall conclusion is the same if only the replies of experienced listeners are considered. Overall the results are very similar to the PESQ scores, which is a positive result.


Trial 1 of 6:

Sound A Preferred ○
Sound B Preferred ○
No Preference [✓ ●]

Figure 8.10: *The window that is presented to the test subjects for each trial of the preference test. The test subject should indicate which sound file is preferred or otherwise choose 'No Preference'.*

NNMFSPS	No Noise Reduction	No Preference
108	6	6

Table 8.5: *The total number of 'votes' for each possible choice in the preference test.*

8.2.2 Preference Test

The preference test is intended to evaluate whether the NNMFSPS algorithm is better than not using any noise reduction in heavy wind conditions (10-20m/s). In heavy wind conditions, the speech is harder to estimate, which results in more distorted speech. If the estimated speech is very distorted, with too much musical noise, it might be more pleasant to listen to the original noisy speech instead.

The noisy speech signals from Dataset 5 is used for this test, resulting in 6 preference trials. This dataset consists of speech from the TIMIT database played back on a speaker and recorded by a microphone, while wind noise of about 10-20m/s is present. The noise codebook for the NNMFSPS algorithm in this test, is trained using Dataset 6 instead of dataset 1, which has been used in all other cases. Each of the 6 trials that should be answered, looks like figure 8.10. In total 20 submission have been acquired for the preference test, where 12 of these indicated that they had experience listening to sound in a critical way. For the 6 trials, the total result of the preference test can be seen in table 8.5. It is concluded that using the NNMFSPS algorithm during heavy wind is better than no noise reduction.

8.2.3 Spectrogram of Filtered Signals

The spectrogram of one of the clean sentences in the MUSHRA test can be seen in figure 8.11. Figure 8.12-8.14 shows the filtered version using methods NNMFSPS, NSSSPS and SPS.

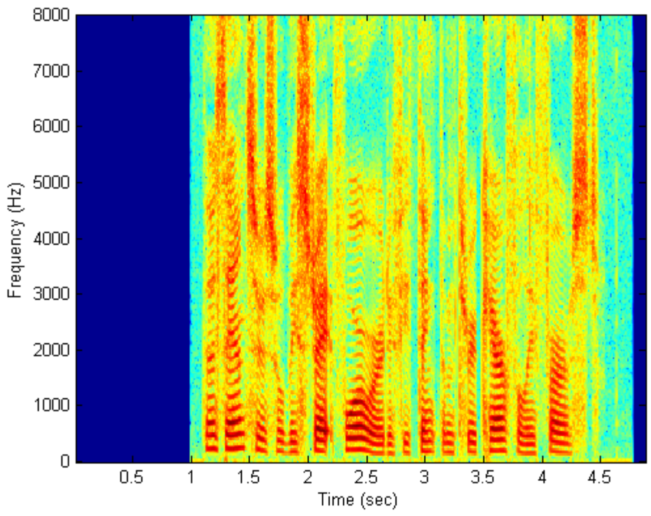


Figure 8.11: *The original speech file from the TIMIT database.*

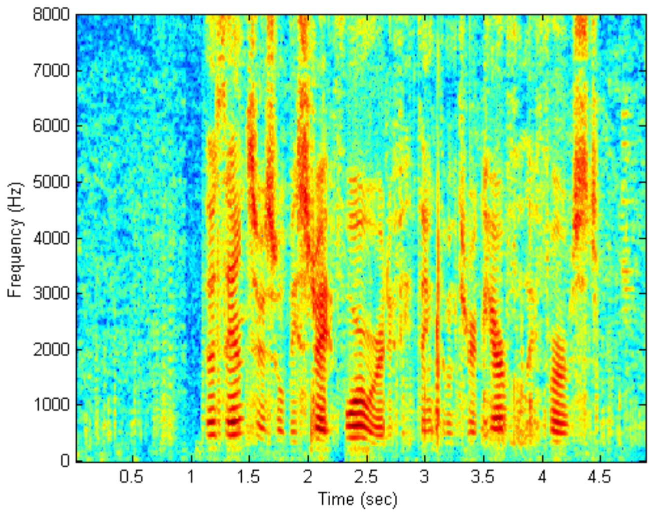


Figure 8.12: *Filtered speech signal using NNMFSPS.*

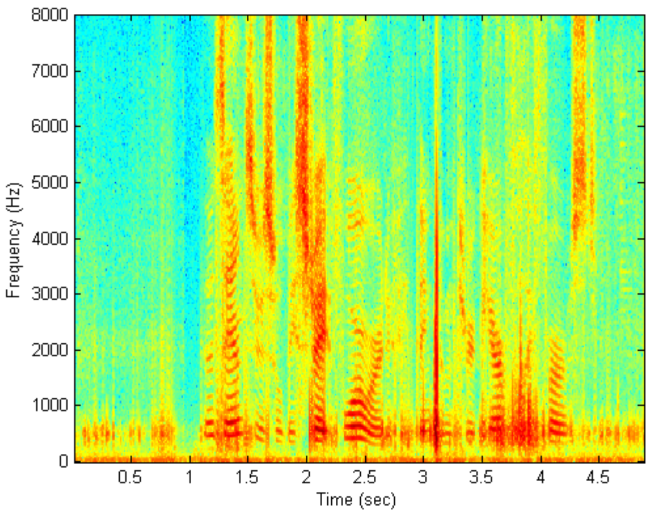


Figure 8.13: *Filtered speech signal using NSSSPS.*

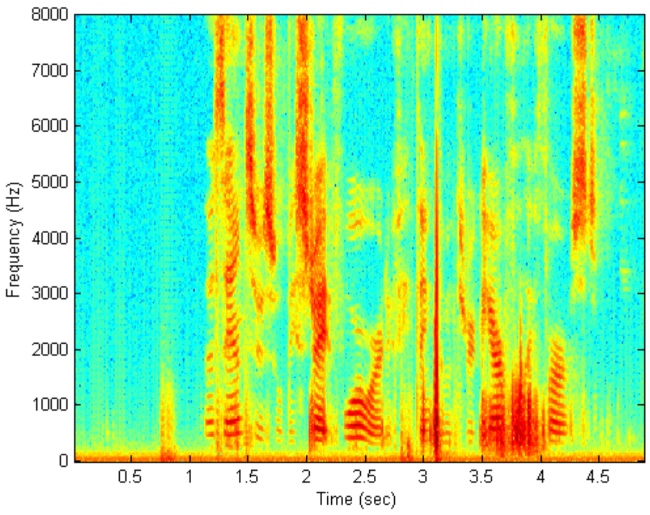


Figure 8.14: *Filtered speech signal using SPS.*

Chapter 9

Conclusion

This chapter contains the main conclusion of the thesis along with a discussion of future work.

In this thesis a number of noise reduction techniques have been reviewed, implemented, and evaluated. The overall conclusion is that the Non-Negative Matrix Factorization algorithm provides the best noise reduction of the investigated methods. This is based on both a perceptual and energy-based evaluation. The perceptual evaluation is only based on speech quality and not intelligibility. An advantage of this method is that it does not need a Voice Activity Detector (VAD) and only assumes a-priori information about the wind noise. In fact, the method can be viewed solely as an advanced noise estimator. The downside of the algorithm is that it has a relatively high computational complexity. This can potentially be improved with a more advanced cost minimization algorithm than gradient descent. In the following, the individual methods and measures will be given a more detailed evaluation.

A Generalized Spectral Subtraction algorithm has been implemented for this thesis. The method serves as a baseline comparison against more advanced methods and is well suited to demonstrate how the generalizations improve the filtering. Among the generalizations, particularly the exponentiation of the magnitude spectrogram and the smoothing of the filter is shown to improve the speech estimate. The method has a low computational complexity. In comparison to the other methods implemented for the thesis, the method has an overall poor performance. Especially the noise residual is very large for this method, because the method does not update the noise estimate during speech activity.

A Non-Stationary Spectral Subtraction method has been implemented. The advantage of this method over the Generalized Spectral Subtraction method is that it is possible to estimate the noise while a person is talking. This is possible by using noise and speech codebooks, containing smoothed spectrums of the signals that are expected to be found in the noisy signal. The computational complexity of the method is higher than for the Generalized Spectral Subtraction, mostly because the codebooks have to be searched to find the best fit to the current noisy signal frame. It is shown that intelligently searching the codebooks can improve the computational complexity. The method outperforms the Generalized Spectral Subtraction in all measures except for the A-weighted SNR measure. This is because the A-weighted measure puts very little emphasis on the low-frequency content of residual noise that is very large for the Generalized Spectral Subtraction method. The generalizations from the Generalized Spectral Subtraction method is shown to improve the perceptual objective PESQ measure. This, however, decreases the energy measures.

A Non-Negative Matrix Factorization algorithm has been implemented. This method factorizes the magnitude spectrogram into a dictionary matrix and a codebook matrix. By introducing a precomputed noise codebook into the factorization, it is shown that speech and noise can be estimated. Like the Non-Stationary Spectral Subtraction method, this method can update the noise estimate during speech. The computational complexity of the method is high compared to the other methods in this thesis. Overall the method has the best performance for filtering wind noise from speech and the generalizations from the Generalized Spectral Subtraction method is shown to improve both the energy and perceptual measures for this method.

The noise reduction methods have been evaluated based on speech and noise that has been summed in a computer and noisy speech that has been recorded while wind noise is present. It is generally found that the noisy speech, that is recorded while wind noise is present suffers from more speech distortion than when speech and noise is summed in a computer.

A Signal-to Noise Ratio and a Noise Residual measure, along with A-weighted and segmented variations of these, have been implemented for evaluating the energy improvement of the noise reduction methods. A PESQ measure, that has high correlation with subjective listening tests, has also been used to optimize and evaluate the methods. Finally the methods have been evaluated using a subjective listening test called MUSHRA. Together these measures cover both the perceptual and energy-wise evaluation of the noise reduction methods. It is shown that the energy measures have high correlation between them. The A-weighted measures have high emphasis on the frequency range from 1000-6000Hz while the other energy measures weights all frequencies equally, which is reflected in how they score the methods. Particular the Generalized Spectral Subtraction method, which does not estimate the low frequency wind noise very well, scores low in the normal energy measures, while it scores almost as good as the other methods in the A-weighted measures. The MUSHRA results have a high uncertainty, because of the uncontrolled testing environment, but the results are similar to the PESQ scores, which is encouraging.

9.1 Future Work

The Non-Negative Matrix Factorization algorithm can be improved by implementing other and more advanced cost-minimization algorithms. This can potentially decrease the convergence time of the minimization problem and lead to a method with lower computational complexity. The current multiplicative update rules only uses first-order information to find the minimum of the costfunction and it is expected that methods that take the curvature of the costfunction into consideration can improve the convergence time. This can be done by the second derivative of the costfunction (the Hessian matrix). This might also improve the actual filtering.

In section 4.7, a suggestion for a real-time implementation of the Non-Negative Matrix factorization is given. The result is basically the same as when the entire magnitude spectrogram is available at once, because of the sliding window. The computational complexity, however, is greatly increased for real time operation and future work could include different ways to improve this implementation.

The perceptual evaluation of the noise reduction methods, only include the speech quality. Other measures, like speech intelligibility, could be used to give a better estimate of how the methods perform in heavy wind conditions.

APPENDIX A

Objective Energy Evaluation of Noise Reduction Methods

This Appendix contains the full SNR and NR graphical evaluation from chapter 8.1 in figures A.1 -A.16.

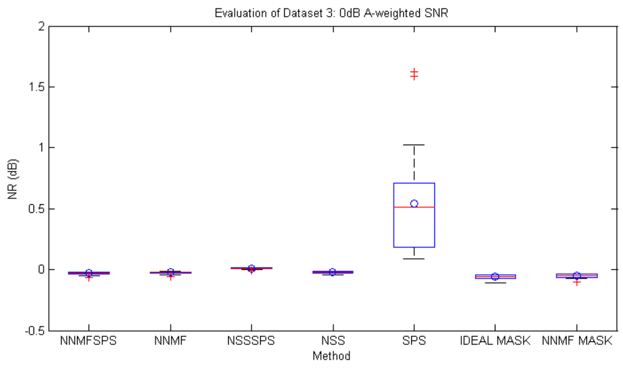


Figure A.1: *The NR results on all methods for the 0dB A-weighted SNR dataset 3.*

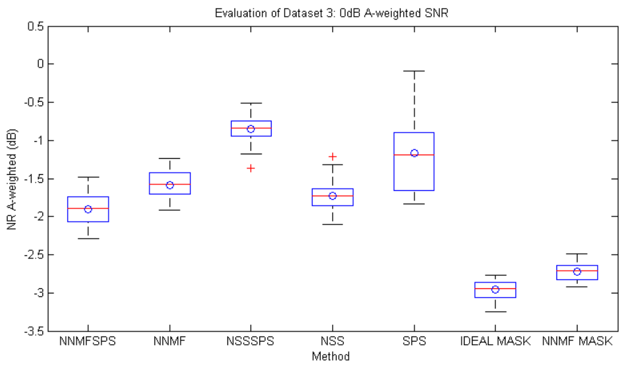


Figure A.2: *The NR A-weighted results on all methods for the 0dB A-weighted SNR dataset 3.*

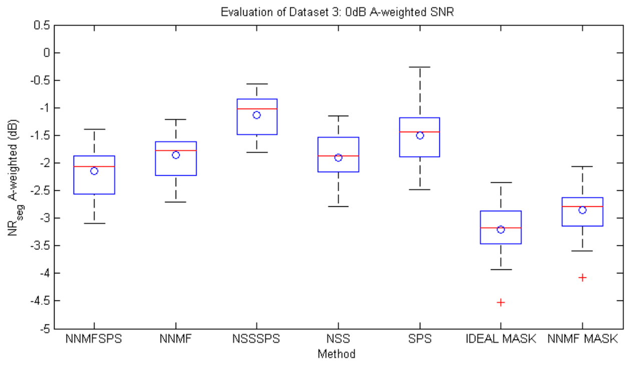


Figure A.3: *The NR_{seg} A-weighted results on all methods for the 0dB A-weighted SNR dataset 3.*

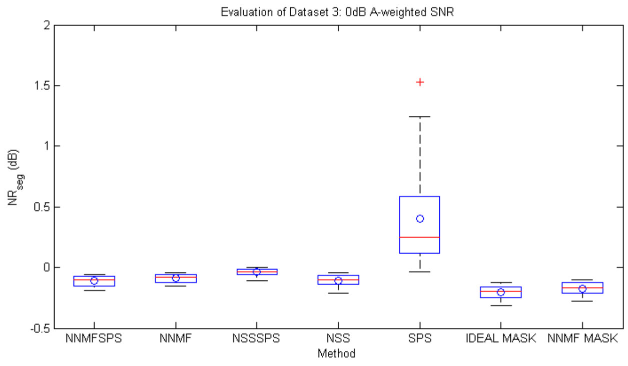


Figure A.4: *The NR_{seg} results on all methods for the 0dB A-weighted SNR dataset 3.*

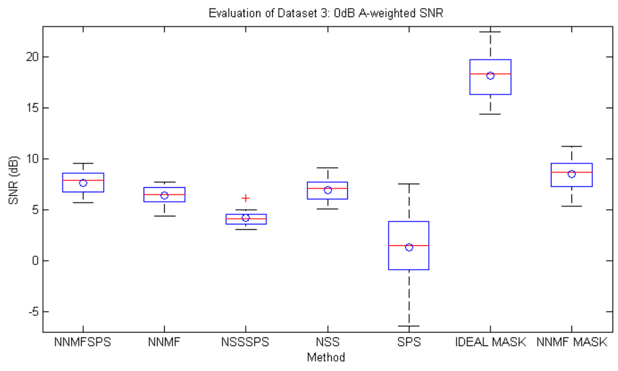


Figure A.5: *The SNR results on all methods for the 0dB A-weighted SNR dataset 3.*

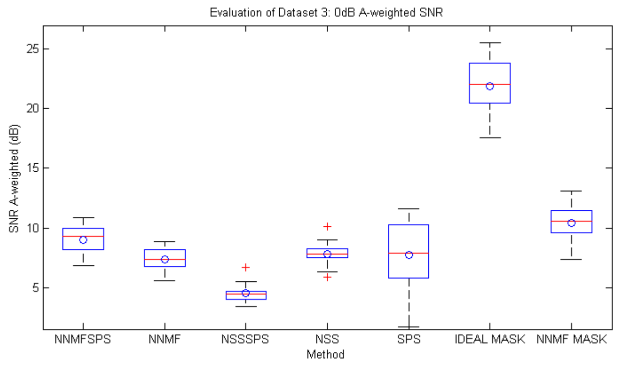


Figure A.6: *The SNR A-weighted results on all methods for the 0dB A-weighted SNR dataset 3.*

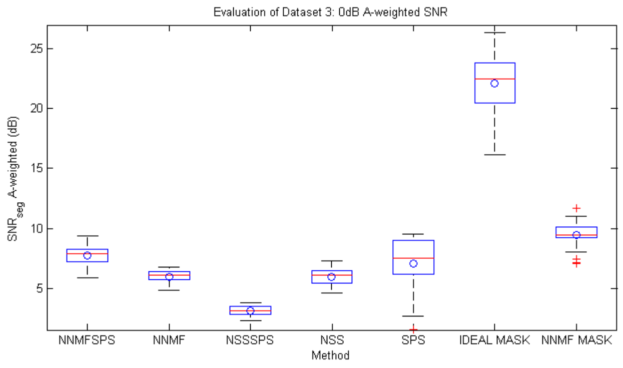


Figure A.7: *The SNR_{seg} A-weighted results on all methods for the 0dB A-weighted SNR dataset 3.*

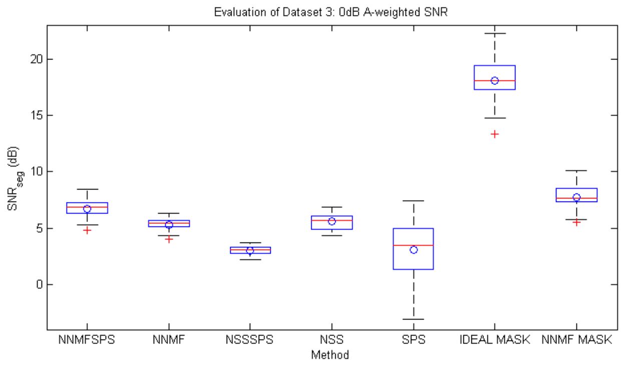


Figure A.8: *The SNR_{seg} results on all methods for the 0dB A-weighted SNR dataset 3.*

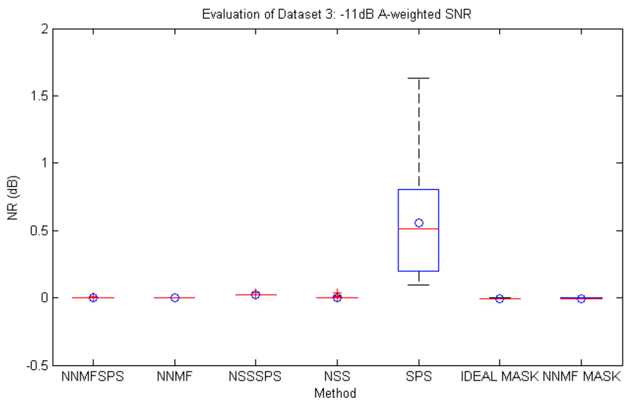


Figure A.9: *The NR results on all methods for the -11dB A-weighted SNR dataset 3.*

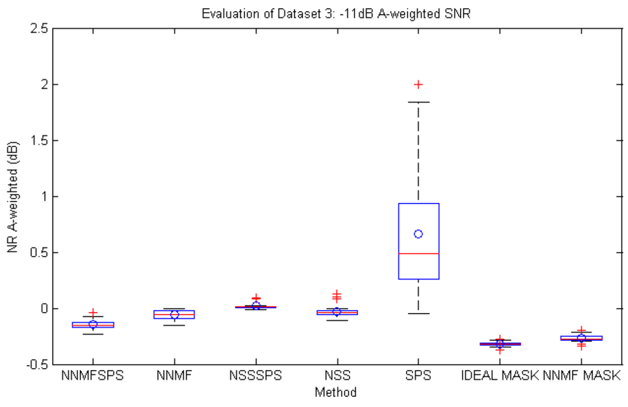


Figure A.10: *The NR A-weighted results on all methods for the -11dB A-weighted SNR dataset 3.*

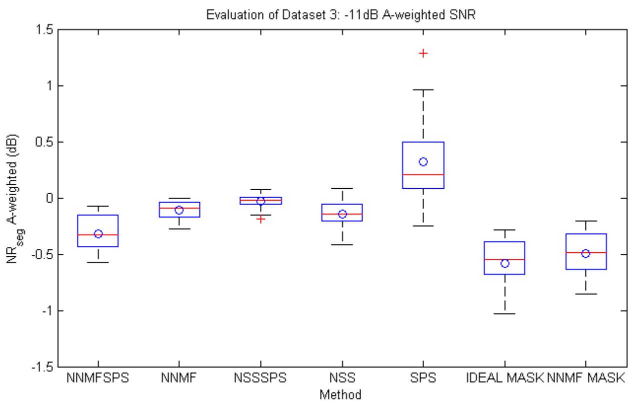


Figure A.11: *The NR_{seg} A-weighted results on all methods for the -11dB A-weighted SNR dataset 3.*

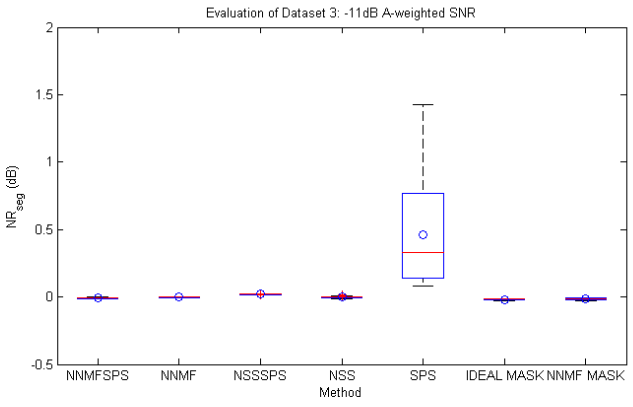


Figure A.12: *The NR_{seg} results on all methods for the -11dB A-weighted SNR dataset 3.*

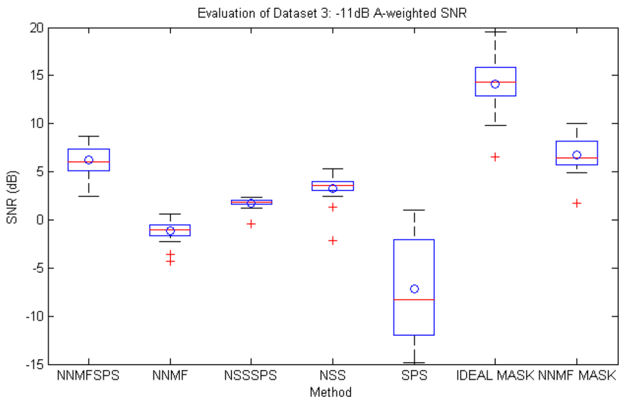


Figure A.13: *The SNR results on all methods for the -11dB A-weighted SNR dataset 3.*

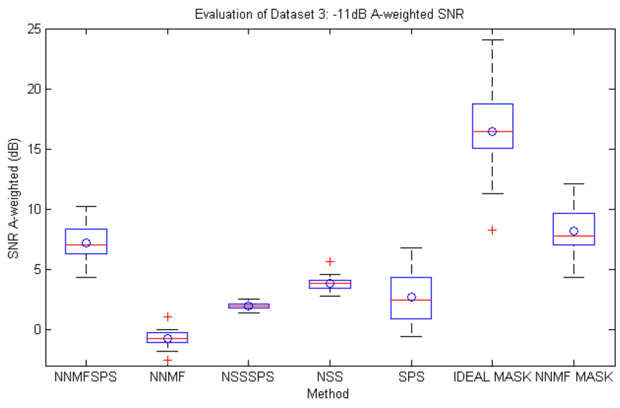


Figure A.14: *The SNR A-weighted results on all methods for the -11dB A-weighted SNR dataset 3.*

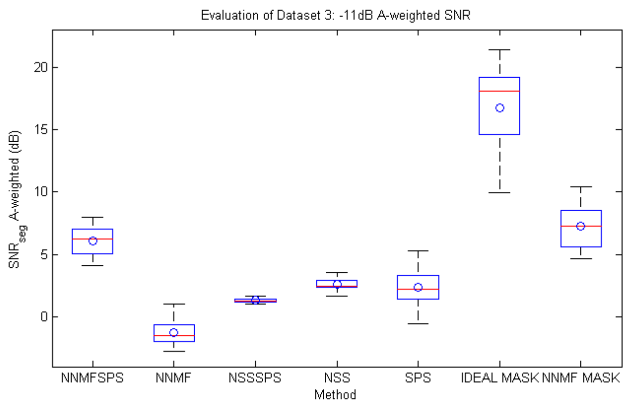


Figure A.15: *The SNR$_{seg}$ A-weighted results on all methods for the -11dB A-weighted SNR dataset 3.*

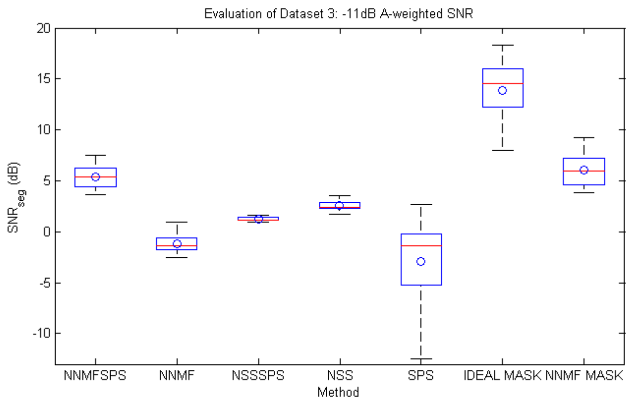


Figure A.16: *The SNR$_{seg}$ results on all methods for the -11dB A-weighted SNR dataset 3.*

Appendix B

Matlab Code

This Appendix contains the main matlab code developed for the thesis.

B.1 Generalized Spectral Subtraction

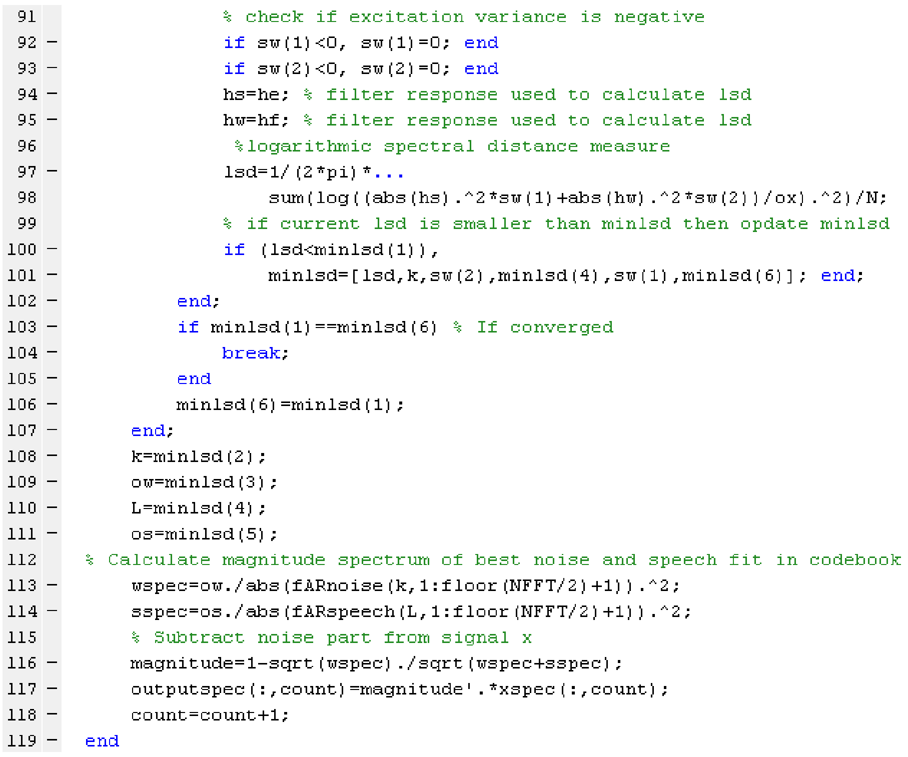
```
function SE=SpS(NSS,WES,lambdah,L,expo,Hexpo,alpha,beta,betafilter)
% SE=SpS(NSS,WES,lambdas,lambdaw,expo,alpha,beta)
%
% Performs Spectral Subtraction with generalizations. If only NSS and WES
% is given as arguments all other parameters will default to standard
% Power Spectral Subtraction.
%
% Input:
% NSS - Noisy Signal Spectrogram
% WES - Magnitude Wind Estimate Spectrogram
% lambdah - Smoothing factor for filter H (0 to 1, 0=no smoothing)
% L - Length of avereging filter over frequency
% expo - Exponent to magnitude spectrogram (0.7 Stevens power law, 1
% Magnitude Spectrogram, 2 Power Spectrogram)
% Hexpo - H exponent (0.5 Power Spectral Subtraction, 1 Wiener Filtering)
% alpha - Oversubtraction factor
% beta - Lower bound on filter output (>0)
% betafilter - Lower bound on output (>0)
```

```
%
% Output:
% SE - Speech estimate
%
% Written by: Kristian Timm Andersen, IMM DTU 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ((nargin<3)||isempty(lambdah))
    lambdah=0;
end
if ((nargin<4)||isempty(L))
    L=0;
end
if ((nargin<5)||isempty(expo))
    expo=1;
end
if ((nargin<6)||isempty(Hexpo))
    Hexpo=1;
end
if ((nargin<7)||isempty(alpha))
    alpha=ones(size(NSS));
end
if ((nargin<8)||isempty(beta))
    beta=0;
end
if ((nargin<9)||isempty(betafilter))
    betafilter=0;
end

[NFFT2,LS]=size(NSS);
NSStemp=abs(NSS); % Magnitude
WEStemp=abs(WES); % Magnitude
NSStemp=NSStemp.^expo; % Exponentiate Noisy Signal
WEStemp=WEStemp.^expo; % Exponentiate Wind Estimate
% Filter and correct estimation errors below zero
H=max((1-alpha.*WEStemp./(NSStemp+eps)),beta).^Hexpo;
Hs=[H(:,1),zeros(NFFT2,LS-1)];
for i=2:LS % smoothe filter
    Hs(:,i)=lambdah*Hs(:,i-1)+(1-lambdah)*H(:,i);
end
H=H.*Hs-Hs.*Hs+Hs.*1; % Weight smoother
for i=1:size(H,1) % average over frequency
    H(i,:)=1/(1+2*L)*sum(H(max(1,i-L):min(size(H,1),i+L),:),1);
end
SE=H.*(abs(NSS).^expo); % Perform Spectral Subtraction
floor=betafilter*WEStemp; % Lower Bound on output
% set floor, exponentiation and phase
SE=max(floor.^(1/expo),abs(SE).^(1/expo)).*exp(j*angle(NSS));
```

B.2 Non-Stationary Spectral Subtraction

```
function [outputspec]=NSSpecSubn(x,N,NARx,poverlap,lwin,NFFT,...
    fARnoise,fARspeech)
% [outputspec]=NSSpecSubn(xspec,N,NARx,poverlap,lwin,NFFT,...
%    fARnoise,fARspeech)
% NSSpecsub assumes x to be a signal containing speech and noise.
% It filters the noise away from the x by a nonlinear spectral
% subtraction algorithm based on the codebooks ARspeech and ARnoise
% containing AR-parameters of speech and noise respectively.
%
% Input:
% x - Input signal
% N - Number of integration points
% NARx - Number of AR-coefficients used to model the noisy speech
% poverlap - percent overlap between frames
% lwin - length of each frame
% NNFT - Number of points used in FFT
% fARnoise - Frequency response of noise codebook
% fARspeech - Frequency response of speech codebook
%
% Output:
% outputspec - Speech estimate spectrogram
%
% By: Kristian Timm Andersen, IMM DTU 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
win=hanning(lwin);
noverlap=round(lwin*poverlap);
xspec=spectrogram(x,win, noverlap, NFFT);
% initialize vector to hold the spetrogram of the output
outputspec=zeros(NFFT/2+1,floor((length(x)/lwin-poverlap)/(1-poverlap)));
count=1; % Counting variable
% Select current timeframe
for m=1:ceil(lwin*(1-poverlap)):(length(x)-lwin+1)
    %AR parameters and excitation variance of current timeframe of x
    [ARx,ox] = aryule(x(m:m+lwin-1),NARx);
    fARx=freqz(ARx,1,N,'whole')';
    fARx2=fARx.*fARx;
    % intialize vector that holds minimum logarithmic spectral distance,
    % and the optimal speech and noise from codebook
    minlsd=[inf,1,0,0,0,0];
    while 1 % Optimization loop
        hd=fARx2./(fARnoise(minlsd(2),:).*fARnoise(minlsd(2),:));
        d=sum(abs(hd).^2)/(N*ox^2);
        hf=fARx./fARnoise(minlsd(2),:);
        f=sum(abs(hf).^2)/(N*ox);
        for L=1:size(fARspeech,1) % Search through entire speech codebook
             % First filter response in excitation variance equation
            ha=fARx2./(fARspeech(L,:).*fARspeech(L,:));
             % Perform numerical integral of the power of the filter
            a=sum(abs(ha).^2)/(N*ox^2);
            % This and following lines are similar to ha
            he=fARx./fARspeech(L,:);
            % This and following lines are similar to a
            e=sum(abs(he).^2)/(N*ox);
            hb=fARx2./(fARspeech(L,:).*fARnoise(minlsd(2),:));
            b=sum(abs(hb).^2)/(N*ox^2);
            c=b;
            sw=[a,b;c,d]\[e;f]; % solve the system of equations
            % check if excitation variance is negative
            if sw(1)<0, sw(1)=0; end
            if sw(2)<0, sw(2)=0; end
            hs=he; % filter response used to calculate lsd
            hw=hf; % filter response used to calculate lsd
            %logarithmic spectral distance measure
            lsd=1/(2*pi)*...
                sum(log((abs(hs).^2*sw(1)+abs(hw).^2*sw(2))/ox).^2)/N;
            % if current lsd is smaller than minlsd then opdate minlsd
            if (lsd<minlsd(1))
                minlsd=[lsd,minlsd(2),sw(2),L,sw(1),minlsd(6)]; end;
        end;
        if minlsd(1)==minlsd(6)
            break;
        end
        minlsd(6)=minlsd(1);
        % First filter response in excitation variance equation
        ha=fARx2./(fARspeech(minlsd(4),:).*fARspeech(minlsd(4),:));
        % Perform numerical integral of the power of the filter
        a=sum(abs(ha).^2)/(N*ox^2);
        % This and following lines are similar to ha
        he=fARx./fARspeech(minlsd(4),:);
        % This and following lines are similar to a
        e=sum(abs(he).^2)/(N*ox);
        for k=1:size(fARnoise,1) % Search through entire noise codebook
            hb=fARx2./(fARspeech(minlsd(4),:).*fARnoise(k,:));
            b=sum(abs(hb).^2)/(N*ox^2);
            c=b;
            hd=fARx2./(fARnoise(k,:).*fARnoise(k,:));
            d=sum(abs(hd).^2)/(N*ox^2);
            hf=fARx./fARnoise(k,:);
            f=sum(abs(hf).^2)/(N*ox);
            sw=[a,b;c,d]\[e;f]; % solve the system of equations
```



```
            % check if excitation variance is negative
            if sw(1)<0, sw(1)=0; end
            if sw(2)<0, sw(2)=0; end
            hs=he; % filter response used to calculate lsd
            hw=hf; % filter response used to calculate lsd
             %logarithmic spectral distance measure
            lsd=1/(2*pi)*...
                sum(log((abs(hs).^2*sw(1)+abs(hw).^2*sw(2))/ox).^2)/N;
            % if current lsd is smaller than minlsd then opdate minlsd
            if (lsd<minlsd(1)),
                minlsd=[lsd,k,sw(2),minlsd(4),sw(1),minlsd(6)]; end;
        end;
        if minlsd(1)==minlsd(6) % If converged
            break;
        end
        minlsd(6)=minlsd(1);
    end;
    k=minlsd(2);
    ow=minlsd(3);
    L=minlsd(4);
    os=minlsd(5);
% Calculate magnitude spectrum of best noise and speech fit in codebook
    wspec=ow./abs(fARnoise(k,1:floor(NFFT/2)+1)).^2;
    sspec=os./abs(fARspeech(L,1:floor(NFFT/2)+1)).^2;
    % Subtract noise part from signal x
    magnitude=1-sqrt(wspec)./sqrt(wspec+sspec);
    outputspec(:,count)=magnitude'.*xspec(:,count);
    count=count+1;
end
```

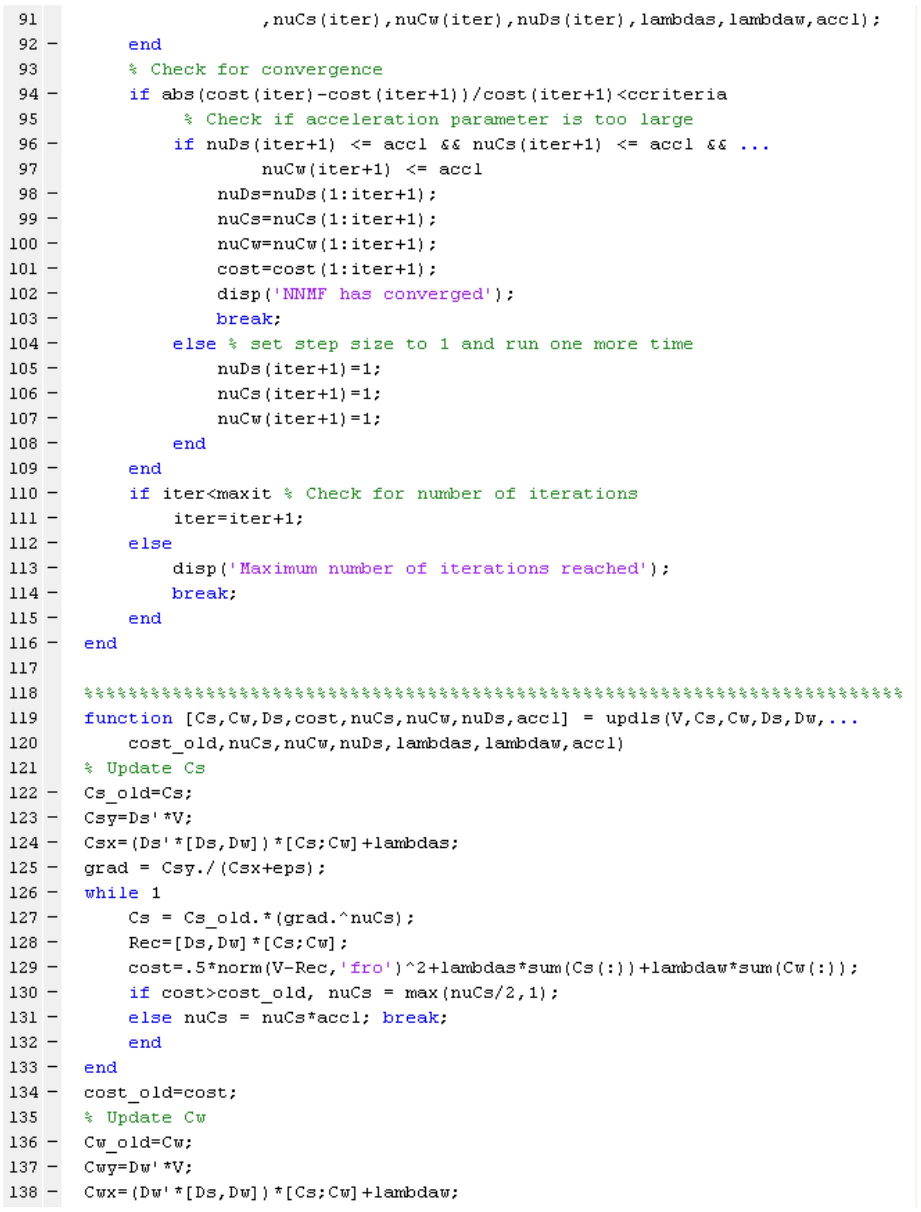
B.3 Non-Negative Matrix Factorization

```
function [Ds,Dw,Cw,Cs,cost,nuCs,nuCw,nuDs]= NNMF(V,Dw,ds,lambdas,...
    lambdaw, maxit,ccriteria,accl,costfct)
% [Ds,Dw,Cw,Cs,cost,nuCs,nuCw,nuDs]= NNMF(V,Dw,ds,lambdas,...
%     lambdaw, maxit,ccriteria,accl,costfct)
%
% Input:
% V - The magnitude spectrogram of the sound to be filtered
% Dw - The windnoise dictionary
% ds - The number of speech elements to be found
% lambdas - The speech sparsity parameter
% lambdaw - The noise sparsity parameter
% maxit - The maximum number of iterations
% ccriteria - The criteria for stopping the optimization of cost function
% accl - A parameter multiplied to each optimization step
% costfct - The used costfunction, 'ls': Least squares (default) or 'kl:
% Kullbach leibler
%
% output:
% Ds - The normalized speech dictionary
% Dw - The normalized noise dictionary
% Cw - The noise codebook
% Cs - The speech codebook
% cost - The cost in all iterations
% nuCs - The acceleration parameter for the speech codebook in all
%   iterations
% nuCw - The acceleration parameter for the noise codebook in all
%   iterations
% nuDs - The acceleration parameter for the speech dictionary in all
%   iterations
%
% The code is an adapted version of a matlab demofile from
% http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=4521
%
% written by: Kristian Timm Andersen, IMM DTU 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Check optinal Input
if ((nargin<4)||isempty(lambdas))
    lambdas=0;
end
if ((nargin<5)||isempty(lambdaw))
    lambdaw=0;
```

```
end
if ((nargin<6)||isempty(maxit))
    maxit=500;
end
if ((nargin<7)||isempty(ccriteria))
    ccriteria=1e-4;
end
if ((nargin<8)||isempty(accl))
    accl=1.3;
end
if ((nargin<9)||isempty(costfct))
    costfct='kl';
end

% Initialize
[Ls,Lt]=size(V);
Cw=rand(size(Dw,2),Lt);
Ds=rand(Ls,ds);
Cs=rand(ds,Lt);
if (size(Dw,1)~=Ls)
    disp('ERROR: Number of rows in V and Dw must be the same');
    return
end
Dw = normalizeD(Dw); % Normalize
Ds = normalizeD(Ds); % Normalize
Rec=[Ds,Dw]*[Cs;Cw];
switch costfct % Calculate costfunction
    case 'ls'
        cost=[0.5*norm(V-Rec,'fro')^2+lambdas*sum(Cs(:))...
            +lambdaw*sum(Cw(:)),zeros(1,maxit)];
    case 'kl'
        cost=[sum(sum(V.*log((V+eps)./(Rec+eps))-V+Rec))...
            + lambdas*sum(Cs(:))+lambdaw*sum(Cw(:)),zeros(1,maxit)];
end
iter=1;
nuDs=[1,zeros(1,maxit)];
nuCs=[1,zeros(1,maxit)];
nuCw=[1,zeros(1,maxit)];
% Optimization loop
while 1
    switch costfct % update factorization
        case 'ls'
            [Cs,Cw,Ds,cost(iter+1),nuCs(iter+1),nuCw(iter+1),...
                nuDs(iter+1),accl] = updls(V,Cs,Cw,Ds,Dw,cost(iter),...
                nuCs(iter),nuCw(iter),nuDs(iter),lambdas,lambdaw,accl);
        case 'kl'
            [Cs,Cw,Ds,cost(iter+1),nuCs(iter+1),nuCw(iter+1),...
                nuDs(iter+1),accl] = updkl(V,Cs,Cw,Ds,Dw,cost(iter)...
```

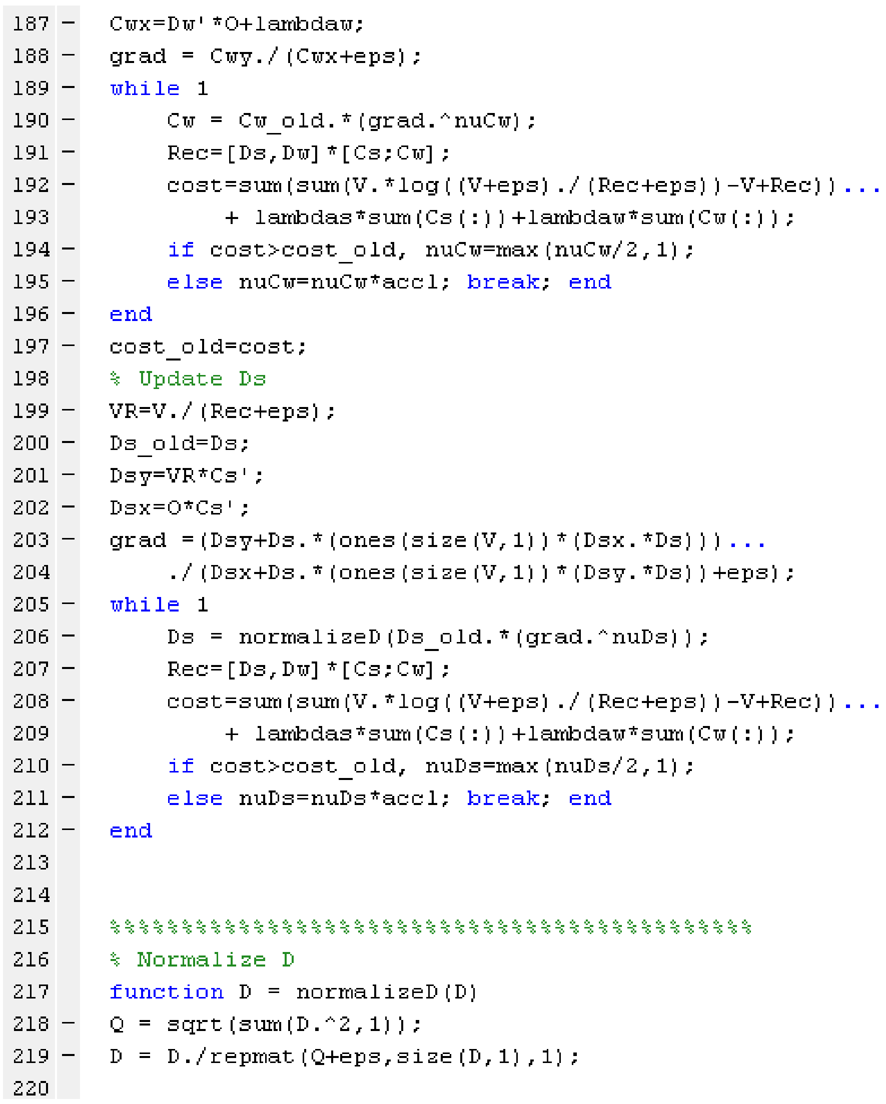


```
                ,nuCs(iter),nuCw(iter),nuDs(iter),lambdas,lambdaw,accl);
    end
    % Check for convergence
    if abs(cost(iter)-cost(iter+1))/cost(iter+1)<ccriteria
         % Check if acceleration parameter is too large
        if nuDs(iter+1) <= accl && nuCs(iter+1) <= accl && ...
                nuCw(iter+1) <= accl
            nuDs=nuDs(1:iter+1);
            nuCs=nuCs(1:iter+1);
            nuCw=nuCw(1:iter+1);
            cost=cost(1:iter+1);
            disp('NNMF has converged');
            break;
        else % set step size to 1 and run one more time
            nuDs(iter+1)=1;
            nuCs(iter+1)=1;
            nuCw(iter+1)=1;
        end
    end
    if iter<maxit % Check for number of iterations
        iter=iter+1;
    else
        disp('Maximum number of iterations reached');
        break;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Cs,Cw,Ds,cost,nuCs,nuCw,nuDs,accl] = updls(V,Cs,Cw,Ds,Dw,...
    cost_old,nuCs,nuCw,nuDs,lambdas,lambdaw,accl)
% Update Cs
Cs_old=Cs;
Csy=Ds'*V;
Csx=(Ds'*[Ds,Dw])*[Cs;Cw]+lambdas;
grad = Csy./(Csx+eps);
while 1
    Cs = Cs_old.*(grad.^nuCs);
    Rec=[Ds,Dw]*[Cs;Cw];
    cost=.5*norm(V-Rec,'fro')^2+lambdas*sum(Cs(:))+lambdaw*sum(Cw(:));
    if cost>cost_old, nuCs = max(nuCs/2,1);
    else nuCs = nuCs*accl; break;
    end
end
cost_old=cost;
% Update Cw
Cw_old=Cw;
Cwy=Dw'*V;
Cwx=(Dw'*[Ds,Dw])*[Cs;Cw]+lambdaw;
```

```matlab
grad = Cwy./(Cwx+eps);
while 1
    Cw = Cw_old.*(grad.^nuCw);
    Rec=[Ds,Dw]*[Cs;Cw];
    cost=.5*norm(V-Rec,'fro')^2+lambdas*sum(Cs(:))+lambdaw*sum(Cw(:));
    if cost>cost_old, nuCw=max(nuCw/2,1);
    else nuCw=nuCw*accl; break; end
end
cost_old=cost;
% Update Ds
Ds_old=Ds;
Dsy=V*Cs';
Dsx=[Ds,Dw]*([Cs;Cw]*Cs');
grad=(Dsy+Ds.*(ones(size(V,1))*(Dsx.*Ds)))...
    ./(Dsx+Ds.*(ones(size(V,1))*(Dsy.*Ds))+eps);
while 1
    Ds = normalizeD(Ds_old.*(grad.^nuDs));
    Rec=[Ds,Dw]*[Cs;Cw];
    cost=.5*norm(V-Rec,'fro')^2+lambdas*sum(Cs(:))+lambdaw*sum(Cw(:));
    if cost>cost_old, nuDs=max(nuDs/2,1);
    else nuDs=nuDs*accl; break; end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Cs,Cw,Ds,cost,nuCs,nuCw,nuDs,accl] = updkl(V,Cs,Cw,Ds,Dw,...
    cost_old,nuCs,nuCw,nuDs,lambdas,lambdaw,accl)
Rec=[Ds,Dw]*[Cs;Cw];
VR=V./(Rec+eps);
O=ones(size(V));
% Update Cs
Cs_old=Cs;
Csy=Ds'*VR;
Csx=Ds'*O+lambdas;
grad = Csy./(Csx+eps);
while 1
    Cs = Cs_old.*(grad.^nuCs);
    Rec=[Ds,Dw]*[Cs;Cw];
    cost=sum(sum(V.*log((V+eps)./(Rec+eps))-V+Rec))...
        +lambdas*sum(Cs(:))+lambdaw*sum(Cw(:));
    if cost>cost_old, nuCs = max(nuCs/2,0.1);
    else nuCs = nuCs*accl; break;
    end
end
cost_old=cost;
%Update Cw
VR=V./(Rec+eps);
Cw_old=Cw;
Cwy=Dw'*VR;
```



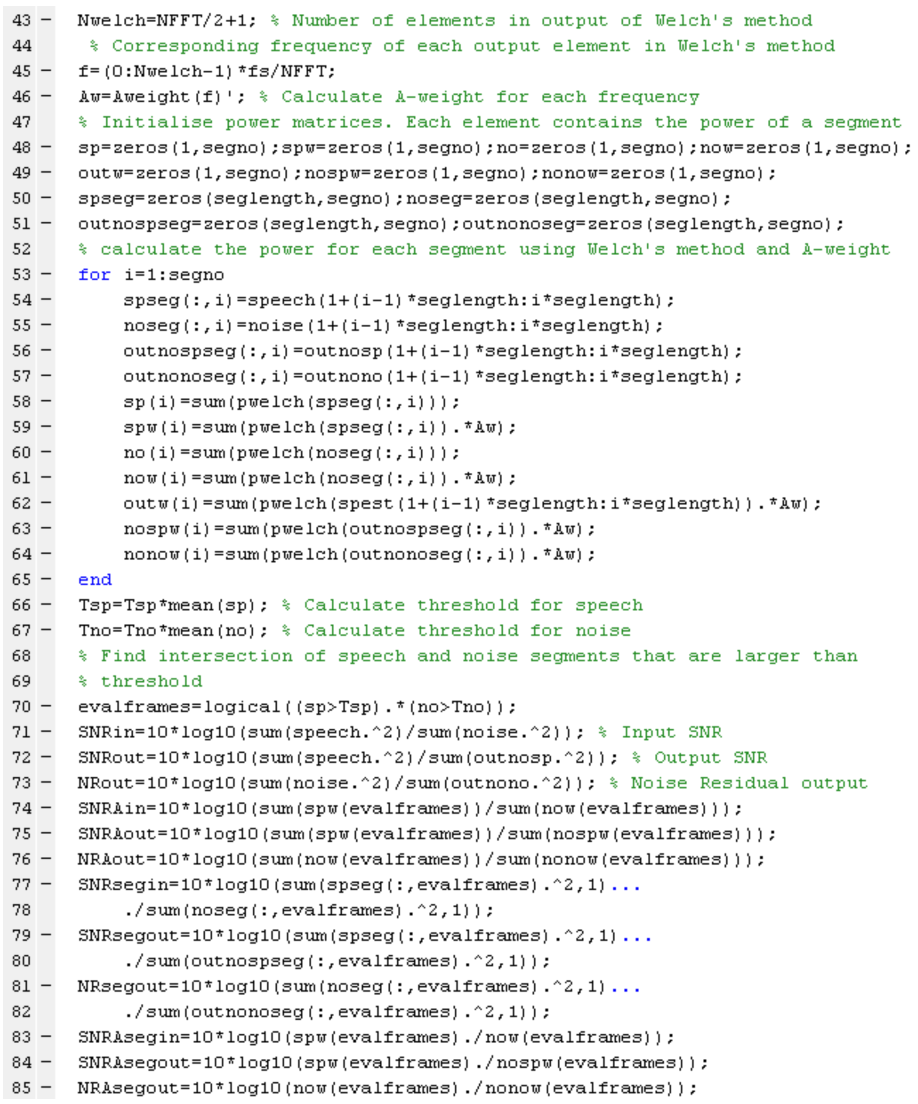
```
Cwx=Dw'*O+lambdaw;
grad = Cwy./(Cwx+eps);
while 1
    Cw = Cw_old.*(grad.^nuCw);
    Rec=[Ds,Dw]*[Cs;Cw];
    cost=sum(sum(V.*log((V+eps)./(Rec+eps))-V+Rec))...
        + lambdas*sum(Cs(:))+lambdaw*sum(Cw(:));
    if cost>cost_old, nuCw=max(nuCw/2,1);
    else nuCw=nuCw*accl; break; end
end
cost_old=cost;
% Update Ds
VR=V./(Rec+eps);
Ds_old=Ds;
Dsy=VR*Cs';
Dsx=O*Cs';
grad =(Dsy+Ds.*(ones(size(V,1))*(Dsx.*Ds)))...
    ./(Dsx+Ds.*(ones(size(V,1))*(Dsy.*Ds))+eps);
while 1
    Ds = normalizeD(Ds_old.*(grad.^nuDs));
    Rec=[Ds,Dw]*[Cs;Cw];
    cost=sum(sum(V.*log((V+eps)./(Rec+eps))-V+Rec))...
        + lambdas*sum(Cs(:))+lambdaw*sum(Cw(:));
    if cost>cost_old, nuDs=max(nuDs/2,1);
    else nuDs=nuDs*accl; break; end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Normalize D
function D = normalizeD(D)
Q = sqrt(sum(D.^2,1));
D = D./repmat(Q+eps,size(D,1),1);

```

B.4 Calculation of SNR and Related Energy Measures

```
function [SNRin,SNRout,NRout,SNRAin,SNRAout,NRAout,SNRsegin,...
    SNRsegout,NRsegout,SNRAsegin,SNRAsegout,NRAsegout]=...
    CalcSNR(noise,speech,spest,fs,seglength,Tsp,Tno)
% [SNRin,SNRout,NRout,SNRAin,SNRAout,NRAout,SNRsegin,...
%    SNRsegout,NRsegout,SNRAsegin,SNRAsegout,NRAsegout]=...
%    CalcSNR(noise,speech,spest,fs,seglength,Tsp,Tno)
%
% CalcSNR calculates different variations of SNR and Noise Residual
% measures.
% Optional arguments are seglength(=100), Tsp(=0.05) and Tno(=0.05).
%
% Input:
% noise - noise that speech is mixed with to produce input to algorithm
% speech - speech that noise is mixed with to produce input to algorithm
% spest - speech estimate output from algorithm
% fs - sample rate
% seglength - length of a segment used to calculate power in ms
% Tsp - Threshold of speech power for a segment in pct of mean
% Tno - Threshold of noise power for a segment in pct of mean
%
% Output:
% SNRin - Signal to noise ratio of the signal used as input to algorithm
% SNRout - Signal to noise ratio of signal output from algorithm
% NRout - Noise Residual of output from algorithm
% outputs with 'A' in the name is the same as corresponding output
% without 'A', but A-weighted.
% outputs with 'seg' in the name is the same as corresponding output
% without 'seg', but calculated for each segment.
%
% Written by: Kristian Timm Andersen, IMM DTU 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for input
if ((nargin<5)||isempty(seglength)),seglength=100;end
if ((nargin<6)||isempty(Tsp)),Tsp=0.05;end
if ((nargin<7)||isempty(Tno)),Tno=0.05;end
Lout=length(spest); % Length of output
outnosp=spest-speech; % output minus speech
outnono=spest-noise; % output minus noise
seglength=floor(fs*seglength/1000); % number of sample points in a segment
segno=floor(Lout/seglength); % Number of segments
% Number of FFT points used in Welch's method
NFFT=max(256,2^ceil(log2(seglength/4.5)));
```



```matlab
Nwelch=NFFT/2+1; % Number of elements in output of Welch's method
 % Corresponding frequency of each output element in Welch's method
f=(0:Nwelch-1)*fs/NFFT;
Aw=Aweight(f)'; % Calculate A-weight for each frequency
% Initialise power matrices. Each element contains the power of a segment
sp=zeros(1,segno);spw=zeros(1,segno);no=zeros(1,segno);now=zeros(1,segno);
outw=zeros(1,segno);nospw=zeros(1,segno);nonow=zeros(1,segno);
spseg=zeros(seglength,segno);noseg=zeros(seglength,segno);
outnospseg=zeros(seglength,segno);outnonoseg=zeros(seglength,segno);
% calculate the power for each segment using Welch's method and A-weight
for i=1:segno
    spseg(:,i)=speech(1+(i-1)*seglength:i*seglength);
    noseg(:,i)=noise(1+(i-1)*seglength:i*seglength);
    outnospseg(:,i)=outnosp(1+(i-1)*seglength:i*seglength);
    outnonoseg(:,i)=outnono(1+(i-1)*seglength:i*seglength);
    sp(i)=sum(pwelch(spseg(:,i)));
    spw(i)=sum(pwelch(spseg(:,i)).*Aw);
    no(i)=sum(pwelch(noseg(:,i)));
    now(i)=sum(pwelch(noseg(:,i)).*Aw);
    outw(i)=sum(pwelch(spest(1+(i-1)*seglength:i*seglength)).*Aw);
    nospw(i)=sum(pwelch(outnospseg(:,i)).*Aw);
    nonow(i)=sum(pwelch(outnonoseg(:,i)).*Aw);
end
Tsp=Tsp*mean(sp); % Calculate threshold for speech
Tno=Tno*mean(no); % Calculate threshold for noise
% Find intersection of speech and noise segments that are larger than
% threshold
evalframes=logical((sp>Tsp).*(no>Tno));
SNRin=10*log10(sum(speech.^2)/sum(noise.^2)); % Input SNR
SNRout=10*log10(sum(speech.^2)/sum(outnosp.^2)); % Output SNR
NRout=10*log10(sum(noise.^2)/sum(outnono.^2)); % Noise Residual output
SNRAin=10*log10(sum(spw(evalframes))/sum(now(evalframes)));
SNRAout=10*log10(sum(spw(evalframes))/sum(nospw(evalframes)));
NRAout=10*log10(sum(now(evalframes))/sum(nonow(evalframes)));
SNRsegin=10*log10(sum(spseg(:,evalframes).^2,1)...
    ./sum(noseg(:,evalframes).^2,1));
SNRsegout=10*log10(sum(spseg(:,evalframes).^2,1)...
    ./sum(outnospseg(:,evalframes).^2,1));
NRsegout=10*log10(sum(noseg(:,evalframes).^2,1)...
    ./sum(outnonoseg(:,evalframes).^2,1));
SNRAsegin=10*log10(spw(evalframes)./now(evalframes));
SNRAsegout=10*log10(spw(evalframes)./nospw(evalframes));
NRAsegout=10*log10(now(evalframes)./nonow(evalframes));
```

Bibliography

[1] S. Dupont H. Garudadri F. Grezl H. Hermansky P. Jain S. Kajarekar N. Morgan A. Adami, L. Burget and S. Sivadas. Qualcomm-icsi-ogi features for asr. *International Conference on Spoken Language Processing (INTERSPEECH)*, pages 21–24, 2002.

[2] R. Zdunek R. Kompass G. Hori A. Cichocki, S. Amari and Z. He. Extended smart algorithms for non-negative matrix factorization. *Artificial Intelligence and Soft Computing, International Conference on (ICAISC)*, pages vol. 4029, pp. 548–562, Jun 2006.

[3] J. A. Muriel M. Palou R. Aichner A. Liria, M. Masi and J. Samuelsson. Acoustic quality enhancement in mobile radio communications applications for public emergency services. 2003.

[4] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[5] S.F. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Trans ASSP*, pages vol. 2: 113–120, 1979.

[6] O. Cappe. Elimination of the musical noise phenomenon with the ephraim and malah noise suppressor. *IEEE Trans. on Speech and Audio Processing*, pages 345–349, 1994.

[7] S. Sra D. Kim and I. S. Dhillon. Fast newton-type methods for the least squares nonnegative matrix approximation problem. *Data Mining, Proceedings of SIAM Conference on*, 2007.

[8] R. J. Weiss D. P. W. Ellis. Model-based monoaural source separation using a vector-quantized phase-vocoder representation. *ICASSP06, IEEE International Conference on Acoustics, Speech and Signal Processing*, pages V/957–V960, 2006.

[9] Stephane Dupont. Noise compensation and the aurora tasks. *http://www.icsi.berkeley.edu/Speech/papers/qio/qio.zip*, 2002.

[10] J. Eggert. Sparse coding and nmf. *IEEE International Conference on Neural Networks - Conference Proceedings*, pages Vol. 4: 2529–2533, 2004.

[11] Y. Cao F. Ding G. Ding, X. Wang and Y. Tang. Speech enhancement based on speech spectral complex gaussian mixture model. *Acoustics, Speech and Signal Processing, 2005. ICASSP05*, pages I/165–I/168, 2005.

[12] P. Händel. Low distortion spectral subtraction for speech enhancement. *4th European Conference on Speech Communication and Technology*, pages p. 1549 – 1552, 1995.

[13] P. O. Hoyer. Non-negative sparse coding. *Neural Networks for Signal Processing, 2002 - Proceedings of the 2002 12th IEEE Workshop*, pages 557–565, 2002.

[14] ITU-R. Bs.1534 : Method for the subjective assessment of intermediate quality levels of coding systems. *http://www.itu.int/rec/R-REC-BS.1534/en*, Jan 2001.

[15] ITU-R. Bs.1116 : Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems. *http://www.itu.int/rec/R-REC-BS.1116-1-199710-I/e*, Oct 1997.

[16] ITU-T. P.862 : Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *http://www.itu.int/rec/T-REC-P.862/en.*

[17] ITU-T. P.800 : Methods for subjective determination of transmission quality. *http://www.itu.int/rec/T-REC-P.800-199608-I/en*, Aug 1996.

[18] S. Makino J. Cernak, S. Araki. Blind source separation based on a beamformer array and time frequency binary masking. *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007*, pages I–145–I–148, 2007.

[19] R. A. Goubran J. P. Dmochowski. Decoupled beamforming and noise cancellation. *IEEE Transactions on Instrumentation and Measurement*, pages 80–88, 2007.

[20] John H. L. Hansen John R., Jr. Deller and John G. Proakis. *Discrete-Time Processing of Speech Signals.* IEEE Press, 2000.

[21] J. B. Møller K. T. Andersen, B. Hansen. Signal processing for reduction of wind noise. *Polytechnical Midterm Project, IMM, Technical University of Denmark*, Jan 2006.

[22] M. Kuropatwinski and W. B. Kleijn. Estimation of the excitation variances of speech and noise ar-models for enhanced speech coding. *IEEE*, 2001.

[23] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature 401*, pages 788–791, 1999.

[24] Jae S. Lim and Alan V. Openheim. All-pole modeling of degraded speech. *IEEE Transactions of Acoustics, Speech and Signal Processing*, pages Vol ASSP–26, no. 3, 1978.

[25] Jae S. Lim and Alan V. Oppenheim. Enhancement and bandwidth compression of noisy speech. *Proceedings of IEEE vol. 67, no. 12*, pages 1586–1604, December 1979.

[26] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, pages vol. 19, pp. 2756–2779, 2007.

[27] K. Linhard and H. Klemm. Noise reduction with spectral subtraction and median filtering for suppression of musical tones. *Proc. of ESCA-NATO Workshop on Robust Speech Recognition for Unknown Communication Channels*, pages 159–162, 1997.

[28] R. Schwartz M. Berouti and J. Mokhoul. Enhancement of speech corrupted by acoustic noise. *Proc ICASSP*, pages 208–211, 1979.

[29] A. Langville V. Pauca M. Berry, M. Browne and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization.

[30] S. Makino M. Knaak, S. Araki. Geometrically constrained independent component analysis. *IEEE Transaction on Audio, Speech and Language Processing*, pages 715–726, 2007.

[31] J. Larsen M. N. Schmidt and F. Hsaio. Wind noise reduction using non-negative sparse coding. *IEEE International Workshop on Machine Learning for Signal Processing*, pages 431–436, aug 2007.

[32] S. Du P. Sajda and L. Parra. Recovery of constituent spectra using non-negative matrix factorization. *wavelets: Applications in Signal and Image Processing, Proceedings of SPIE*, pages Vol. 5207: 321–331, 2003.

[33] T. Aboulnasr Pan Qiongfeng. Efficient blind speech signal separation combining independent components analysis and beamforming. *Canadian Acoustics*, pages 118–119, 2007.

[34] Yoiti Suzuki Paul D. Schomer and Fumitaka Saito. Evaluation of loudness-level weightings for assessing the annoyance of environmental noise. *The Journal of the Acoustical Society of America Volume 110, Issue 5*, November 2001.

[35] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing.* Prentice Hall, 1996.

[36] Sam. T. Roweis. One microphone source separation. *Neural Information Processing System 13*, pages 793–799, 2000.

[37] M. N. Schmidt and M. Mørup. Nonnegative matrix factor 2-D deconvolution for blind single channel source separation. In *ICA2006*, apr 2006.

[38] Mikkel N. Schmidt and Hans Laurberg. Non-negative matrix factorization with gaussian process priors. *Submitted to Computational Intelligence and Neuroscience*, 2008.

[39] P. Sovka and P. Pollak. The study of speech/pause detectors for speech enhancement methods. *Proc of the 4th European Conference on Speech Communication and Technology*, pages 1575–1578, 1995.

[40] S. Sra and I. S. Dhillon. Nonnegative matrix approximation: Algorithms and applications. *University of Texas at Austin, Tech. Rep.*, Jun 2006.

[41] Jonas Samuelsson Sriram Srinivasan and W. Bastiaan Kleijn. Speech enhancement using a-priori information. 2003.

[42] G. Stoll and F. Kozamernik. Ebu listening tests on internet audio codecs. *EBU Technical Review*, Jun 2000.

[43] V. Hohmann T. Rohdenburg and B. kollmeier. Objective perceptual quality measures for the evaluation of noise reduction schemes. *9th International Workshop on Acoustic Echo and Noise Control. Eindhoven*, pages p. 169–172, 2005.

[44] Norbert Wiener. *Extrapolation, Interpolation and Smoothing of Stationary Times Series.* New York: Wiley, 1949.

[45] R. Zdunek and A. Cichocki. Non-negative matrix factorization with quasinewton optimization. *Artificial Intelligence and Soft Computing, International Conference on (ICAISC)*, pages vol. 4029, pp. 870–879, June 2006.