Digital Audio Restoration

Denoising phonograph recordings

Vegard Hella

TTT4900 SAM Master's Thesis

This document is available in PDF together with audio examples at:
`https://www.dropbox.com/s/ma6th27dhkiqiye/Master_Thesis_Vegard_Hella.zip`

Abstract

This master's thesis realize an audio noise reduction tool by use of digital signal processing. The tool is used to restore phonograph recordings. The recordings are restored on behalf of Ringve Museum, Norway's national museum of music and musical instruments. Sometimes the noise can be louder than the actual audio. In the view of a museum or library institution, this makes them less valuable as they are not presentable to the general public.

A common restoration environment will include multiple tools. We will only specialize in one of them reducing broadband, stationary and additive noise. This is often perceived as static hiss or buzz. To realize the tool we use the numerical computation environment MATLAB. In MATLAB the calculations are written using a high-level programming language with many embedded functions.

There are several established algorithms specializing in noise reduction of audio and speech. We will look at some basic and some complex algorithms that are based on the Short Time Fourier Transform (STFT). This technique slices the audio in short time frames to be able to analyze the local complex frequency spectrum. The noise reduction procedure compare the audio spectrum with its estimated noise spectrum to calculate an attenuation at each frequency. The attenuated signal is transformed back into time domain. Some of the algorithms are based on the Wiener filter or AR-modeling. The program will include a user interface with selectable algorithms and parameters. In old recordings a certain level of noise may be wanted to preserve authenticity. Thus a noise floor generator will be implemented.

Some necessary theory of digital signal processing will be given, but some general knowledge will be required. The noise reduction theory is presented before the realization and program flow is explained. A listening test will be conducted. Audio examples are used to illustrate the general results, and the development process, results and further work is discussed.

The program gave better results than one of the commercial available softwares. Another important result is that the stationary property is a poor approximation. The phonograph noise exhibits periodical properties with longer time periods than used in the short time transform. A model incorporating this feature should be implemented.

Sammendrag

Masteroppgaven omhandler støyreduksjon av lydopptak ved bruk av digital signalbehandling. Hovedmålet er å lage et verktøy som kan restaurere digitaliserte fonografruller. På opptakene er bakgrunnsstøyen ofte høyere enn selve lydmaterialet. Dette gir opptakene liten verdi, da de ikke kan presenteres for et publikum. Fonografrullene er restaurert på veine av Ringve museum, Norges nasjonale museum for musikk og musikalske instrumenter.

Vi antar at støyen er stasjonær, bredbåndet og additiv. Dette oppfattes gjerne som en konstant summing eller during. Verktøyet vil kun være et av flere nødvendige i et fullstendig restaureringsmiljø. For å realisere verktøyet bruker vi MATLAB, et matematikkprogram som bruker et høynivå programmeringsspråk til å manipulere numeriske data.

Det finnes en rekke etablerte algoritmer og artikler på området. Vi skal se på flere algoritmer av forskjellig kompleksitet, men kun konsentrere oss om en bestemt gruppe. Denne gruppen benytter seg av korttids Fouriertransform. Her partisjoneres lydfilen i korte tidsintervaller for å analysere det instantane frekvensspektrumet. Ved å sammenligne lydspektrumet med det estimerte støyspektrumet kan vi bestemme en dempning ved hver frekvens. Noen av algoritmene er basert på Wiener filteret og AR-modellering. Programmet vil ha et grensesnitt med valg av algoritmer og parametre. I gamle opptak kan det være ønskelig å ivareta et støygulv, en slik funksjon vil være viktig å implementere.

Det mest nødvendige av teori om digital signalbehandling blir gjennomgått, men en generell kompetanse er forventet. Teorien bak støyreduksjon presenteres før den realiseres i programkode og programflyten beskrives. En generell lyttetest blir også gjennnført. Lytteeksempler illustrerer de generelle resultatene fra prosessen, og i etterkant diskuterer vi løsningene fra utviklingsfasen, resultatene og fremtidige forbedringer.

Verktøyet viste seg å gi gode resultater på nivå med kommersielle alternativ. Et annet viktig resultat er at antagelsen om at støyen er stasjonær er en dårlig tilnærmelse. Fonografrullene har en støy med stor varianse og periodiske trekk som varer lenger enn korttidstransformen. En modell tilpasset disse egenskapene burde utvikles for bedre verktøyet.

Contents

List of Figures

Abbreviations

AMAP Approximate Maximum A Posteriori Spectral Amplitude Estimator

AR Autoregressive

DFT Discrete Fourier Transform

EMSR Epharim-Malah Suppression Rule

FFT Fast Fourier Transform

JMAP Joint Maximum A Posteriori Spectral Amplitude and Phase Estimator

MAP Maximum A Posteriori

MMSE Minimum Mean Square Error

MMSEP Minimum Mean Square Error Spectral Power Estimator

OLA Overlap-Add

RMS Root Mean Square

STFT Short-Time Fourier Transform

STSA Short-Time Spectral (Amplitude) Attenuation

Preface

This thesis was done at the Norwegian University of Science and Technology (NTNU) at the Department of Electronics and Telecommunications (IET), in the first half of 2013. It is an individual master's thesis, course code TTT4900 (30p), specializing in a subject within signal processing, acoustics and media (SAM). Thanks to associate professor Jan Tro at the acoustics group for the given task, and curator Mats Krouthén at Ringve Museum.

Vegard Hella
NTNU, Trondheim. June 9, 2013.

1 Introduction

1.1 Outline

In this paper we will look at noise reduction by digital signal processing. The goal is to denoise old audio recordings, hence we start to look at established speech enhancement- and audio restoration algorithms. To realize the restoration tool we use the numerical computation environment, MATLAB [1]. MATLAB feature a high-level programming language and several functions to visualize data. The premises of our tool are that the noise is considered *stationary*, *broadband* and *additive*. Other tools are needed to perform impulsive noise removal, pitch restoration or saturation reduction.

The first section presents related work and a brief introduction to conservation- and digitalization of phonograph recordings. This gives us some information on the target audience. Next, a chapter on necessary signal processing theory and in-depth noise reduction theory. Some basic knowledge of digital signal processing is required. The procedure chapter explains the program flow, some restoration guidelines and a listening test. The result section describes intermediate results and the noise reducer performance. Next, we discuss the development, results and further work. At last a short conclusion of the project.

1.2 Background

This thesis is a continuation of a semester project in digital audio restoration, which focused on impulsive noise/click reduction [2].

To obtain some old recordings I contacted *Ringve*, Norway's national museum of music and musical instruments, located in Trondheim. I was given several phonograph recordings. These recordings were digitalized at the *National Library of Norway*, located in Mo i Rana. The cylinders are from the 1900's. The audio material needs restoration and is supposed to be featured at Ringve Museum towards the end of the semester. This gives an excellent incentive to realize a noise reduction tool.

The cylinder phonograph was invented by Thomas Edison in 1877 and commercially available by the late 1880's, and throughout the 1910's [3]. It was a mechanical device and used wax cylinders as the recording medium. The sound was imprinted by a steel needle, which vibrated by acoustical energy focused through a horn. The cylinder rotated either by a hand driven wrench or a spring system. The effective frequency range was approximately 160-2000 Hz [4].

1.3 Related work

Similar tools on noise reduction are developed by DARTECH Inc. and CEDAR Ltd. and is discussed in [2]. DARTECH's de-hissing tool is used for comparison [5].

The first algorithms were developed in the late 70's and early 80's to digitally enhance speech perception. They are based around the Short-Time Fourier Transform (STFT). This technique analyze the audio by extracting short time frames and transforming them into the frequency domain. Throughout the 90's more papers have been released focusing on audio restoration. There are primarily two popular papers presenting two different algorithms. The first is a Wiener filter based approach given in the book *Digital Audio Restoration*

- a statistical model based approach by *Godsill* and *Rayner* [4]. The second is a more complex algorithm presented in the article *Elimination of the Musical Noise Phenomenon with the Ephraim and Malah Suppressor* by *Oliver Cappé* [6]. There exist other algorithms not based on the STFT, but they are not discussed in this thesis. Those solutions are based on the Wavelet Transform [7], and the recently suggested block thresholding method [8].

     An earlier, pre-digital noise reduction technique is the noise gate. It attenuates the signal unless it is above a certain threshold. This cancel the noise in silent transitions and works best if the noise is masked by audio otherwise. A more practical noise gate use overlapping band pass filters with individual thresholds to further remove noise in live audio content [9].

1.4   Conservation and digitalization

The conservation and digitalization of phonograph records are typically handled by a library institution for the sake of preserving cultural heritage. Another important aspect of conservation is availability to the public. As I have learned from Ringve museum, old objects are most valuable when in reach by an audience. Digitalization is essential, as it is limitless in form of availability and preservation. A known fact is that people are shy of distortion and noise, as well as bad resolution. Thus as noise increases the availability per se decreases. This could be said about distortion and bandwidth as well.

     My initial thought was that the digital recordings were made with a microphone placed in front of an old phonograph. This is not the case. The digital recordings are made with a modern turntable pickup mounted on a specialized playback system. Either the pickup is mounted on an extra long tone arm to give a small change in angle between the needle and groove, or a device tracks the needle parallel to the cylinder. The tracking is crucial as a force in the wrong direction could damage the fragile imprint. A modern lightweight pickup is used not only to minimize wear, but give a better frequency response [10].

     The pickup consists of a moving magnet and coil producing a small electrical signal. The signal is amplified in the analog domain. The preamplifier may need an equalizer. The standard RIAA preamplifier for LP records use a fixed equalization curve [11], and is not suitable for cylinder recordings. At last the signal is recorded through an ADC.

2   Theory

2.1   Digital signal processing

2.1.1   Short-Time Fourier Transform

The STFT is a signal processing technique used to determine the phase and frequency of non-stationary time signals [12]. We wish to look at the spectral content as the signal varies through time. This is done by chopping up the signal in small sections or short *time frames*, analyzing each frame as a stationary part using the *Fourier transform*. The STFT is invertable and allow us to manipulate the signal in frequency domain.

The discrete STFT of the signal $x(n)$ is

$$X(p,k) = \sum_{l=0}^{L-1} g_{win}(l)x(pM+l)e^{-j2\pi kl/L} \quad k = 0,1,2,\ldots,L-1 \tag{1}$$

where p is the frame number, starting at zero [4]. In frame p, k is the discrete frequency bin and l is the sample index. L is the total number of samples, or *frame size*. M is the *step size* or number of samples between successive frames, and is typically smaller than L to give an overlap of $L - M$ samples. $g_{win}(l)$ is a *window function*.

When using the Discrete Fourier Transform (DFT) on a truncated signal, the limits become a part of the signal's frequency response. If the frame acts as a rectangular window the abrupt transition from zero to the first sample will generate undesirable frequency components. These components can interfere with components from the actual signal. If the frame is viewed through a smoother window, as the *Hamming* window in figure 1, the transition becomes less abrupt and generates less undesirable frequency components. On the other hand there are less accurate points to determine the signal frequency, hence we loose some *frequency resolution*, but gain *dynamic range*. Not to loose so much information at the tapered ends of the window, we overlap successive frames. A greater description of window functions are given in [13].

We invert the STFT using an Overlap-Add (OLA) method. First each frame has to be inverse transformed.

$$x_p(l) = g_{corr}(l) \sum_{k=0}^{L-1} X(p,k)e^{j2\pi kl/L} \quad l = 0,1,2,\ldots,L-1 \tag{2}$$

$x(n)$ is now recognized as the chopped up overlapping sections abbreviated $x_p(l)$. $g_{corr}(l)$ is a gain correction function to reverse the amplitude modifications done by the overlap and windowing.

$$g_{corr}(l) = \begin{cases} \dfrac{1}{g_{win}(l)} & M > L/2 \\[3ex] \dfrac{M}{\sum\limits_{l=0}^{L-1} g_{win}(l)} & M \leq L/2 \end{cases} \tag{3}$$

With less than 50% overlap the inverse window is used. This is only applicable for the Hamming window or other non zero tapered windows. With 50% or more overlap the signal has been amplified (almost uniformly depending on window type) by the window integral, or discrete sum, divided by the step size. The inverse is used to scale down the signal. This is also valid for zero tapered windows. Another gain correction function is described in [4].

To regain the complete signal the overlapping sections are added together.

$$x(n) = \sum_p x_p(n - pM) \qquad 0 \leq n - pM < L - 1 \tag{4}$$

2.1.2 AR-modeling

The digital filter is essential in digital signal processing. As a signal is sent through the filter, it is manipulated to enhance certain aspects of the signal. The filter behavior is described by a transfer function with a set of *filter coefficients*. The top coefficients, b_i represent *zeroes* in the transfer function, and the bottom ones, a_i represent *poles*. The digital filter transfer function as found in [14]:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}} \tag{5}$$

An *autoregressive process* generates a signal where the current values correlate with the previous values. Autoregressive (AR)-modeling tries to model such a process by an all pole Infinite Impulse Response (IIR) filter, which exited by white noise produces an output similar to the autoregressive process [15]. The model is represented by the filter coefficients and the white noise variance. The AR-model is defined as

$$x(n) = -\sum_{i=1}^{\rho} a_i x(n - i) + w(n) \tag{6}$$

where $x(n)$ is the output signal, a_i the AR-model coefficients, ρ the model order, and $w(n)$ a white noise signal with variance σ_e^2 [16]. A higher order gives a better model, but demands more computation. Solving for the coefficients, often called the autocorrelation method, involves solving the *normal equations* also called *Yule-Walker equations*. A complete solution also involves finding an estimate of the white noise variance.

2.1.3 Wiener-filter

The Wiener filter suppress additive noise by comparing the noisy signal with an estimated noiseless signal [17]. The filter is derived from the performance criterion of Minimum Mean Square Error (MMSE), and takes the following form in the frequency domain:

$$H(\omega) = \frac{\mathcal{S}_X(\omega)}{\mathcal{S}_X(\omega) + \mathcal{S}_D(\omega)} \tag{7}$$

where ω is the continuous frequency [4]. $\mathcal{S}_X(\omega)$ is the *energy density spectrum* of the noiseless signal, and $\mathcal{S}_D(\omega)$ is the energy density spectrum of the noise.
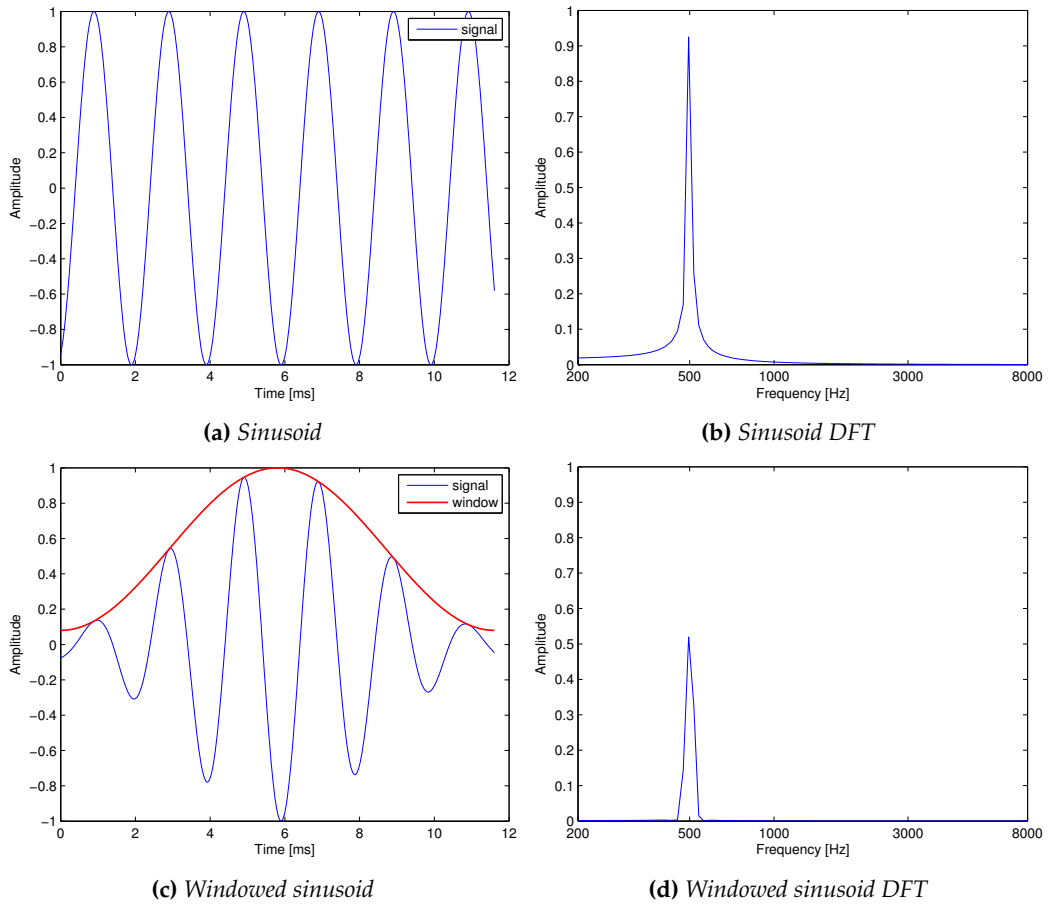
(a) *Sinusoid*

(b) *Sinusoid DFT*

(c) *Windowed sinusoid*

(d) *Windowed sinusoid DFT*

Figure 1: *Hamming windowing a 500 Hz tone.*

A basic estimate of a energy density spectrum is

$$\mathcal{S}_X(\omega) = |X(\omega)|^2 \tag{8}$$

For reference an estimate of the discrete *power density spectrum* is

$$\mathcal{P}_X(k) = \frac{1}{N}|X(k)|^2 \tag{9}$$

where N is the length of the power spectrum [16].

2.2 Stationary noise reduction

The noise reduction procedure use the STFT. The signal- and noise spectrum is calculated using the same frame size. Based on the ratio between the spectrums a *suppression rule* determines the amount of attenuation of each frequency bin. The resulting gain is applied to the complex noisy signal. Since we are multiplying the complex signal with a factor, the phase remains as the original. This could introduce some phase-modulating effects in the restored output [4]. The noiseless audio frames are transformed back into time domain, and spliced together using the OLA method. Such a noise reduction procedure is often referred to as Short-Time Spectral (Amplitude) Attenuation (STSA) [18].
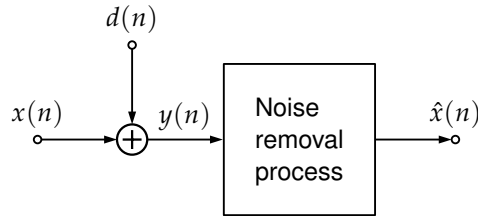
Figure 2: *Noise reduction signals.*

By figure 2, $y(n)$ is the known, observable noisy signal. The noise $d(n)$ can be observed when $x(n)$ is zero (no audio present). This is referred to as the *noise print* and is used to estimate the noise spectrum. $\hat{x}(n)$ is the estimated noiseless audio.

2.2.1 Basic suppression rules

There are three basic suppression rules. All very similar and derived from the Wiener filter [4].

The Wiener filter solution is given in equation 7. As $\mathcal{S}_X(\omega)$ is unknown it is substituted with the related noisy spectrum, $\mathcal{S}_Y(\omega)$.

$$\mathcal{S}_Y(\omega) = \mathcal{S}_X(\omega) + \mathcal{S}_D(\omega) \tag{10}$$

The principle is transformed into the discrete domain by substituting the continuous frequency variable, ω with the discrete frequency variable, k. This gives a pseudo-Wiener

filter solution. Combining 7 and 10 gives us the following suppression rule

$$G_{wiener}(k) = \begin{cases} \dfrac{\mathcal{S}_Y(k) - \mathcal{S}_D(k)}{\mathcal{S}_Y(k)} & \mathcal{S}_Y(k) > \mathcal{S}_D(k) \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

The rule is only valid when the difference between the signal- and noise spectrum is positive, otherwise the gain must remain zero.

Another rule is spectral subtraction. Here the noise amplitude spectrum is subtracted from the signal amplitude spectrum.

$$G_{ssub}(k) = \begin{cases} \dfrac{|Y(k)| - |D(k)|}{|Y(k)|} & |Y(k)| > |D(k)| \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

The third variant is power subtraction. Here the gain is the square root of the Wiener solution.

$$G_{psub}(k) = \begin{cases} \sqrt{\dfrac{\mathcal{S}_Y(k) - \mathcal{S}_D(k)}{\mathcal{S}_Y(k)}} & \mathcal{S}_Y(k) > \mathcal{S}_D(k) \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

There are small differences in the output produced by each suppression rule. The spectral subtraction suppress the most, and the power subtraction the least. The Wiener solution is somewhere in between. A gain curve describing the relation can be seen in [4].

2.2.2   Noise spectrum estimation

There is little in the articles describing how to estimate the noise spectrum. The obvious way is to compute the spectrum from the raw DFT components.

The noise print may vary over a longer period than one frame. To get a good estimate the noise print is partitioned with the same window function, frame- and step size as the noise reduction process. To estimate the spectrum from the DFT components the Fourier transform is done as well. The spectrums are then averaged. When the STFT with overlap is used to compute an averaged power spectrum, it is often called *Welch's method* [16].

Typical averaging methods are the *arithmetic mean*, *Root Mean Square (RMS)* and *maxima*.

$$\hat{D}_{mean}(k) = \frac{\sum\limits_p D(p,k)}{P} \quad p = 0, 1, 2, \ldots, P-1 \quad (14)$$

where P is the total number of frames.

$$\hat{D}_{rms}(k) = \sqrt{\frac{\sum\limits_p D(p,k)^2}{P}} \quad (15)$$

$$\hat{D}_{max}(k) = \max_p D(p,k) \quad k = 0, 1, 2, \ldots, L-1 \quad (16)$$

Alternatively the spectrum can be estimated using an AR-model instead of the Fourier transform. To get the spectrum of each frame we calculate the frequency response of the

digital filter in equation 5, equipped with the AR-model coefficients from equation 6 and the white noise variance. The order of filter coefficients control the spectrum precision. Fewer coefficients will describe the spectrum more coarsely, possibly carving out more noise. The complex frequency response is given by

$$H(e^{j\omega}) = \frac{\sqrt{\sigma_e^2}}{1 + \sum\limits_{i=1}^{\rho} a_i e^{-j\omega i}} \tag{17}$$

2.2.3 The Ephraim-Malah Suppression Rule

The basic suppression rules can result in some annoying artifacts called *musical noise*. This occurs when the estimated noise spectrum mismatch the noisy data, leaving peaks at random frequencies. To suppress this we need a smoothing effect in our suppression rule. The ordinary Epharim-Malah Suppression Rule (EMSR) was derived in [19] and further developed in [18]. In recent years others have continued the development [6][20]. The smoothing effect is implemented by analyzing both the current and previous restored frame, combining them in a feedback topology. The EMSR is derived with a performance criterion of MMSE, based on modeling speech and noise as independent random Gaussian variables. The restored phase is shown to be the same as in the noisy audio. We skip the derivation and head straight for the good stuff.

The EMSR as defined by Cappé [6]:

$$G_{emsr}(k) = \frac{\sqrt{\pi}}{2} \sqrt{\left(\frac{1}{1 + R_{post}(k)}\right) \left(\frac{R_{prio}(k)}{1 + R_{prio}(k)}\right) \cdot M\left[(1 + R_{post}(k)) \left(\frac{R_{prio}(k)}{1 + R_{prio}(k)}\right)\right]} \tag{18}$$

R_{post} is the *a posteriori* SNR at each frequency bin. In other words the SNR of the latest frame. R_{prio} is the *a priori* SNR, weighted between the previous and latest frame. $M(\theta)$ stems from a confluent hypergeometric function [18], described in equation 22.

$$R_{post}(k) = \frac{|Y(p,k)|^2}{|D(k)|^2} - 1 \tag{19}$$

$$R_{prio}(k) = (1 - \alpha)P\left(R_{post}(k)\right) + \alpha \frac{|G(p-1,k)Y(p-1,k)|^2}{|D(k)|^2} \tag{20}$$

p denotes the frame number. The definition by Cappé introduce a new parameter, α ($0 \leq \alpha \leq 1$). The parameter acts as a weighting factor, controlling the amount of feedback. As α approaches 1 the SNR of the previous restored frame dominates. $P(x)$ ensures zero or positive output.

$$P(x) = \begin{cases} x & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

$$M(\theta) = e^{-\theta/2}\left[(1 + \theta)I_0 + \theta I_1\right] \tag{22}$$

I_0 and I_1 are the modified Bessel functions of first kind, of zero and first order. Their formulas are presented in [21].

The EMSR is derived with the assumption that the signal is always present. The suppression rule can be further expanded to include the *probability of signal absence*, $q(k)$. This includes a new definition of R_{prio}.

$$R'_{prio}(k) = \frac{R_{prio}(k)}{1 - q(k)} \tag{23}$$

where the new R'_{prio} replace all the previous instances of R_{prio}. A likelihood ratio, Λ is defined to modify the original EMSR.

$$\Lambda(k) = \frac{\mu(k)e^{\theta}}{1 + R'_{prio}(k)} \tag{24}$$

where μ is defined as

$$\mu(k) = \frac{1 - q(k)}{q(k)} \tag{25}$$

Finally, the modified EMSR is

$$G'_{emsr}(k) = \frac{\Lambda(k)}{1 + \Lambda(k)} G_{emsr}(k)\Big|_{R_{prio}=R'_{prio}} \tag{26}$$

2.2.4   EMSR alternatives

Due to the complexity of computing Bessel functions, alternatives to the EMSR have been purposed [22]. They use the same modeling assumptions and SNR definitions as the original EMSR. Here they are fitted with the same SNR definitions as above. The phase is shown to be the same as in the noisy audio. The frequency variable k is omitted for simplicity.

Joint Maximum A Posteriori Spectral Amplitude and Phase Estimator (JMAP) is a solution with a Maximum A Posteriori (MAP) performance criterion to estimate the optimal phase and amplitude.

$$G_{jmap} = \frac{R_{prio} + \sqrt{R_{prio}^2 + 2\left(1 + R_{prio}\right)\dfrac{R_{prio}}{1 + R_{post}}}}{2\left(1 + R_{prio}\right)} \tag{27}$$

Approximate Maximum A Posteriori Spectral Amplitude Estimator (AMAP) uses an approximation of the Bessel function, together with the MAP criterion to estimate the optimal amplitude.

$$G_{amap} = \frac{R_{prio} + \sqrt{R_{prio}^2 + \left(1 + R_{prio}\right)\dfrac{R_{prio}}{1 + R_{post}}}}{2\left(1 + R_{prio}\right)} \tag{28}$$

Minimum Mean Square Error Spectral Power Estimator (MMSEP). The same MMSE criterion as in the EMSR, but used to estimate the optimal power.

$$G_{mmsep} = \sqrt{\left(\frac{R_{prio}}{1 + R_{prio}}\right)\left(\frac{1 + \theta}{1 + R_{post}}\right)} \tag{29}$$

Again θ is

$$\theta = \left(1 + R_{post}\right) \frac{R_{prio}}{1 + R_{prio}} \tag{30}$$

Gain curves comparing the EMSR, JMAP, AMAP and MMSEP can be seen in [22]. The JMAP and MMSEP are very close to the EMSR.

3   Procedure

3.1   The De-Hisser tool

This section describes the realization of the noise reduction theory as a computer program. The program is specialized to reduce noise in WAV-files and aims to be one link in a toolchain to perform audio restoration. A block diagram of the *De-Hisser* tool is shown in figure 3.

3.1.1   Top level script

The top level MATLAB script is found in appendix A.1. The script features an interface to select files and parameters, and runs the subroutines (functions) to estimate the noise spectrum and perform noise suppression. It also controls a low pass filter and a noise floor generator.

The noise print is extracted via the input file by entering a start and stop sample, or by loading a separate file. The input feature a trim option, primarily to test short excerpts to determine parameter values. The parameter values are attached to the output filename.

Three strings select the suppression rule, noise spectrum estimation- and averaging method. The strings are passed on to the underlying functions. There are seven selectable suppression rules: spectral subtraction, Wiener solution, power subtraction, EMSR, JMAP, AMAP and MMSEP. There are three spectrum estimation methods: Fast Fourier Transform (FFT), AR-modeling and Welch's method. And there are three spectrum averaging methods: maxima, mean and RMS.

Further the interface presents the STFT parameters *frame size*, *step size* and *window function*. The sizes are given by the total number of samples, and the frame size is preferred to be a power of two, to avoid bad signal partitioning. The step size is selected as a fraction of the frame size. The window function is selected by a string. Further the string is evaluated as code, executing the corresponding MATLAB window function, where the frame size is used as an argument setting the window length. This setup allow us to choose among seventeen window functions. A list is displayed by typing 'help window' in the command window. The STFT parameters are used by the *spectrum estimator*- and *noise suppressor* functions.

Next there are parameters adjusting the *noise reduction gain*, *noise floor gain* and *AR-model order*. The estimated noise spectrum is amplified or attenuated by the noise reduction gain. A noise floor is generated by adding a fraction of the *residue* to the output. The residue is the difference between the original- and noiseless signal. The fraction is set with the noise floor gain, if set to zero no noise is added. The AR-model order is used by the spectrum estimation function. In addition to all the other parameters the EMSR, JMAP, AMAP and MMSEP have the *frame weight* parameter. The EMSR is also controlled by the *signal absence* parameter.

At last a Butterworth low pass filter can be enabled together with a cutoff frequency. The filter is fixed to fourth order. The frequency is given in Hertz. The low pass filtering is performed by a secondary function before the noise floor is added.

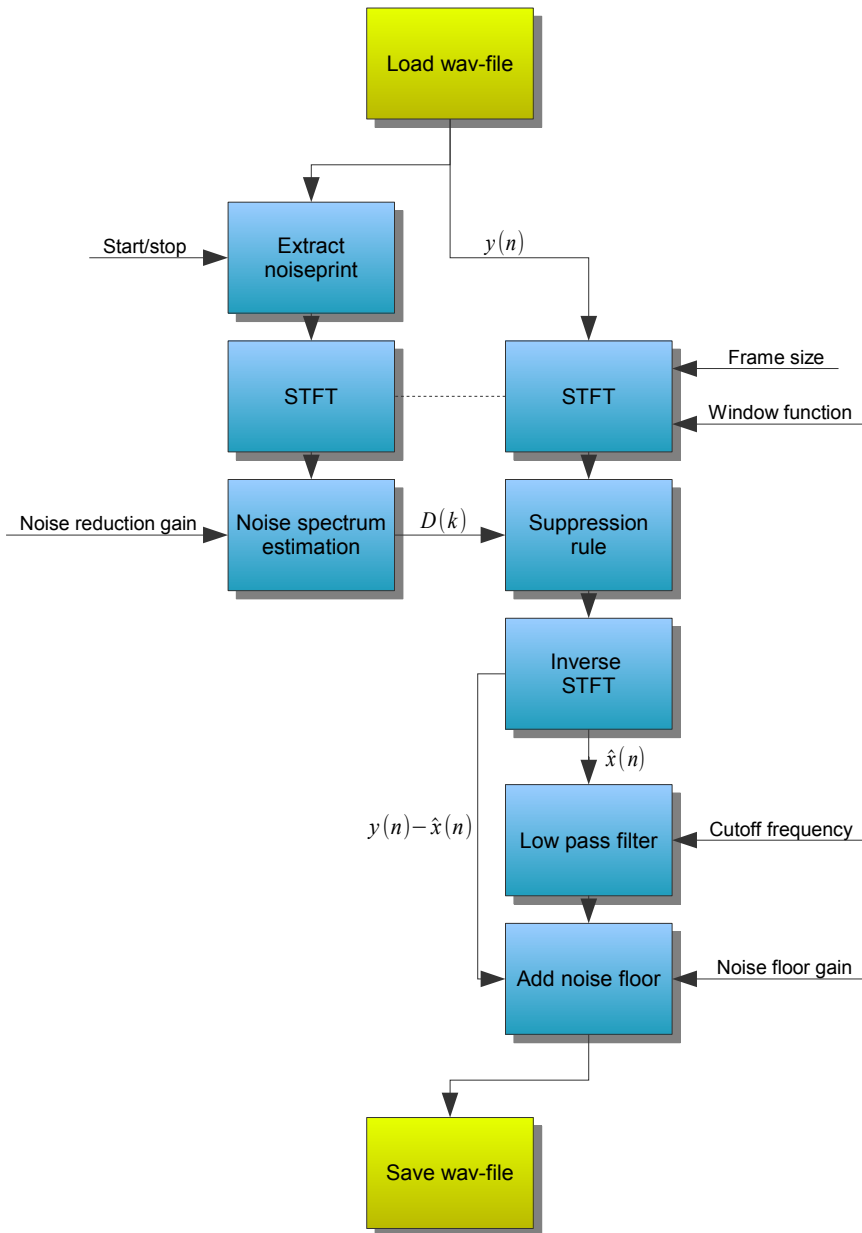Following the parameters are settings to play the selected noise print, original file, the

Figure 3: *De-hisser tool, flowchart.*

restored output and the residue. By listening to the residue we can determine the amount of good audio that is removed. At the end of the main script the output is saved with the same resolution as the input.

3.1.2   STFT and windowing

The STFT is computed sequentially and the frames are estimated as the signal progresses. This loop is present in both the spectrum estimator and the noise suppressor function. In each function only the window function and step size are taken as arguments. The frame size is set as the window function length.

At the first and last frame there is an uneven distribution of the window function, due to the overlap. A solution to this was developed by zero padding the signal. One empty frame is padded to each end of the signal. To make a whole number of frames fit, the number of frames are rounded up and the lack of samples are zero padded. To be able to access parts of an yet to be estimated signal it has to be preallocated in memory with correct length. After the signal have been partitioned, it is multiplied by the window function in time domain.

Next, the DFT is computed using the common FFT algorithm. The FFT is not scaled down by it's length, although doing so is a common practice to get the correct amplitude. This is default by MATLAB and not necessary since we are dealing in ratios. It is neither truncated to a single sided spectrum. This makes it easier to do the inverse transform. The signal is now ready to be processed in the frequency domain.

3.1.3   The spectrum estimator

The spectrum estimator function can be found in appendix A.2. After the STFT have generated a frame a *switch case* jumps to the selected estimation- and averaging method. If a FFT estimate is chosen all there is to compute is the FFT of the partitioned frame. The noise spectrum output is kept complex, not to discard any information that might become useful.

In the case of an AR-modeled spectrum the coefficients are calculated using MATLAB's *aryule* function. The frame and model order are required input arguments. The *aryule* function use the Levinson-Durbin recursion to solve the Yule-Walker equations. The coefficients are normalized by the first coefficient [23], as in equation 5. The variance of the white noise input is also given by *aryule*. Solving the AR-model of only zeros results in coefficients of infinite value, this corrupts our estimate, therefore a control sequence skips the zero-valued frames. The complex frequency response of the AR-model is computed with MATLAB's *freqz* function. Since the FFT gives a two-sided spectrum, the response is calculated around the whole unit circle, giving a two-sided spectrum. The number of points calculated is equal to the frame size. To get the amplitude similar to the FFT, the frequency response is amplified by the square root of both the variance and signal length [24].

MATLAB's own Welch's method function, *pwelch* can be used. This bypasses the STFT and averaging process of the spectrum estimator function. In theory it should produce a similar result to the FFT and is implemented out of curiosity. Welch's method only produce a mean averaged spectrum. The *pwelch* function takes the noise print, window function and number of overlapping samples as input arguments. The option 'twosided' is selected with same reason as in the AR-model case. The resulting spectrum is of the same length as the window function. The output is not a complex spectrum, but an unbiased estimated power spectrum. To scale the result to a similar amplitude as the FFT, the output is scaled

by $\sum_{k} g_{win}(k)^2 \, / \, ||g_{win}(k)||^2$, where $||\cdot||$ denotes the second norm, or Pythagorean theorem [25].

3.1.4 The noise suppressor

The noise suppressor function is executed from the top level after the spectrum estimation, and is shown in appendix A.3. Again a switch case is used to select the suppression rule.

If one of the basic suppression rules is selected, it is first ensured to get only positive resulting arguments, otherwise the gain remains zero. The condition to prove a positive result is simplified by only comparing the amplitude. All the suppression rules result in a frequency gain vector that is applied to the complex signal.

If the Ephraim-Malah based suppression rules are selected, separate functions are executed. Common to all are the calculation of the a posteriori and a priori SNRs. First there is a divide by zero protector. All the frequency bins equal to zero at the input are excluded from the calculations, and set to zero in the gain vector. As the a priori SNR is calculated, the MATLAB function *max* corresponds to the positive output function of equation 21. In the case of JMAP and AMAP the gain vector is ready to be calculated and returned to the noise suppressor function. In the case of MMSEP, the θ variable is calculated before the gain vector.

The EMSR is more complex. Before θ is calculated the a priori SNR is modified by the probability of signal absence parameter. If the parameter is set to zero, the SNR remains unchanged. The exponential growth of θ can result in overflow and infinite values. By trial the MATLAB exponential function seem to overflow above 700. It is the a posteriori SNR that gives θ high values, thus it is truncated when values reach above 700. After θ and the Bessel functions are computed the hypergeometric function and gain vector is calculated. If the signal absence probability is above zero the gain vector is modified as stated in the theory section.

After processing in the frequency domain, the window gain correction is applied to the inverse transformed frame. At last the restored frame is added to the preallocated estimated signal. When the noiseless signal is complete the zero padding is removed.

3.2 Using the De-Hisser

From Ringve I got 20 recordings in various quality. A few of them were not recorded by the National Library. The first task was to sort out a few phrases that would only need de-hissing and had usable noise prints. Both song, speech and instrumentals with many and few instruments were picked out. Recordings that would need click- and/or saturation reduction were excluded from the project. *REAPER*, an audio editing software was used to slice audio and extract noise prints [26].

Three phrases were picked out as examples supporting the results: A horn section with fast transients and a high pitched bell, *Herbert Victor and His Orchestra - Dream Melody*. The original file can be heard in *Herbert-orig.wav* ◀꣸. A speech example from the record *Harlan and Stanley - Two Rubes and the Tramp Fiddler*, as heard in *Stanley-orig.wav* ◀꣸. A piano piece by *Edvard Grieg - Romances and Ballads for voice and piano, Op 9- No.4, Outward Bound* recorded by an unknown source. This gives a totally different soundscape with more low end and less high end. The original is heard in *Grieg-orig.wav* ◀꣸.

Most parameter combinations were thoroughly tested on about 10 different phrases throughout the whole project. Hundreds of different restored files were generated. Most of

the digital recordings were 16-bit with 44.1 kHz sampling rate, but one case was tested at 24-bit and 96 kHz. Most of the time the gain was set to produce maximum noise reduction without any distortion. The FFT with maxima averaging was the go to noise spectrum estimation algorithm. The frame size was typically tested at 1024, 2048 and 4096 samples. The step size was typically half, quarter or an eight of the frame size, corresponding to 50, 75 or 87.5% overlap. The rectangular, Hamming and Hann windows were tested. The frame weight was most of the time set to 0.98 as recommended by Cappé [6]. When restoring recordings on the behalf of Ringve, a noise floor was often left to give an authentic sound and not to generate any noticeable artifacts.

Some of the recordings seem to be missing some high- and low end, especially to be recorded with a modern pickup. Maybe they were filtered in or after the recording process. I contacted the National Library, responsible for the recording, but they couldn't help me validate if filters were used, or hand me any copies of the raw digital recordings. However they were still considered equally relevant.

3.3 Conducting a listening test

A pilot listening test was conducted with 13 participants. The test was simply to rank six audio clips with varying degree of noise reduction by preference. If the clips were indistinguishable they could be ranked equally. There were three such groups with different recordings. The goal was to identify any preference of noise floor or algorithm. The two main algorithms, spectral subtraction and EMSR were chosen with and without a noise floor. The original was included, and a clip with an alternative EMSR parameter setup. The test was sent by email and the participants were asked to note the type of loudspeakers or headphones that were used.

Before calculating the results the sixth alternative was excluded, as the parameters were inconsistent between groups and rather experimental. The results were given a score from zero to four, four being the most preferable. A tie resulted in the lowest score, also lowering the possible top score. In each of the five cases, the mean of all the participants and groups were taken.

4   Results

4.1   De-Hisser subroutines

Figure 4a shows a raw FFT spectrum together with its corresponding AR-modeled spectrum. The estimates are of equal gain and RMS averaged values. In practice the spectrums are moved slightly with the gain parameter to give the optimum noise reduction. The same spectrum estimated by Welch's method is shown in figure 4b. Here the average is calculated by the arithmetic mean.

It was found that the FFT and AR-modeled noise spectrum amplitude depends heavily on step size. The dependence is most prominent with either RMS or mean averaging. A larger step size result in larger amplitude. Welch's method produced a consistent result regardless of step size. The FFT and AR-model estimates seem to coincide with Welch's method when the step size is a quarter of the frame size.
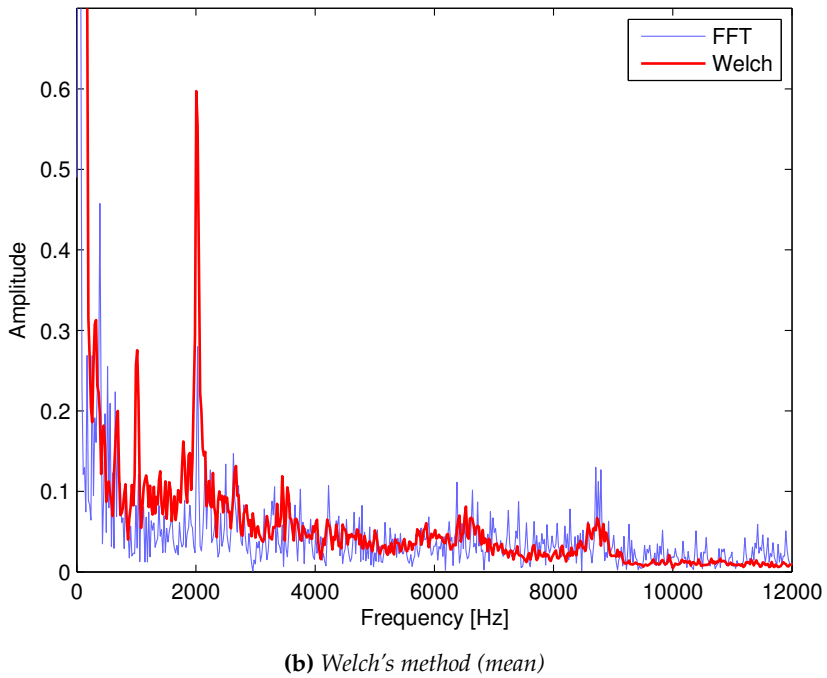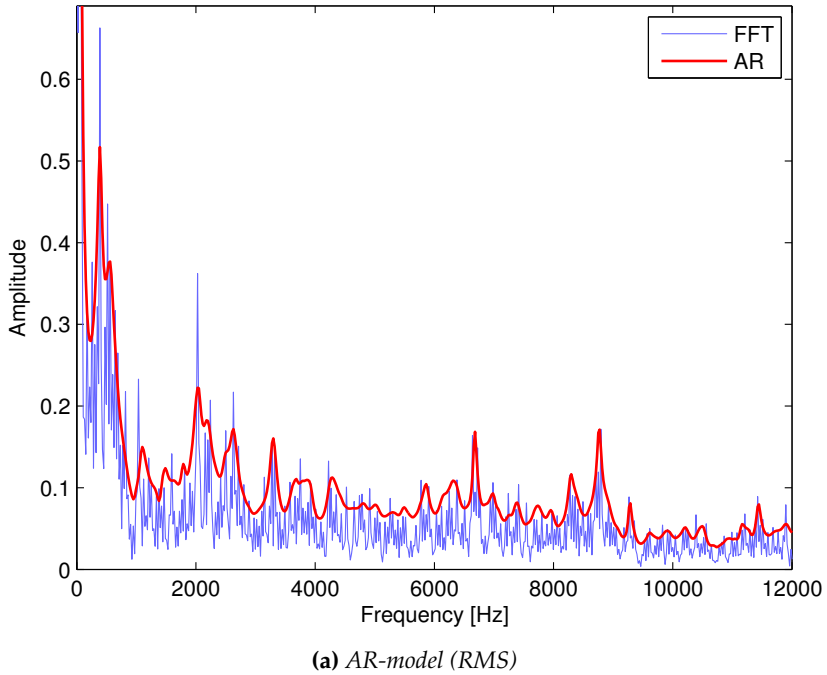
(a) *AR-model (RMS)*

(b) *Welch's method (mean)*

Figure 4: *An estimated spectrum.*

4.2   **Phonograph noise reduction**

In general the spectral subtraction resulted in a noise floor like remainder most of the time. The EMSR could suppress the noise even further, but often gave an unnatural sounding remainder. To mask these artifacts the noise floor had to be raised equally high as that of the spectral subtraction. Reducing the noise as much as possible and later adding the noise floor often yield a more natural sounding result. The Wiener solution and power subtraction were to weak, producing poor results. Some of the restorations exhibit similarities to the musical noise phenomenon.

Figure 5 shows the Herbert Victor Orchestra sound file before and after noise reduction. The original together with the EMSR restored output is heard in *Herbert-orig-emsr.wav*◀᠉. The spectral subtraction variant together with its AR-model and Welch's method equivalent, is heard in *Herbert-fft-ar-welch.wav*◀᠉. The maxima FFT estimate was in general better or equal to the AR-model, and always better than Welch's method when trying to achieve the best end result. If fixed STFT parameters and a mean average were used the Welch's method prove better at low frequencies than the FFT, but didn't reduce the high end sufficiently.

The Stanley and Harley speech example is presented in *Stanley-orig-ssub-emsr.wav*◀᠉. The EMSR gave more noise reduction, but a huge increase in computation time. The JMAP and AMAP were very similar to the EMSR. An EMSR versus AMAP example is heard in *Stanley-emsr-amap.wav*◀᠉. The MMSEP led to an unnatural sounding noise floor. All the EMSR alternatives ran much faster. With medium frame- and step sizes the spectral subtraction clocked in at 0.9 seconds on 10 seconds of audio. The AMAP used 2.1 seconds, whereas the EMSR used 5.2 seconds. The Grieg example show another general case, *Grieg-orig-ssub-emsr.wav*◀᠉. The DART software did not perform as good as the De-Hisser. An example is heard in *Herbert-dart.wav*◀᠉.

A high gain could easily create distortion, increasing gradually with the noise reduction. Light distortion could be suppressed with a low pass filter of about 5 kHz, but often led to dulling of the signal. However overestimating and filtering the signal could give a better first impression. As for the EMSR, the frame weight parameter could give distortion in fast audio material. An example is heard in *Herbert-emsr-dist.wav*◀᠉. Here the distortion is exaggerated giving max noise reduction.

As for the STFT parameters, the frame size was very dependent on material and of course sampling rate. With spectral subtraction a larger frame size often yield better results. A small step size often led to better noise reduction and less distortion, but a significant increase in computation time. The Hamming and Hann windows gave very similar results. The rectangular window with no overlap gave very poor results.

The frame weight parameter gave a reverb/echo like effect with large frame sizes. The signal absence parameter could reduce reverberation and other noise artifacts in rapid pulse like material, such as speech. In *Stanley-absence.wav*◀᠉ the parameter is set to zero, then 0.3. A high probability of signal absence gave distortion and small amplitude modulations to slow material, such as continuous tones.

By analyzing the restored recordings with a band pass filter and a spectrum analyzer, the usable frequency range was found to be approximately 100-6000 Hz.

(a) *Original*

(b) *De-hissed using the EMSR.*
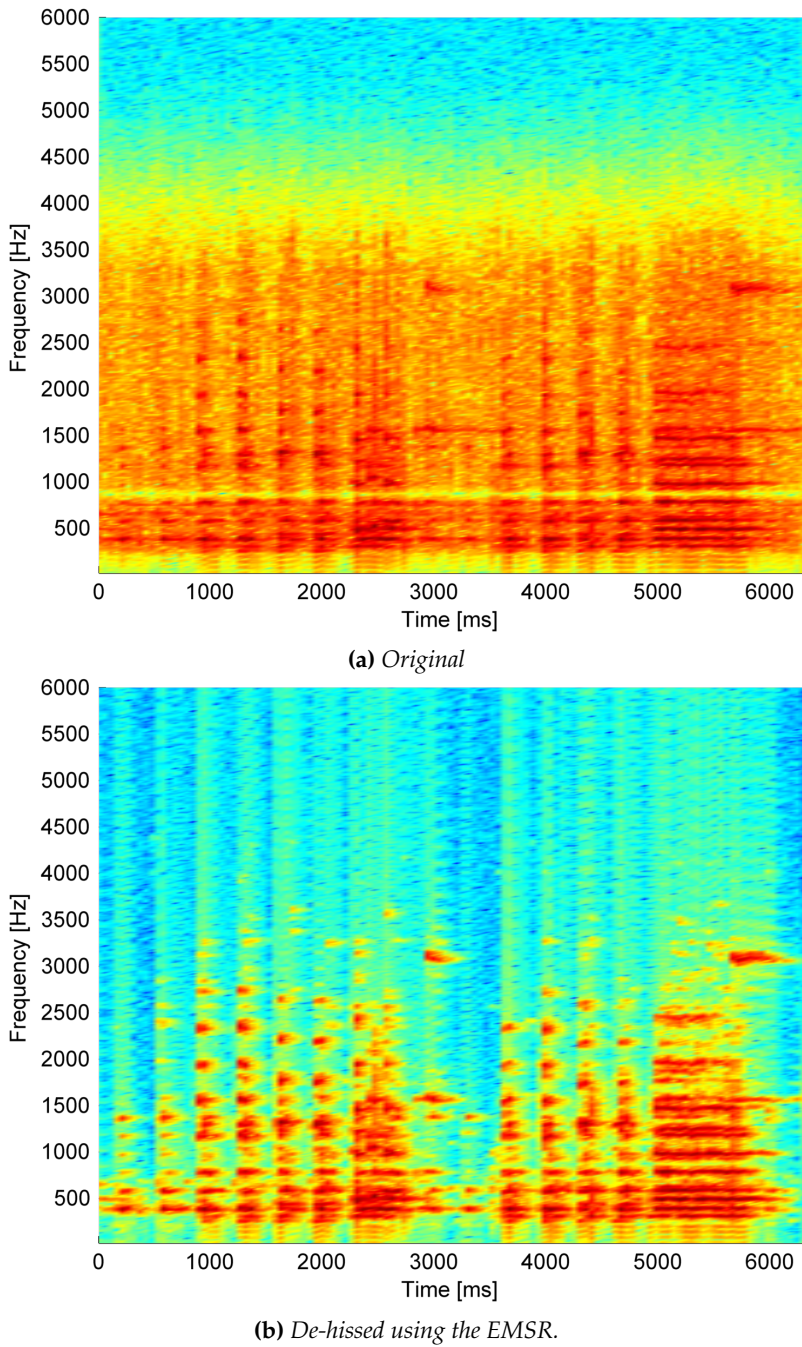
Figure 5: *Spectrogram, Herbert Victor and his Orchestra.*

4.3   Listening test results

The test results are shown in figure 6. All the participants used acceptable playback systems, no laptop speakers or similar. Some reported the EMSR and spectral subtraction (without knowing) as indistinguishable. Not in any case were the original preferred above any other version. Two of the participants reported that some of the clips were sounding like heavily compressed MP3s or sent through a phaser effect.
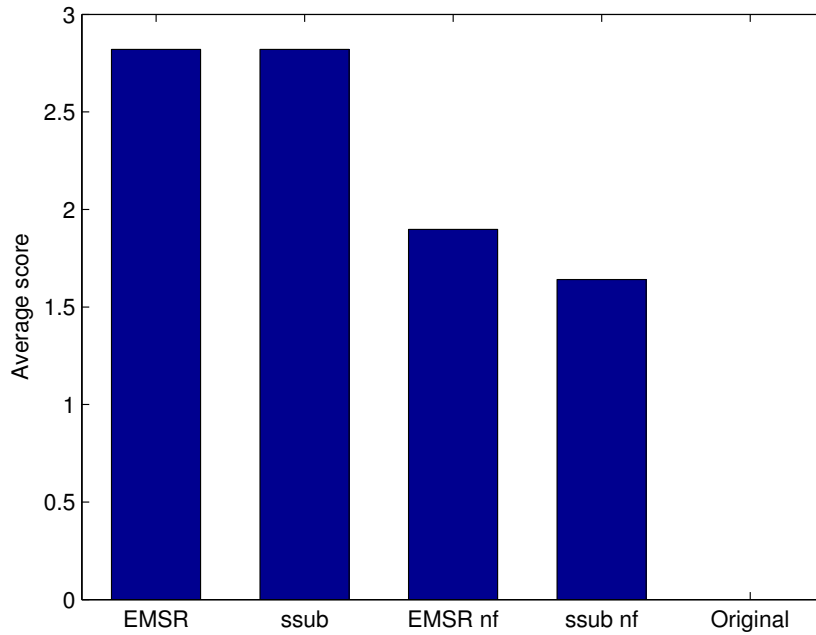
Figure 6: *Listening test results.*

5   Discussion

5.1   De-Hisser

5.1.1   Development

An easier way to partition the signal in the STFT is by using MATLAB's *buffer* function [27]. This slices the signal in overlapping sections as a matrix, allowing all the frames to be processed at once. Also the frame-fitting and some zero padding is done automatically. On the other hand it would have taken up more memory, and the window gain correction would require an individual solution at the signal ends, if not padded manually. I didn't quite figure out the window gain correction function described by [4], as it seems unclear in the inversion of zero tapered window functions.

The frame size parameter could have been presented in milliseconds, allowing a constant frame length with respect to sample rate. To escape unnecessary warnings the size should be forced to a power of two. As of now the dependency fault between step size and spectrum amplitude is corrected by adjusting the noise reduction gain. The fault is most likely in the averaging routine, and a gain correction similar to the window gain correction is probably needed. The amplitude calculations in the AR-model and Welch's method could be avoided by scaling the FFT spectrum.

A lot of time was spent on scaling the θ parameter, not to produce overflow in the *bessli* function. The Bessel function was monitored by checking the returned error flag vector [21]. Only later was the problem found to be the exponential function and solved by truncation, as purposed in [28]. This problem was difficult to separate from the need of a divide by zero protector. The final form, where the noisy signal has to be checked for zeros, was somewhat hidden as the a posteriori SNR could occur as minus one giving a zero valued denominator in the gain vector equation. Another time killer was all the testing of parameter combinations, a more rigorous test method should have been used.

The low pass filter could have been placed pre- or post residue calculation. The latter was selected as not to add back the filtered part in the noise floor generator. This way the residue have to be updated. This update was not present until the very end of the project, thus an untrue residue have been used during the development. In other words good audio removed by the low pass filter have not been heard. This may have led to unnecessary dulling of some signals.

Three types of noise floor generators could have been used: One, by setting a minimum gain in the suppression rule. Two, by adding a fraction of the original signal. Or three, adding the residue. In the EMSR, Cappé suggests setting a minimum a priori SNR [6]. The more general method of adding the residue was selected since the De-Hisser now supports multiple suppression rules. A minimum gain method was actually tested and gave a different sounding, but suitable noise floor using the basic suppression rules. The method suggested by Cappé gave an unnatural sounding noise floor.

5.1.2   Use and results

The large variance of noise in phonograph recordings (and poor noise prints) may have moved the problem of musical noise from the spectral subtraction to the EMSR. As the

spectral subtraction is now only capable of reducing the noise to an acceptable floor, the EMSR give some artifacts similar to musical noise. To assume that the typical phonograph noise is stationary is probably too crude. The noise has a periodic behavior due to the rotational motion of the cylinder. Incorporating a statistic model representing the revolutions should be considered. A more hands one approach could be synchronizing the noise print throughout the audio by cross-correlation or an adaptive gain. In the current model a long noise print have to be compressed into one frame. This results in overestimation as the noise vary more slowly throughout the audio.

The AMAP and JMAP outperform the EMSR. The result can be very similar, but the EMSR alternatives have less complexity. It is always interesting discussing real-time capabilities, and the basic suppression rules or the EMSR alternatives prove to be the most promising algorithms. The suppression rules can process the audio more than ten times faster than real-time. In addition to that, neither the program code nor MATLAB is especially designed for real-time processing.

The frame weight distortion is discussed in [6], and confirms that less distortion is achieved with smaller step sizes. A smaller step size result in less delay before the gain vector is updated, this actually gives less smoothing. The reverb/echo effect is probably a result of the smoothing effect being so delayed that it shapes the current audio with resemblance to the previous audio. As the results show the signal absence parameter is most applicable to speech signals. The parameter is not as useful when maintaining a noise floor, since reverberation is easily masked by noise.

The low pass filter can make it difficult to discover distortion, and should first be enabled after the user is certain no distortion is created. The results also confirm that the use of a modern pickup give an enhanced frequency response.

5.1.3 Further work

An objective test could be performed using a synthetic example, where a clean audio source is contaminated by a random noise generating process. Also more window functions could be tested. Greater bit depth and sampling rate was neither tested enough to give any results, and should be studied further. In theory higher resolution would give more precise spectrum estimations, that could help differentiate noise and good audio.

Some other suppression rules have been purposed. Ephraim and Malah purpose a log-spectral EMSR in [29]. There are also other perceptually motivated suppression rules purposed by Wolfe and Godsill in [30]. A modification to improve the EMSR's transient distortion is purposed in [20], called the *W2* modification. The AMAP and JMAP could have been tested with the signal absence modifier, this is also purposed in [22].

Another good implementation would be a noise gate in between the noise suppressor and noise floor generator. The noise gate could probably do some of the noise removal that the low pass filter now does, but without dulling the signal. On the other hand a standalone noise gate could be preferred in a restoration environment.

As discussed in [2], and still a bit of mystery is that the DART tool do not necessarily need an user selected noise print. It may be possible to extract the noise print automatically by looking for short sections with minimal energy. Anyhow a noise print can be selected, and the DART tool feature a graphical equalizer applied to the noise print. This is a neat feature and would correspond to a frequency dependent noise reduction gain in our case.

The De-Hisser interface was setup for further implementation of a graphical interface. MATLAB have both graphical interface- and standalone application support to compile the

program as a finished software product.

5.2 Listening test

Surprisingly the EMSR and spectral subtraction were equally preferred without any noise floor. Another subjective listening test found in [31], show more preference towards the EMSR. In figure 6, we see the EMSR have a slight edge over spectral subtraction in the clips containing a noise floor. If we look at the two bars with noise floor versus the bars without, it is clearly the participants prefer no noise floor. Due to the nature of the examples and rank setup, the participants may have been caught up in finding the clip with least noise, rather than actually choosing the one they preferred. The *ABX* test method is a good alternative. From a statistical standpoint the clips should have been graded instead of ranked. This would allow a measure of distance between the clips.

The listening test material were the first batch of recordings restored with the program, so the parameters were set with little experience. A too low cutoff frequency was used and a too much gain, resulting in a dull and distorted output. Today the parameters would have been set differently, and the difference between the algorithms might had been clearer. The test should have included a speech or song example.

The phase modulation effect noted by some, could be the zero phase shift behavior of the suppression rules. The same phenomenon is surely present in MP3 encoding and should be studied further. I chose not to do any objective tests as there are multiple in other articles [8][22][31].

6   Conclusion

A de-hissing tool were created with MATLAB and proved to be very powerful. We were able to reduce noise in old phonograph recordings by a significant amount. Both some basic- and state of the art noise reduction algorithms were successfully realized. The EMSR or the alternative AMAP algorithm gave most noise reduction, although the basic spectral subtraction method gave a pleasant result with less complexity. Some noise artifacts are still present, and a few guidelines and further development have been purposed.

We built a solid framework around the STFT. The program is easily expandable to support more algorithms and a graphical interface. Noise reduction by STSA prove to be a huge research field with all its algorithms and parameters. It takes some time before you can start modify, automate and qualify the algorithms best suited for phonograph restoration.

One important result was that phonograph audio, with its large noise variance, should only be considered semi-stationary. A model supporting the periodic noise behavior should be implemented. A subjective test show that people prefer the restorations by far, and the availability of old recordings highly depend on noise reduction techniques.

References

[1] Inc. The MathWorks. Matlab, 2013. `http://www.mathworks.se/products/matlab/`.

[2] Vegard Hella. Digital audio restoration. *TTT4561, NTNU*, 2012.

[3] Wikipedia. Phonograph cylinder, May 2013. `http://en.wikipedia.org/wiki/Phonograph_cylinder`.

[4] Simon J. Godsill and Peter J.W. Rayner. *Digital Audio Restoration - a statistical model based approach*. Springer, September 1998.

[5] DARTECH Inc. Dart pro 24, 2006. `http://www.dartpro.com/products/Dart_Pro_24.asp`.

[6] Olivier Cappé. Elimination of the musical noise phenomenon with the ephraim and malah noise suppressor. *IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 2, NO. 2*, April 1994.

[7] Bernhard Wieland. Speech signal noise reduction with wavelets. *Diplomarbeit an der Universität Ulm*, October 2009.

[8] Goushen Yu and Stéphane Mallat. Audio denoising by time-frequency block thresholding. *IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 56, NO. 5*, May 2008.

[9] Wikipedia. Noise gate, May 2013. `http://en.wikipedia.org/wiki/Noise_gate`.

[10] Thomas Bårdsen. Fra evig is til evig tid. *MUV450 Masteroppgave i musikkvitenskap, Høgskolen i Nesna*, 2010.

[11] Wikipedia. Riaa equalization, May 2013. `http://en.wikipedia.org/wiki/RIAA_equalization`.

[12] Wikipedia. Short-time fourier transform, May 2013. `http://en.wikipedia.org/wiki/Short-time_Fourier_transform`.

[13] Wikipedia. Window function, May 2013. `http://en.wikipedia.org/wiki/Window_function`.

[14] Wikipedia. Digital filter, May 2013. `http://en.wikipedia.org/wiki/Digital_filter`.

[15] Mathworks. Linear prediction and autoregressive modeling, May 2013. `http://www.mathworks.se/help/signal/examples/linear-prediction-and-autoregressive-modeling.html`.

[16] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing third edition*. Prentice-Hall International, Inc., 1995.

[17] Wikipedia. Wiener filter, March 2013. `http://en.wikipedia.org/wiki/Wiener_filter`.

[18] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-32, NO. 6*, December 1984.

[19] Y. Ephraim and D. Malah. Speech enhancement using optimal non-linear spectral amplitude estimation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1983.

[20] S. Canazza, G. De Poli, G. A. Mian, and A. Scarpa. Real time comparison of audio restoration methods based on short time spectral attenuation. *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01), Limerick, Ireland*, December 2001.

[21] Mathworks. Modified bessel function of first kind, March 2013. `http://www.mathworks.se/help/matlab/ref/besseli.html`.

[22] Patrick J. Wolfe and Simon J. Godsill. Efficient alternatives to the ephraim and malah suppression rule for audio signal enhancement. *EURASIP Journal on Applied Signal Processing*, February 2003.

[23] Mathworks. aryule, April 2013. `http://www.weizmann.ac.il/matlab/toolbox/signal/aryule.html`.

[24] Hyee Wang. Lpc spectra estimate, February 2009. Help forum at: `http://compgroups.net/comp.dsp/lpc-spectra-estimate/1179362`.

[25] Mathworks. Does the pwelch function provide unbiased estimation of the power spectral density?, June 2009. `http://www.mathworks.se/support/solutions/en/data/1-17M0Q/index.html?product=SG&solution=1-17M0Q`.

[26] Cockos Incorporated. Reaper digital audio workstation, 2011. `http://www.reaper.fm/`, 28.11.12.

[27] Mathworks. buffer, April 2013. `http://www.mathworks.se/help/signal/ref/buffer.html`.

[28] Christoph Domes. Hiss reduction, basic methods for audio restoration. *Digitale Audiotechnik 2, Signal Processing and Speech Communication Laboratory, Graz University of Technology*, May 2009.

[29] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-33, NO. 2*, April 1985.

[30] Patrick J. Wolfe and Simon J. Godsill. Perceptually motivated approaches to music restoration. *Journal of New Music Research*, January 2001.

[31] A. Akbari Azirani, R. Le Bouquin Jeannés, and G. Faucon. Speech enhancement using a wiener filtering under signal presence uncertainty. *Laboratoire du Traitement du Signal et de l'Image - Université de Rennes*, - -.

[32] Joseph Nuzman. Audio restoration: An investigation of digital methods for click removal and hiss reduction. *University of Maryland Institute for Advanced Computer Studies*, January 2004.

A   MATLAB code

A.1   De-hisser STSA

```
1  %% De-Hisser STSA
2  % Vegard Hella, NTNU, 2013.
3  close all;
4  clear all;
5  clc;
6
7  %-------------------- file dialog -----------------------
8  inputfile      = '../audio_examples/Stanley-orig.wav';
9  %audio_start = 1;       audio_stop = 277000;          %optional trim
10
11 noiseprintfile = '../audio_examples/Stanley-np.wav';   %=inputfile
12 %noise_start = 1;       noise_stop = 158000;          %optional trim
13
14 outputfile     = '../audio_examples/Stanley.wav';
15
16 %------------------- user parameters --------------------
17 %Algorithms
18 suppression_rule      = 'ssub';   %ssub, wiener, psub, emsr, jmap,
19                                   %amap, mmsep
20 estimation_algorithm  = 'fft';    %fft, ar, welch (mean)
21 noise_averaging       = 'max';    %max, mean, rms
22
23 %STFT
24 L          = 2*1024;       %frame size        [typ. 1024, 2048, 4096]
25 M          = L/4;          %step size         [typ. L/2, L/4, L/8]
26 gwin       = 'hann';       %rectwin, hamming, hann, '>>help window'
27
28 %Noise reduction
29 gain       = 1.00;         %noise reduction gain
30 Rmin       = 0.00;         %noise floor gain              [0 -> 1]
31 rho        =  170;         %AR-model order              [1 -> (L-1)]
32
33 %EMSR
34 alpha      = 0.98;         %previous frame weight         [0 -> 1]
35 qk         = 0.20;         %probability of signal absence [0 -> 0.99]
36
37 %LP-filter
38 lowpass    = false;        %optional low pass filter
39 f3db       = 6000;         %cutoff frequency [Hz]
40
41 %Playback
42 listen_original   =  false;
43 listen_noiseprint =  false;
44 listen_result     =  true;
45 listen_residue    =  false;
46
47 %Save
48 save_result    =  false;
49 save_residue   =  false;
50
```

```matlab
51
52  %---------------------- application ------------------------
53  % Load files
54  [wav_file, Fs, nbits] = wavread(inputfile);
55  if exist('audio_stop', 'var')
56      wav_file = wav_file(audio_start:audio_stop);        %trim
57  end
58
59  noise_print = wavread(noiseprintfile);   %extract noise print
60  if exist('noise_stop', 'var')
61      noise_print = noise_print(noise_start:noise_stop);  %trim
62  end
63
64  win = gwin;
65  gwin = eval([gwin, '(L)']);          %generate window function
66
67  % Estimate noise print
68  noise_spectrum = gain * spectrum_estimator(noise_print, ...
69      gwin, M, estimation_algorithm, rho, noise_averaging);
70
71  %save(['grieg2/noise spectrum/', estimation_algorithm, '_', ...
72  %    noise_averaging], 'noise_spectrum');
73
74  % Noise removal
75  tic
76  wav_estimate = noise_suppressor(wav_file, noise_spectrum, ...
77      gwin, M, suppression_rule, alpha, qk);
78  toc
79
80  wav_residue = wav_file - wav_estimate;
81
82  % Low pass filter signal
83  if lowpass
84      wav_estimate = lpfilter(wav_estimate, f3db, Fs); %low pass filter
85  end
86
87  % Add noise floor
88  if (Rmin > 0)
89      wav_estimate = wav_estimate + Rmin*wav_residue;
90  end
91
92  wav_residue = wav_file - wav_estimate;           %update residue
93
94
95  if listen_original          %listen to original file
96      sound(wav_file, Fs);
97  end
98  if listen_noiseprint        %listen to noiseprint
99      sound(noise_print, Fs);
100 end
101 if listen_result            %listen to result
102     sound(wav_estimate, Fs);
103 end
104 if listen_residue           %listen to residue
105     sound(wav_residue, Fs);
106 end
107
108
109 % Save files, state parameters in filename
110 outputfile = [outputfile(1:end-4), '_', suppression_rule, ...
```

```matlab
111          '_', estimation_algorithm, '_', noise_averaging, ...
112          '_', win, ...
113          '_N', num2str(L), ...
114          '_M', num2str(M), ...
115          '_g', num2str(gain), ...
116          '_Rm', num2str(Rmin), '.wav'];
117
118 if lowpass
119          outputfile = [outputfile(1:end-4), ...
120          '_f', num2str(f3db), '.wav'];
121 end
122 if max(strcmpi(estimation_algorithm, {'ar', 'ceps'}))
123          outputfile = [outputfile(1:end-4), ...
124          '_r', num2str(rho), '.wav'];
125 end
126 if strcmpi(suppression_rule, 'emsr')
127      outputfile = [outputfile(1:end-4), ...
128          '_a', num2str(alpha), ...
129          '_q', num2str(qk), '.wav'];
130 end
131
132 if save_result
133      wavwrite(wav_estimate, Fs, nbits, outputfile);
134 end
135 if save_residue
136      wavwrite(wav_residue, Fs, nbits, [outputfile(1:end-4), '_res.wav']);
137 end
```

A.2   Spectrum estimator

```matlab
1  function Dk_hat = spectrum_estimator(dn, gwin, M, algorithm, rho, averaging)
2  %Dk_hat = SPECTRUM_ESTIMATOR(dn, gwin, M, algorithm, rho) estimates the
3  %twosided spectrum, 'D(k) hat' from the time signal 'd(n)'.
4  %   'gwin' is a window function describing a frame of size 'L'. 'M' is the
5  %   step size between sucessive frames. 'algorithm' gives the type of
6  %   estimation method:
7  %       'fft'       (Fast Fourier Transform)
8  %       'ar'        (AR-model, with order 'rho')
9  %       'ceps'      (cepstrum)
10 %       'welch'     (Welch's method, a modified periodogram)
11 %   'rho' is the AR-model or cepstrum order.
12 %   'averaging' gives the averaging process of multiple frames:
13 %       'max'       (maximum noise amplitudes)
14 %       'mean'      (arithmetic mean)
15 %       'rms'       (root mean square)
16 %   'Dk_hat' is the resulting spectrum of size 'L'. Depending on
17 %   algorithm 'Dk_hat' may be complex.
18
19 L = length(gwin);
20
21 if strcmpi(algorithm, 'welch')
22     Dk_hat = pwelch(dn, gwin, (L-M), 'twosided');
23     Dk_hat = sum(gwin)^2/norm(gwin)^2 * Dk_hat;      %pwelch scale
24 else
25
26 N = length(dn);
27 dn = [zeros(L,1); dn; zeros((ceil(N/L)*L - N) + L, 1)];     %zero padding
28 N = length(dn);                                            %update length
29
30 Dk_hat = zeros(L, 1);                                      %preallocate memory
31
32 for n = 1:M:N-(L-M)                    %n is the signal sample index
33
34     dn_frame = gwin .* dn(n:n+L-1);
35
36     switch algorithm
37         case 'fft'
38             Dk = fft(dn_frame);
39
40         case 'ar'
41             if sum(abs(dn_frame)) > 0
42                 [A, sigma2] = aryule(dn_frame, rho);
43                 Dk = sqrt(sigma2*L) * freqz(1, A, L, 'whole');
44                 %spectrum shape is represented by the filter coefficients.
45                 %signal energy is represented by the variance, sigma^2.
46             else
47                 Dk = 0;
48             end
49
50         otherwise
51             disp('Error: no recognizable choice of algorithm.')
52             break;
53     end
54
55     switch averaging
56         case 'max'
```

```
57              i = abs(Dk) > abs(Dk_hat);
58              Dk_hat(i) = Dk(i);
59
60          case 'mean'
61              Dk_hat = (Dk_hat + Dk) / 2;
62
63          case 'rms'
64              Dk_hat = sqrt((Dk_hat.^2 + Dk.^2) / 2);
65
66          otherwise
67              disp('Error: no recognizable choice of averaging method.')
68              break;
69      end
70
71  end
72
73  end
74
75  end
```

A.3   Noise suppressor

```matlab
1  function xn_hat = noise_suppressor(yn, Dk, gwin, M, rule, alpha, qk)
2  %xn_hat = NOISE_SUPPRESSOR(yn, Dk, gwin, M, rule) reduces noise in 'y(n)',
3  %based on noise spectrum 'D(k)'.
4  %    'Dk' can be complex. 'Dk' is of length 'L'.
5  %    'gwin' is a window function describing a STFT frame of size 'L'.
6  %    'M' is the step size between successive frames. 'rule' is a selection
7  %    of noise suppression rules:
8  %         'ssub'      (spectral subtraction)
9  %         'wiener'    (wiener solution)
10 %         'psub'      (power subtraction)
11 %         'emsr'      (Ephraim-Malah suppression rule)
12 %         'jmap'      (Joint MAP Spectral Amplitude and Phase Estimator)
13 %         'amap'      (Approximate MAP Spectral Amplitude Estimator)
14 %         'mmsep'     (MMSE Spectral Power Estimator)
15 %    'xn_hat' is then the estimated noiseless signal. 'alpha' is a
16 %    smoothing factor (EMSR), and 'qk' is the probability of signal
17 %    absence (EMSR).
18
19 L = length(gwin);
20 N = length(yn);
21
22 yn = [zeros(L,1); yn; zeros((ceil(N/L)*L - N) + L, 1)];      %zero padding
23 Npad = length(yn);                        %update padded length
24
25 xn_hat = zeros(Npad, 1);                  %preallocate memory
26 Gk = zeros(L, 1);
27 Xk_hat = zeros(L, 1);
28
29 for n = 1:M:Npad-(L-M)      %n is the signal sample index
30
31     yn_frame = yn(n:n+L-1);
32     Yk = fft(gwin .* yn_frame);
33
34     switch rule
35         case 'ssub'                                %Spectral subtraction
36             i = (abs(Yk) - abs(Dk)) > 0;
37             Gk(i) = (abs(Yk(i)) - abs(Dk(i))) ./ abs(Yk(i));
38
39         case 'wiener'                              %Wiener filter solution
40             i = (abs(Yk) - abs(Dk)) > 0;
41
42             SY = abs(Yk(i)).^2;
43             SD = abs(Dk(i)).^2;
44
45             Gk(i) = (SY - SD) ./ SY;
46
47         case 'psub'                                %Power spectrum subtraction
48             i = (abs(Yk) - abs(Dk)) > 0;
49
50             SY = abs(Yk(i)).^2;
51             SD = abs(Dk(i)).^2;
52
53             Gk(i) = sqrt( (SY - SD) ./ SY );
54
55         case 'emsr'                                %Ephraim-Malah suppression rule
56             Gk = emsrc(Yk, Dk, Xk_hat, alpha, qk);
```

50

```matlab
57                  %[Gk, Rpost] = emsrcw2(Yk, Dk, Xk_hat, alpha, qk);
58
59          case 'lsanc'
60              disp(['The Log Spectral Amplitude Non Causal Estimator', ...
61                  '(LSANC) is not available.'])
62              break;
63
64          case 'mmsep' %Minimum Mean-Square Error Spectral Power Estimator
65              Gk = mmsep(Yk, Dk, Xk_hat, alpha);
66
67          case 'amap'
68              Gk = amap(Yk, Dk, Xk_hat, alpha);
69
70          case 'jmap'
71              Gk = jmap(Yk, Dk, Xk_hat, alpha);
72
73          otherwise
74              disp('Error: no recognizable suppression rule.')
75              break;
76      end
77
78      Xk_hat = Gk .* Yk;
79
80      xn_frame = ifft(Xk_hat);
81
82      if M > L/2                                  %gain correction
83          xn_frame = xn_frame .* (1./gwin);
84      else
85          xn_frame = xn_frame .* (M/sum(gwin));
86      end
87
88      xn_hat(n:n+L-1) = xn_hat(n:n+L-1) + xn_frame;    %overlap add
89
90  end
91
92  xn_hat = xn_hat(L+1:L+N);                        %zero padding removal
93
94  end
```

51

A.4   EMSR

```matlab
function Gk = emsrc(Yk, Dk, Xk_hat, alpha, qk)
%Gk = EMSRC(Yk, Dk, Xk_hat) The Ephraim-Malah suppression rule as defined
%by Cappé.
%   where 'Yk' is the noisy signal spectrum, 'Dk' the noise spectrum and
%   'Xk_hat' is the previous frame of the estimated noiseless signal. 'Gk'
%   is then the resulting frequency gain vector. 'alpha' is the frame
%   weighting, and 'qk' the probability of signal absence.

    L = length(Yk);
    Gk = zeros(L, 1);

    i = (abs(Yk) > 0) & (abs(Dk) > 0);             %divide by zero protector

    Rpost = (abs(Yk(i)).^2 ./ abs(Dk(i)).^2) - 1;

    Rprio = (1-alpha) * max(Rpost, 0) ...   %positives
        + alpha * (abs(Xk_hat(i)).^2 ./ abs(Dk(i)).^2);

    Rprio = Rprio ./ (1 - qk);

    Rpost(Rpost > 700) = 700;    %exponential of theta overflow protection

    theta = (1+Rpost) .* (Rprio ./ (1+Rprio));

    I0 = besseli(0, theta/2);        %zero order
    I1 = besseli(1, theta/2);        %first order

    M_em = exp(-theta/2) .* ((1+theta).*I0 + theta.*I1);

    Gk(i) = (sqrt(pi)/2) ...
        * sqrt( (1./(1+Rpost)) .* (Rprio./(1+Rprio)) ) .* M_em;

    if qk > 0                                  %signal presence uncertainty
        mu = (1-qk) ./ qk;
        Lambda = mu .* exp(theta) ./ (1+Rprio);
        Gk(i) = Lambda ./ (1+Lambda ) .* Gk(i);
    end

end
```

A.5   JMAP

```matlab
function Gk = jmap(Yk, Dk, Xk_hat, alpha)
%Gk = JMAP(Yk, Dk, Xk_hat) Joint Maximum A Posteriori Spectrum Amplitude
%and Phase Estimator.
%   where 'Yk' is the noisy signal spectrum, 'Dk' the noise spectrum and
%   'Xk_hat' is the previous frame of the estimated noiseless signal.
%   'alpha' is the frame weight. 'Gk' is then the frequency gain vector.

    N = length(Yk);
    Gk = zeros(N, 1);

    i = (abs(Yk) > 0) & (abs(Dk) > 0);          %divide by zero protector

    Rpost = (abs(Yk(i)).^2 ./ abs(Dk(i)).^2) - 1;

    Rprio = (1-alpha) * max(Rpost, 0) ...                    %positives
        + alpha * (abs(Xk_hat(i)).^2 ./ abs(Dk(i)).^2);

    Gk(i) = (Rprio + sqrt(Rprio.^2 + 2*(1+Rprio).*(Rprio./(1+Rpost)))) ...
        ./ (2*(1+Rprio));

end
```

A.6 AMAP

```matlab
1  function Gk = amap(Yk, Dk, Xk_hat, alpha)
2  %Gk = AMAP(Yk, Dk, Xk_hat) Approximate Maximum A Posteriori Spectral
3  %Amplitude Estimator.
4  %   where 'Yk' is the noisy signal spectrum, 'Dk' the noise spectrum and
5  %   'Xk_hat' is the previous frame of the estimated noiseless signal.
6  %   'alpha' is the frame weight. 'Gk' is then the frequency gain vector.
7
8      N = length(Yk);
9      Gk = zeros(N, 1);
10
11     i = (abs(Yk) > 0) & (abs(Dk) > 0);          %divide by zero protector
12
13     Rpost = (abs(Yk(i)).^2 ./ abs(Dk(i)).^2) - 1;
14
15     Rprio = (1-alpha) * max(Rpost, 0) ...                      %positives
16         + alpha * (abs(Xk_hat(i)).^2 ./ abs(Dk(i)).^2);
17
18     Gk(i) = (Rprio + sqrt(Rprio.^2 + (1+Rprio).*(Rprio./(1+Rpost)))) ...
19         ./ (2*(1+Rprio));
20
21 end
```

54

A.7 MMSEP

```matlab
function Gk = mmsep(Yk, Dk, Xk_hat, alpha)
%Gk = MMSEP(Yk, Dk, Xk_hat, alpha) Minimum Mean-Square Error Spectral Power
%Estimator.
%   where 'Yk' is the noisy signal spectrum, 'Dk' the noise spectrum and
%   'Xk_hat' is the previous frame of the estimated noiseless signal.
%   'alpha' is the frame weight. 'Gk' is then the frequency gain vector.

    N = length(Yk);
    Gk = zeros(N, 1);

    i = (abs(Yk) > 0) & (abs(Dk) > 0);          %divide by zero protector

    Rpost = (abs(Yk(i)).^2 ./ abs(Dk(i)).^2) - 1;

    Rprio = (1-alpha) * max(Rpost, 0) ...                   %positives
        + alpha * (abs(Xk_hat(i)).^2 ./ abs(Dk(i)).^2);

    theta = (1+Rpost) .* (Rprio ./ (1+Rprio));

    Gk(i) = sqrt((Rprio./(1+Rprio)) .* ((1+theta)./(1+Rpost)));


end
```

A.8 Low pass filter

```matlab
function yn = lpfilter(xn, f0, Fs, order)
%yn = LPFILTER(xn, f0, Fs, order) Low pass butterworth filter.
%   where 'xn' is the input signal, 'f0' the cut off frequency, 'Fs' the
%   sampling time and the filter 'order'. Then 'yn' is the filtered output
%   signal.

if nargin < 4
    order = 4;
end

Wn = f0 / (Fs/2);
[B, A] = butter(order, Wn, 'low');  %butterworth LP coefficients

yn = filter(B, A, xn);                         %filter the signal

end
```

B   Listening test results

	I	U	Ma	G	L	P	M	N	T	H	K	J	C
A1 emsr nf	2	3	4	2	3	2	3	3	3	3	2	2	2
A2 orig	1	1	1	1	1	1	1	1	1	1	1	1	1
A3 emsr	3	4	2	3	4	4	4	4	5	4	3	3	4
A4 ssub nf	2	2	5	2	2	3	2	2	2	2	4	2	3
A5 ssub	4	5	3	4	4	5	5	5	4	5	5	3	5
B1 emsr	4	5	2	4	5	2	4	5	3	5	4	5	5
B2 ssub	3	4	3	3	3	3	3	4	3	3	3	4	4
B3 ssub nf	2	3	2	3	4	3	2	3	2	4	5	3	3
B4 orig	1	1	1	1	1	1	1	1	1	1	1	1	1
B5 emsr nf	2	2	4	2	2	4	5	2	3	2	2	2	2
C1 ssub nf	2	2	3	2	3	3	4	2	2	2	2	2	2
C2 ssub	3	4	3	3	2	4	5	4	3	5	5	4	4
C3 orig	1	1	1	1	1	1	1	1	1	1	1	1	1
C5 emsr nf	4	3	3	4	4	5	3	3	3	3	4	3	3
C6 emsr	4	5	2	5	5	2	2	4	3	4	3	5	5

C   Environmental risk analysis

NTNU		Utarbeidet av	Nummer	Dato
HMS	Kartlegging av risikofylt aktivitet	HMS-avd.	HMSRV2601	22.03.2011
		Godkjent av	Side	Erstatter
		Rektor	1 av 1	01.12.2006

Enhet: Institutt for Elektronikk og Telekommuniksjon (IET)
Deltakere ved kartleggingen (m/ funksjon): Vegard Hella (student), Jan Tro (veileder)

Dato: 06.06.13

Kort beskrivelse av hovedaktivitet/hovedprosess: Programmering

ID nr.	Aktivitet/prosess	Ansvarlig	Eksisterende dokumentasjon	Eksisterende sikringstiltak	Lov, forskrift o.l.	Kommentar
1	Musearm og stive skuldre	VH				

	NTNU		
	⬛		
	HMS/KS		

	utarbeidet av	Nummer	Dato
	HMS-avd.	HMSRV2603	04.02.2011
	godkjent av	side	Erstatter
	Rektor	1 av 2	9.2.2010

Risikovurdering

Enhet: Institutt for Elektronikk og Telekommunikasjon (IET)   **Dato:** 06.06.13
Linjeleder: Ragnar Hergum
Deltakere ved risikovurderingen (m/ funksjon): Vegard Hella (student), Jan Tro (veileder)

ID nr	Aktivitet fra kartleggings-skjemaet	Mulig uønsket hendelse/ belastning	Vurdering av sannsyn-lighet (1-5)	Vurdering av konsekvens:					Risiko-verdi	Kommentarer/status Forslag til tiltak
				Menneske (A-E)	Ytre miljø (A-E)	Øk/ materiell (A-E)	Om- dømme (A-E)			
1	Musearm og stive skuldre		3	B	A	A	A		3B	Sitte riktig, pauser.

Sannsynlighet
1. Svært liten
2. Liten
3. Middels
4. Stor
5. Svært stor

Konsekvens
A. Svært liten
B. Liten
C. Moderat
D. Alvorlig
E. Svært alvorlig

Risikoverdi (beregnes hver for seg):
Menneske = Sannsynlighet x Konsekvens Menneske
Ytre miljø = Sannsynlighet x Konsekvens Ytre miljø
Økonomi/materiell = Sannsynlighet x Konsekvens Øk/matriell
Omdømme = Sannsynlighet x Konsekvens Omdømme

59

NTNU
HMS/KS

	utarbeidet av	Nummer	Dato
	HMS-avd.	HMSRV2603	04.02.2011
	godkjent av	side	Erstatter
	Rektor	2 av 2	9.2.2010

Risikovurdering

Sannsynlighet vurderes etter følgende kriterier:

Svært liten 1	Liten 2	Middels 3	Stor 4	Svært stor 5
1 gang pr 50 år eller sjeldnere	1 gang pr 10 år eller sjeldnere	1 gang pr år eller sjeldnere	1 gang pr måned eller sjeldnere	Skjer ukentlig

Konsekvens vurderes etter følgende kriterier:

Gradering	Menneske	Ytre miljø Vann, jord og luft	Øk/materiell	Omdømme
E Svært Alvorlig	Død	Svært langvarig og ikke reversibel skade	Drifts- eller aktivitetsstans >1 år.	Troverdighet og respekt betydelig og varig svekket
D Alvorlig	Alvorlig personskade. Mulig uførhet.	Langvarig skade. Lang restitusjonstid	Driftsstans > ½ år Aktivitetsstans i opp til 1 år	Troverdighet og respekt betydelig svekket
C Moderat	Alvorlig personskade.	Mindre skade og lang restitusjonstid	Drifts- eller aktivitetsstans < 1 mnd	Troverdighet og respekt svekket
B Liten	Skade som krever medisinsk behandling	Mindre skade og kort restitusjonstid	Drifts- eller aktivitetsstans < 1uke	Negativ påvirkning på troverdighet og respekt
A Svært liten	Skade som krever førstehjelp	Ubetydelig skade og kort restitusjonstid	Drifts- eller aktivitetsstans < 1dag	Liten påvirkning på troverdighet og respekt

Risikoverdi = Sannsynlighet x Konsekvens
Beregn risikoverdi for Menneske. Enheten vurderer selv om de i tillegg vil beregne risikoverdi for Ytre miljø, Økonomi/materiell og Omdømme. I så fall beregnes disse hver for seg.

Til kolonnen "Kommentarer/status, forslag til forebyggende og korrigerende tiltak":
Tiltak kan påvirke både sannsynlighet og konsekvens. Prioriter tiltak som kan forhindre at hendelsen inntreffer, dvs. sannsynlighetsreduserende tiltak foran skjerpet beredskap, dvs. konsekvensreduserende tiltak.