universität wien

# MASTERARBEIT

Titel der Masterarbeit

## Signaltransformation via Gabor Multiplier

Verfasserin

Roswitha Sibylle Bammer, BSc

angestrebter akademischer Grad

Master of Science (MSc)

Wien, April 2015

| | |
|---|---|
| Studienkennzahl lt. Studienblatt: | A 066 821 |
| Studienrichtung lt. Studienblatt: | Masterstudium Mathematik |
| Betreuerin: | Dr. Monika Dörfler |

*To Genoveva and Johanna*

# Table of Contents

# List of Figures

# List of Tables

## Acknowledgment

First of all I want to thank Michael Grosser, who introduced me to Monika Dörfler. He therefore helped me finding an advisor who aroused my passion for the field of *Gabor analysis*. Of course I want to thank my advisor Monika Dörfler for her inspiration in combining music and mathematics and also for her support during the last year. Also many thanks to Andrea Hüthmayr, who proofread this thesis. Finally I want to thank my parents for supporting me over the last years of my studies and also Matthäus Zimmer for mental support.

## Abstract

*Following the thesis of Anaïk Olivero with the name " Les Multiplicateurs Temps-Fréquence. Applications à l'Analyse et la Synthèse de Signaux Sonores et Musicaux", a transformation of two sufficiently similar signals through Gabor multipliers is provided. Therefore the main idea of Gabor analysis and Gabor frames has to be developed. Searching Gabor multipliers used as linear time-varying filters give rise to an inverse problem. This can be solved explicitly in the special case of only diagonal entries in the signals Gabor transformation. For the general case, proximal gradient methods and its improvements for solving the minimization problem are introduced. The thesis ends with discussing some applications created by using the theory in a developed* MATLAB *code.*

# 1 Introduction

Musical sounds, occurring in everyday life, are phenomena which can be completely described through mathematics.
In this thesis we don't want to describe the sound itself, but we want to give a description of a transformation. A transformation between two musical sounds with sufficient similarity. As general order in signal processing we follow the structure: Analysis of the signals - performing a certain transformation - and synthesis of the modified analysis coefficients. We can ask the question what is happening, if we go from one signal to another. To do so, we consider a transformation taking place in the time-frequency plane. There we practice Gabor analysis and we can look at time and frequency at the same time, although we cannot be precise simultaneously in both arguments. The main theory needed for this thesis which captures the topic of Gabor analysis, can be found in [16] and [13].
The considered transformation is developed step by step, starting with linear time-varying filters and makes the way up to Gabor multipliers, such that this thesis could be understood by non-experts of the field of Gabor analysis. For further information to linear time-varying filters view Chapter 11 of [5]. Some theory to Gabor multiplier can be observed in [18] and [19]. The main part of this thesis is based on these references.
The goal of this thesis is to find the best fitting Gabor multiplier through given input and output signals. Such a transformation through Gabor multipliers, done in the time-frequency plane, gives rise to an inverse problem. We want to solve the corresponding problem through regularization. In Chapter 4 explicit solutions of this problem can be calculated in the diagonal case, using different regularization terms. This chapter is followed by a discussion of the properties and difficulties of the different regularization terms by means of an example. Chapter 6 finds the solution of the general inverse problem with the help of proximal gradient methods. Therefore basic theory of such methods is stated in [9] and theory of its improvement can be found in [3]. With the knowledge of the developed theory, a MATLAB code is provided in Chapter 7. Some applications follow in Chapter 8 which emphasize the behavior of different regularization terms and associated regularization parameters. These differences yield a multitude of different Gabor multipliers. For example, there exist interesting differences between the transformation from start into target signal and the other way round, i.e. the transformation from target into start signal. Another point that will be highlighted in the experimental chapter, is the behavior of the Gabor multiplier with respect to different dynamics.
To develop this theory, some basics are needed, i.e. the way of displaying a signal. We don't want to show the signals only in the time plane, as displayed normally, or in the frequency plane, computed through its fast Fourier transformation. We want them to be displayed in both, the time-frequency plane. This leads to the theory of Gabor analysis and Gabor frames, which is essential for the understanding of this thesis.

# 2 Basic Theory of Time-Frequency Analysis

In the context of signal processing a simultaneous representation of temporal and spectral behavior seems necessary. If we consider a piece of music as a function of time $f(t)$, we get the temporal behavior and maybe the rhythmic patterns, but we don't know the tone pitch that is played. If we look in contrast at the Fourier transformation $\hat{f}(\omega)$, we get the frequencies of the prevailing notes, but we don't know anything about the duration of this note. Like our ear provides information about time and frequency at the same time, we want to obtain a function that is both dependent on time and on frequency, a function that imitates our ear.
Of course, trying to find an ideal time-frequency representation occurs some problems. The uncertainty principle states that "a function $f(t)$ and its Fourier transformation $\hat{f}(\omega)$ cannot be supported on arbitrarily small sets"[16, p. 26]. Means, if we want to determine the current frequency of our signal $f$ at time $t$, we need to observe the signal at least over a short period, for example $[t-\delta, t]$. This leads by the uncertainty principle a possibly dependence of $\hat{f}(\omega)$ on $\delta$ and moreover the support of this Fourier transformation cannot be small. So it doesn't make sense to assign the time $t$ a specific frequency.
For the next theorems we are using the notation of the book [15, p. 211–231] which means that we are going to use locally compact Abelian groups for more generality and to develop our theory. We write $\mathcal{G}$ as the locally compact Abelian group (LCA group). The space we use is defined as

$$L^2(\mathcal{G}) = \{f \text{ measurable} : \|f\|_2 = (\int_{\mathcal{G}} |f(x)|^2 \, dx)^{1/2} < \infty\}.$$

The dual group of $\mathcal{G}$ is denoted by $\hat{\mathcal{G}}$ and if this group is equipped with the right topology, it is again a LCA group. In this thesis we basically think about LCA groups like $\mathcal{G} = \mathbb{R}$ in the continuous case and like $\mathcal{G} = \mathbb{C}^L$ for the discrete case. Basically we will use the discrete case, because we do numerical computations.
For additional information of this chapter we refer to [16] and [15] as literature for Fourier and Gabor theory. Use [12] to obtain an overview about time-frequency theory and [7] to gain more insight of frame theory. But for the convenience of the reader, we include/conclude the main definitions and results in the following subsections.

## 2.1 The Short-Time Fourier Transform (STFT)

In this section we are first going to look at the linear and continuous representation of time-frequency analysis, the so called short-time Fourier transformation. We want to localize the function $f(t)$ by cutting it into pieces. Because these pieces are rarely periodic, some discontinuities occur when applying the Fourier transform. So the idea is a localization of the function by multiplying it with a smooth, means compactly supported window function.

**Definition 2.1** ([16, p. 37])**. (Short-Time Fourier Transform)**
*Let $g \in L^2(\mathcal{G})$ be a given window function, then the short-time Fourier transform (STFT) of a function $f \in L^2(\mathcal{G})$ with respect to $g$ is defined as*

$$V_g f(x, \omega) = \int_{\mathcal{G}} f(\tau)\overline{g(\tau - x)}e^{-2\pi i \tau \omega} d\tau \quad \text{for } x \in \mathcal{G}, \omega \in \hat{\mathcal{G}}.$$

In order to rewrite the STFT as an inner product, we introduce the translation and the modulation operator.

**Definition 2.2** ([16, p. 6])**. (Translation and Modulation)**
*Let $x, t \in \mathcal{G}$ and $\omega \in \hat{\mathcal{G}}$,*

- *then the translation (time shift) operator is defined as*

$$T_x f(t) = f(t - x).$$

- *then the modulation (frequency shift) operator is defined as*

$$M_\omega f(t) = e^{2\pi i t\omega} f(t).$$

*Then the operators of the form $T_x M_\omega$ or $M_\omega T_x$ are called time-frequency shifts.*

**Theorem 2.3** ([16, p. 6])**. (Commutation Relation)**
*Let $x \in \mathcal{G}$ and $\omega \in \hat{\mathcal{G}}$ then the following commutation relation hold*

$$T_x M_\omega f(t) = e^{-2\pi i x\cdot\omega} M_\omega T_x.$$

*Proof.* Let $f \in L^2(\mathcal{G})$, then

$$\begin{aligned} T_x M_\omega f(t) &= (M_\omega f)(t - x) \\ &= e^{2\pi i\omega\cdot(t-x)} f(t - x) \\ &= e^{-2\pi i x\cdot\omega} e^{2\pi i\omega\cdot t} f(t - x) \\ &= e^{-2\pi i x\cdot\omega} M_\omega T_x f(t). \end{aligned}$$

It follows that $T_x$ and $M_\omega$ commute if and only if $x \cdot \omega \in \mathbb{Z}$. □

Now we can rewrite our STFT as an inner product.

**Lemma 2.4** ([16, p. 39])**.** If $f, g \in L^2(\mathcal{G})$, then $V_g f : \mathcal{G} \times \hat{\mathcal{G}} \to \mathcal{G}$, with $\mathcal{G} = \mathbb{R}^d$, is uniformly continuous and can be written as

$$V_g f(x, \omega) = \langle f, M_\omega T_x g\rangle. \tag{2.1}$$

Because time-frequency analysis of signals is mostly done in three steps which we are going to state, the introduction of the inversion formula seems necessary.

**Corollary 2.5** ([16, p. 44])**. (Inversion formula for the STFT)**
Let $g, \gamma \in L^2(\mathcal{G})$ and $\langle \gamma, g\rangle \neq 0$. Then for all $f \in L^2(\mathcal{G})$

$$f = \frac{1}{\langle g, \gamma\rangle} \int_{\mathcal{G}} \int_{\hat{\mathcal{G}}} V_g f(x, \omega) M_\omega T_x \gamma \, d\omega dx.$$

*Proof.* Since $f, g \in L^2(\mathcal{G})$ it follows from the relation $\|V_g f\|_2 = \|f\|_2 \|g\|_2$ that $V_g f \in L^2(\mathcal{G} \times \hat{\mathcal{G}})$ (we are not going to proof this). Therefore the vector-valued integral

$$\tilde{f} = \frac{1}{\langle g, \gamma\rangle} \int_{\mathcal{G}} \int_{\hat{\mathcal{G}}} V_g f(x, \omega) M_\omega T_x \gamma \, d\omega dx$$

is well-defined in $L^2(\mathcal{G})$. If we use now the orthogonality relations [1], we obtain

$$\begin{aligned}\langle \tilde{f}, h\rangle &= \frac{1}{\langle g,\gamma\rangle}\int_{\mathcal{G}}\int_{\hat{\mathcal{G}}} V_g f(x,\omega)\overline{\langle h, M_\omega T_x \gamma\rangle} d\omega dx \\ &= \frac{1}{\langle g,\gamma\rangle}\langle V_g f, V_\gamma h\rangle = \langle f, h\rangle.\end{aligned} \tag{2.2}$$

It follows that $\tilde{f} = f$ and we have proved the inverse formula. □

Now we shortly mention the three steps the time-frequency analysis is basically consisting.

- Analysis-step: In this step the STFT $V_g f(x,\omega)$ of a given function $f$ will be computed and interpreted as a time-frequency representation.

$$f \mapsto V_g f$$

- Processing-step: Now the obtained STFT of our signal $f$ is going to get transformed into kind of a new function $F(x,\omega)$. In this step some signal processing happens.

$$V_g f \mapsto F$$

- Synthesis-step: Using a suitable synthesis window $\gamma$ the now processed signal will be reconstructed by the modified inversion formula:

$$\tilde{f} = \int_{\hat{\mathcal{G}}}\int_{\mathcal{G}} F(x,\omega) M_\omega T_x \gamma dx d\omega.$$

We have to point out that distinct windows for the analysis-step and the synthesis-step could and maybe have to be used. Although we should be satisfied with this achievements, the STFT leads to the disadvantage of high redundancy. This means that the supporting sets of two neighbor points have a large intersection, so their STFT contains almost the same information. Therefore the representation of the original function through the inversion formula is highly redundant.
Now we are going to reduce the redundancy by sampling the STFT. Sampling can be done in time and in frequency separately described in the following. We observe a band-limited window, filter with its modulated versions, therefore have a band-limited output and the function can be sampled in time. It follows that the STFT can be reconstructed from values we observe after discretization in time. Furthermore, if we consider a window of finite length, then each windowed section of our signal has finite length and therefore finite duration. Then we can again reconstruct our STFT, but now from its samples evaluated over the discretization in frequency. In general, because our function is dependent in both, time and frequency, we would prefer sampling in both components in the time-frequency plane to avoid the loss of information [12].

[1] Orthogonality relations [16, p. 43]:
Let $f_1, f_2, g_1, g_2 \in L^2(\mathcal{G})$, then $V_{g_j} f_j \in L^2(\mathcal{G}\times\hat{\mathcal{G}})$ for $j = 1, 2$ and

$$\langle V_{g_1} f_1, V_{g_2} f_2\rangle = \langle f_1, f_2\rangle\overline{\langle g_1, g_2\rangle}.$$

Imposing a set of points $\{(\tau_n, \nu_m) \mid n, m \in \mathbb{Z}\}$, the time-frequency shifts of our function along these points should span a dense subspace of $L^2(\mathcal{G})$. Because the so spanned plane is dense, $f$ can be uniquely reconstructed. Instead of equation (2.1) we observe the inner product

$$V_g f(\tau_n, \nu_m) = \langle f, M_{\nu_m} T_{\tau_n} g \rangle. \tag{2.3}$$

Since signal processing is mostly done in computers and therefore has to be digitalized, we are thinking of discrete signals with finite length $L$. This could be understood as a vector of $\mathbb{C}^L$, the complex vector space of dimension $L$ which is also a Hilbert space. But in general we don't want to calculate the inner product in every point like equation (2.3). This we would pay with a redundancy of $L$. So we are going to down sample our mesh, in time by $a$ and in frequency by $b$. This reduces our redundancy to $\frac{L}{ab}$ and leads to the following definition of a lattice.

**Definition 2.6** ([15, p. 216])**. (Lattice)**
*A subgroup $\Lambda$ of $(\mathcal{G} \times \hat{\mathcal{G}}, +)$ is called a lattice, if the quotient $(\mathcal{G} \times \hat{\mathcal{G}})/\Lambda$ is a compact group.*

*Remark* 2.7. As example we could observe $\mathcal{G} = \mathbb{R}$ and $\Lambda = \mathcal{A}\mathbb{Z}^2$. If $\mathcal{A}$ is diagonal, the lattice is called separable. A separable lattice can be written as $\Lambda = a\mathbb{Z} \times b\mathbb{Z}$ for $a, b > 0$.
If we observe $\mathcal{G} = \mathbb{C}^L$ we get a separable lattice considering $\Lambda = \mathbb{C}^{\frac{L}{a}} \times \mathbb{C}^{\frac{L}{b}}$ for $a, b > 0$.

With the definition of the lattice we are moving forward from Fourier expansion to Gabor expansion. But before defining Gabor frames, we need a short summary of frame theory.

## 2.2 Frame Theory

The reason why we want to improve the concept of frames, is because we want perfect reconstruction and frames generalize the concept of a basis. This means that all bases are frames, but a frame is not necessarily a basis. A further essential point is that a frame need not to be linear independent. On the other hand, if it is linear independent, it is more robust against noise or data loss. In this subsection we are using a general Hilbert space $\mathcal{H}$, as example we can keep $L^2(\mathcal{G})$ in mind.

**Definition 2.8** ([16, p. 85])**. (Frame, Frame Bounds, Tight Frame)**
*A sequence $(g_k)_{k=1}^{\infty}$ of elements in a Hilbert space $\mathcal{H}$ is called frame if there exist positive constants $A, B > 0$ such that for all $f \in \mathcal{H}$*

$$A\|f\|^2 \leq \sum_{k=1}^{\infty} |\langle f, g_k \rangle|^2 \leq B\|f\|^2. \tag{2.4}$$

*Any two constants $A, B$ satisfying equation (2.4) are called frame bounds. If $A = B$, then we call $(g_k)_{k=1}^{\infty}$ a tight frame.*

Note that $A > 0$ ensures that there are no zero projections of the signals on all representative elements. In addition we want to define Bessel sequences.

**Definition 2.9** ([7, p. 52])**. (Bessel Sequences)**
*A sequence $(g_k)_{k=1}^{\infty}$ of elements in a Hilbert space $\mathcal{H}$ is called Bessel sequence, if there exists at least $B > 0$ satisfying the right hand side of equation (2.4).*

For a better understanding of frames and reconstruction methods, we want to impose the following important operators.

**Definition 2.10** ([7, p. 51, 90])**. (Analysis-, Synthesis- and Frame operator)** *Let $(g_k)_{k=1}^{\infty}$ be a sequence in a Hilbert space $\mathcal{H}$ and $f \in \mathcal{H}$ then the coefficient operator or analysis operator $T : \mathcal{H} \to \ell^2(\mathbb{N})$ is defined as*

$$Tf = \Big(\langle f, g_k\rangle\Big)_{k=1}^{\infty}. \tag{2.5}$$

*Its adjoint $T^* : \ell^2(\mathbb{N}) \to \mathcal{H}$ is the synthesis operator or reconstruction operator and is defined for a finite sequence $c = (c_k)_{k=1}^{\infty}$ by*

$$T^*c = \sum_{k=1}^{\infty} c_k g_k. \tag{2.6}$$

*Combining now these two operators leads to the definition of the frame operator $S : \mathcal{H} \to \mathcal{H}$*

$$Sf = T^*Tf = \sum_{k=1}^{\infty} \langle f, g_k\rangle g_k. \tag{2.7}$$

**Lemma 2.11** ([7, p. 90])**. (Canonical dual frame)**
Let $(g_k)_{k=1}^{\infty}$ be a frame for $\mathcal{H}$ with frame bounds $0 < A, B < \infty$ and the frame operator $S : \mathcal{H} \to \mathcal{H}$. Then we get

i) $S$ is bounded, invertible, self-adjoint and positive definite.

ii) $\Big(S^{-1}g_k\Big)_{k=1}^{\infty}$ is a frame with bounds $B^{-1}$ and $A^{-1}$ and its frame operator is $S^{-1}$. Moreover it is called the canonical dual frame of $(g_k)_{k=1}^{\infty}$.

*Proof.* See [7, p. 90,91]. □

**Theorem 2.12** ([7, p. 95])**. (Canonical tight frame)**
*Let $(g_k)_{k=1}^{\infty}$ be a frame for $\mathcal{H}$ with frame operator $S$. Denote the positive square root of $S^{-1}$ by $S^{-1/2}$. Then $\Big(S^{-1/2}g_k\Big)_{k=1}^{\infty}$ is the canonical tight frame for $(g_k)_{k=1}^{\infty}$ with frame bounds equal to 1, and for all $f \in \mathcal{H}$*

$$f = \sum_{k=1}^{\infty} \langle f, S^{-1/2}g_k\rangle S^{-1/2}g_k. \tag{2.8}$$

*Proof.* See [7, p.95] □

**Theorem 2.13** ([7, p. 91])**.** *Let $(g_k)_{k=1}^{\infty}$ be a frame for $\mathcal{H}$ with frame operator $S : \mathcal{H} \to \mathcal{H}$, then*

$$f = \sum_{k=1}^{\infty} \langle f, S^{-1}g_k\rangle g_k, \quad \text{for all } f \in \mathcal{H}. \tag{2.9}$$

*The series converges unconditionally for all $f \in \mathcal{H}$.*

*Proof.* See[7, p.92]. □

A frame $(\gamma_k)_{k=1}^{\infty}$ is called dual frame for $(g_k)_{k=1}^{\infty}$, if it satisfies $f = \sum_{k=1}^{\infty}\langle f, \gamma_k\rangle g_k$. The canonical dual frame is the unique frame among all dual frames thats coefficients have minimal $\ell^2$-norm.

## 2.3 Gabor Theory

In the following subsection we introduce a short summery of Gabor theory, considering a time-frequency representation with less redundancy.

**Definition 2.14** ([16, p. 93])**. (Gabor System, Gabor Frame Operator)**
*Given a window function $0 \neq g \in L^2(\mathcal{G})$ and lattice parameters $a, b > 0$, the set of time-frequency shifted versions of $g$*

$$G(g, a, b) = \{T_{na} M_{mb} g : n, m \in \mathbb{Z}\}$$

*is called a Gabor system.*
*If $G(g, a, b)$ is a frame for $L^2(\mathcal{G})$, it is called a Gabor frame or Weyl-Heisenberg frame. The associated frame operator, the Gabor frame operator, has the form*

$$\begin{aligned} S_{g,a,b} f &= \sum_{n,m,\in\mathbb{Z}} \langle f, T_{na} M_{mb} g \rangle T_{na} M_{mb} g \\ &= \sum_{n,m\in\mathbb{Z}} V_g f(na, mb) T_{na} M_{mb} g. \end{aligned} \tag{2.10}$$

Usually the uniform sampling of STFT on a time-frequency lattice $\Lambda = \{(na, mb) \mid n, m \in \mathbb{Z}\}$ leads to Gabor systems. In this lattice the parameters $a$ and $b$ control the density of the grid points. This leads to a distinction in three cases [13, p. 13]:

- $ab < 1$ : This is the oversampling case, which guarantees the existence of frames with good time-frequency properties.
- $ab = 1$ : This is the critical sampling case. If there exist frames or orthonormal bases, they exist without good time-frequency localization.
- $ab > 1$ : This is called the under sampling case, because of incompleteness of any Gabor family, one cannot have a frame of $L^2(\mathcal{G})$. This case can also be compared to a Nyquist criterion for Gabor systems.

Furthermore we are introducing a functional space, called Feichtinger algebra. The functions of this space have nice localization and smoothness properties and therefore this function space plays a fundamental role in time-frequency analysis. More information about the Feichtinger algebra can be found in [16] in Chapter 11 and 12 and for more associated properties view Chapter 3 of [14]. In our case this functional space helps to choose the windowed functions properly and in this context we refer to [18].

**Definition 2.15** ([18, p. 25])**. (Feichtinger Algebra)**
*Let $g$ be a Gaussian window $g(t) = e^{-\pi t^2}$. The Feichtinger algebra $S_0(\mathcal{G})$ is defined as*

$$S_0(\mathcal{G}) = \{f \in \mathcal{S}' : \|V_g f\|_{L^1(\mathcal{G}\times\hat{\mathcal{G}})} < \infty\},$$

*where $\mathcal{S}$ is the Schwartz space of the functions that are infinitely often differentiable and rapidly decreasing. Here $\mathcal{S}'$ is the dual space, the space of temperated distributions.*

If we choose $g \in S_0(\mathcal{G})$, we safely get a pair of parameters $(a, b)$ to obtain a Gabor frame $G(g, a, b)$. Moreover $g \in S_0(\mathcal{G})$ if and only if $\hat{g} \in S_0(\mathcal{G})$.
The following proposition implies why Gabor theory is so useful. This is due to the fact that the dual frame of a Gabor frame is again a Gabor frame generated by the same lattice and by the dual window.

**Proposition 2.16** ([16, p. 94])**. (Dual Gabor frame)**
If $G(g,a,b)$ is a frame for $L^2(\mathcal{G})$, then there exists a dual window $\gamma \in L^2(\mathcal{G})$, such that the dual frame of $G(g,a,b)$ is $G(\gamma,a,b)$. Therefore, every $f \in L^2(\mathcal{G})$ has the expansions

$$\begin{aligned} f &= \sum_{n,m\in\mathbb{Z}} \langle f, T_{na}M_{mb}g\rangle T_{na}M_{mb}\gamma \\ &= \sum_{n,m\in\mathbb{Z}} \langle f, T_{na}M_{mb}\gamma\rangle T_{na}M_{mb}g \end{aligned}$$

with unconditional convergence in $L^2(\mathcal{G})$.

*Proof.* In the following we are writing $S$ instead of $S_{g,a,b}$, the Gabor frame operator of (2.10).
Now we show that the Gabor frame operator $S$ commutes with time-frequency shifts $T_{na}M_{mb}$. For a given function $f \in L^2(\mathcal{G})$ and $r,s \in \mathbb{Z}$,

$$\Big(T_{ra}M_{sb}\Big)^{-1} S T_{ra}M_{sb} f = \sum_{n,m\in\mathbb{Z}} \langle T_{ra}M_{sb}f, T_{na}M_{mb}g\rangle \Big(T_{ra}M_{sb}\Big)^{-1} T_{na}M_{mb}g.$$

By commutativity, we get

$$\Big(T_{ra}M_{sb}\Big)^{-1}\Big(T_{na}M_{mb}\Big) = e^{-2\pi iab(n-r)s} T_{a(n-r)}M_{b(m-s)}.$$

Then the phase factor $e^{-2\pi iab(n-r)s}$ vanishes and after renaming the indices we get the equation

$$\Big(T_{ra}M_{sb}\Big)^{-1} S T_{ra}M_{sb} f = \sum_{n,m\in\mathbb{Z}} \langle f, T_{a(n-r)}M_{b(m-s)}g\rangle T_{a(n-r)}M_{b(m-s)}g = Sf.$$

Hence $S^{-1}$ also commutes with $T_{ra}M_{sb}$ and the dual frame is assembled as

$$S^{-1}(T_{na}M_{mb}) = T_{na}M_{mb}S^{-1}g.$$

This leads to the choice of $\gamma = S^{-1}g$ as dual window. □

**Corollary 2.17** ([16, p. 95])**. (Inverse frame operator)**
If $G(g,a,b)$ is a frame for $L^2(\mathcal{G})$ with dual window $\gamma = S^{-1}g \in L^2(\mathcal{G})$, then the inverse frame operator is given by

$$S^{-1}f = \sum_{n,m\in\mathbb{Z}} \langle f, T_{na}M_{mb}\gamma\rangle T_{na}M_{mb}\gamma. \tag{2.11}$$

*Proof.* See [16, p.95]. □

# 3 Formulation of the Problem

After discussing the basic theory in the preceding chapter, we want to introduce the central problem this master thesis is going to treat. In general we are looking at two given signals, a source or input signal $s$ and a target or output signal $z$. We assume that these signals deal with sufficient similarity in the time-frequency plane, i.e. we consider for example two different instruments which play the same note. Our aim is to find, or rather to estimate, the transfer function between these two sounds.
After introducing the theory to our problem, we are going to do some experiments considering different tone pitch and different volume in Chapter 8. But now we introduce linear time-varying filters (LTV), show that they are linear operators and in the following we will specify on Gabor multipliers.

## 3.1 Linear Time-Varying (LTV) Filter

In many applications it is practicable to use the design of linear time-varying filters for example for weighting, suppressing or separating non-stationary signal components. In this subsection, we follow Chapter 11 of [5]. Considering a general model with $s \in L^2(\mathcal{G})$ as the input function and $z \in L^2(\mathcal{G})$ as the output, yields

$$z(t) = (Hs)(t) = \int_{-\infty}^{\infty} h(t,\tau)s(\tau)d\tau,$$

where $H : L^2(\mathcal{G}) \to L^2(\mathcal{G})$ is the LTV filter and $h(t,\tau)$ its kernel. Since the input, the output and our filter $H$ are non-stationary, we may wish to use time-frequency representations for the creation, analysis or implementation of such a LTV filter. Before we briefly discuss two fundamentally different approximations to the time-frequency presentation of LTV filters and finally concentrate on one of these two, we want to introduce the time-frequency weight function $w(t,\omega) \in L^\infty(\mathcal{G} \times \hat{\mathcal{G}})$ which helps us giving some elements more weight.

- We have the possibility of an explicit design of the LTV filter $H$ by using the *generalized Weyl symbol* or alternatively the *Wigner distribution of an LTV system*, cf. [5, p. 467].
- The second approach which we are going to look at, is the implicit design of the LTV filter, where the filter $H$ is designed in a three-step analysis-processing-synthesis procedure, means the filter is designed during filtering.

We now give a rough description of these three steps and go more into detail, explaining the steps with Gabor filters. The first step is the analysis step, where the time-frequency representation of the input signal is computed. In the second step we are going to multiply the previously calculated time-frequency representation with a weight function $w(t,\omega)$ (or any other signal processing function) and in the third step, we calculate the output from the second step by synthesis. Because all these processing steps are linear, the entire process results in an LTV filter.
Compared to the preceding chapter (p. 4), we now consider the Gabor transform instead of the STFT. Therefore we are discussing the Gabor filter on the lattice $\Lambda = \{(na, mb) \mid$

$n, m \in \mathbb{Z}\}$, with parameters $a, b > 0$. This means that we are going to use the Gabor transform to determine the time-frequency coefficients.

- Analysis step: Here we calculate the Gabor coefficients of the source signal $s \in L^2(\mathcal{G})$
$$c_{n,m} = \sum_{m,n\in\mathbb{Z}} \langle s, T_{na} M_{mb} g\rangle = \sum_{m,n\in\mathbb{Z}} V_g s(na, mb).$$

- Processing step: In this step we multiply the Gabor coefficients by the weights $w_{n,m} = w(na, mb)$ and we get
$$w_{n,m} \cdot c_{n,m}.$$
(But it also could be done some other signal processing procedure.)

- Synthesis step: Now we obtain the target signal $z \in L^2(\mathcal{G})$ by Gabor synthesis
$$z = \sum_{m,n\in\mathbb{Z}} w_{n,m} \cdot \langle s, T_{na} M_{mb} g\rangle T_{na} M_{mb} \gamma.$$

This Gabor filter obtained through the previous three steps depends on the weight function, the analysis and the synthesis window. Usually we want the analysis and the synthesis windows $g$ and $\gamma$ to satisfy the perfect reconstruction as defined in the Preliminaries as dual windows. Furthermore we need to ensure $ab = 1$ or $ab < 1$, where we can obtain a Gabor system (cf. Subsection 2.3 p. 7).
In the next subsection we are going to consider special LTV filters, called Gabor multipliers.

## 3.2 Gabor Multipliers

Going more into detail, for Gabor transform we have the possibility of constructing a time-varying filter by the usage of Gabor multipliers instead of the weighting function. For further information compare [2] and [19]. Considering the context of the three processing steps in the last subsection described by analysis - processing/transformation - synthesis, the signal transformation can be explained as a pointwise multiplication of the analysis coefficients with a weighting respectively transfer function in a given representation space. This kind of transformation is now called multiplier.
As mentioned in the previous subsection, the Gabor transform is used for the analysis step, where we compute the time-frequency coefficients and before synthesizing the result, the analysis coefficients are multiplied with a fixed time-frequency mask.

**Definition 3.1** ([2, p. 577])**. (Gabor multiplier)**
*Let $\mathcal{H}_1, \mathcal{H}_2$ be Hilbert spaces, let $(g_\ell)_{\ell\in\Lambda} \subseteq \mathcal{H}_1$ and $(\gamma_\ell)_{\ell\in\Lambda} \subseteq \mathcal{H}_2$ be Gabor frames. Fix a sequence $m = (m_\ell)_{\ell\in\Lambda} \in L^\infty(\mathcal{G} \times \hat{\mathcal{G}})$, then we define the operator*

$$\mathbb{M}_{m;g,\gamma} : \mathcal{H}_1 \to \mathcal{H}_2,$$

*the Gabor multiplier for the Gabor frames $(g_\ell)$ and $(\gamma_\ell)$, as the operator*

$$\mathbb{M}_{m;g,\gamma}(f) = \sum_\ell m_\ell \langle f, g_\ell\rangle \gamma_\ell.$$

The sequence $(m_\ell)_{\ell\in\Lambda}$ mentioned in this definition is called the *symbol mask* of $\mathbb{M}$ and can be interpreted as a time-frequency transfer function. Define $\Upsilon_m$ as the linear pointwise operator of the sequence $(m_\ell)_{\ell\in\Lambda}$ with function $K$ , i.e.

$$\Upsilon_m(K)[\ell] = K[\ell]m_\ell$$

at position $\ell$, we can rewrite the Definition 3.1 from above as

$$\mathbb{M}_{m;g,\gamma} = V_\gamma^* \circ \Upsilon_m \circ V_g.$$

$\mathbb{M}_{m;g,\gamma}$ is now a linear operator on the space from which we are taking our signals. Moreover it is a diagonal operator in the representation of $(g_\ell)_{\ell\in\Lambda}$ of the Gabor frame. In the following we consider the sequence $m$ as bounded and complex valued and therefore $(m_\ell)_{\ell\in\Lambda} \in L^\infty(\mathcal{G}\times\hat{\mathcal{G}})$. The discrete group $\mathcal{G} = \mathbb{C}^L$ implies $L^\infty(\mathcal{G}\times\hat{\mathcal{G}}) = \ell^\infty$. Then it follows that the corresponding multiplier is a bounded operator (cf. [2, p. 579]). Now our problem is not computing the output through a given input and Gabor mask, but the corresponding inverse problem, i.e. the identification of the system in the time-frequency plane. In the following subsection we are trying to approximate the identification problem of the linear system. First in a general way, using the filter $H$ and then we consider again Gabor multipliers.

## 3.3 Identification of the Linear Time-Varying System

The following subsection is mainly based on [18]. Given a source signal $s \in L^2(\mathcal{G})$ and a given target signal $z \in L^2(\mathcal{G})$, we want to look for the linear system, such that $z = Hs$. For the sake of simplicity we suppose the filter $H$ to be diagonal in the time-frequency plane. Because we are considering LTV systems, we have something that looks like

$$z(t) = \int_{\mathcal{G}} h(t, t-\tau)s(\tau)d\tau,$$

depending on time. Here $h(t, t-\tau)$ denotes the kernel of $H$, as in Subsection 3.1. Then the Gabor transformation of the target sound $z$ is given as

$$V_g z(na, mb) = \int_{\mathcal{G}}\int_{\mathcal{G}}\int_{\hat{\mathcal{G}}} \hat{h}_2(t,\xi)e^{2\pi i\xi(t-\tau)}\langle s(t), T_{na}M_{mb}g(t)\rangle d\xi d\tau dt. \tag{3.1}$$

In this formula $\hat{h}_2$ is the Fourier transform of $h$ with respect to the second coordinate. Because it seems that we are far away from a pointwise multiplication and the operator $h$ looks difficultly connected with source signal $s$, see Formula (3.1), we use Taylor expansion. Then we obtain

$$V_g z(na, mb) = \hat{h}_2(na, mb)V_g s(na, mb) + \mathcal{R}_1(na, mb) + \mathcal{R}_2(na, mb),$$

where we can estimate the remainders for all $(na, mb) \in \mathbb{Z}^2$ by

$$|\mathcal{R}_1| \le \|s\|_2 \left\| \frac{\partial \hat{h}_2}{\partial t} \right\|_\infty \left( \int_{\mathbb{R}} |tg(t)|^2 dt \right)^{1/2},$$

$$|\mathcal{R}_2| \le \|\hat{s}\|_2 \left\| \frac{\partial \hat{h}_2}{\partial \xi} \right\|_\infty \left( \int_{\mathbb{R}} |\xi\hat{g}(t)|^2 d\xi \right)^{1/2}.$$

As approximation of the Gabor transformation we therefore get

$$V_g z(na, mb) \approx \hat{h}_2(na, mb) V_g s(na, mb).$$

For simplicity we use the same window for analysis and synthesis, but the error estimation is also valid, if we use different windows. The estimation done so far is quite simple to handle, but the approximation of a Gabor multiplier looks quite different, because the Gabor mask strongly depends on the input and the output signal.

## 3.4 Estimation of the Gabor Multiplier

In the following subsection, we consider a normed tight Gabor frame $G(g, a, b)$ (means frame bounds $A = B = 1$) where $g, \gamma \in \mathcal{S}_0(\mathcal{G})$, and parameters $a, b > 0$. Assume the input and output signal $s, z \in L^2(\mathcal{G})$ to be given and the relation

$$z = \mathbb{M}_{m;g,\gamma} s$$

to be valid. Now we want to identify the linear system, where the system is treated as a Gabor multiplier. It is obvious that if the transformation into the time-frequency plane $V_g s$ has no zero entries, a solution for $\mathbb{M}_{m;g,\gamma}$ is of course $\frac{V_g z}{V_g s}$. But such a solution can be very oscillating and difficult to interpret. Furthermore the redundancy of the Gabor transformation leads to loss of uniqueness of the solution, cf. [18, p. 40]. In the following subsection we are going to introduce the inverse problem. We can reformulate optimality as the minimization of a functional and its estimation can therefore be transformed into a linear inverse problem.

### 3.4.1 Formulation of the Inverse Problem

For the formulation of our inverse problem, we are using the notation of [18] and also refer to [19] and [20]. Since $\mathbb{M}_{m;g,\gamma}$ is a linear operator assigned to $m$, we can introduce an operator defined as

$$\begin{aligned} \mathcal{O} : L^\infty(\mathcal{G} \times \hat{\mathcal{G}}) &\to L^2(\mathcal{G}) \\ m &\mapsto \mathcal{O}m = V_\gamma^* \circ \Upsilon_{V_g s} = V_\gamma^*(m V_g s). \end{aligned} \tag{3.2}$$

As mentioned previously $\Upsilon_k : L^\infty(\mathcal{G} \times \hat{\mathcal{G}}) \to L^\infty(\mathcal{G} \times \hat{\mathcal{G}})$ is a diagonal operator which preserves a pointwise multiplication by a bounded sequence $k$. This construction yields

$$\mathbb{M}_{m;g,\gamma} s = \mathcal{O}m.$$

Then we can define the adjoint operator which looks like

$$\begin{aligned} \mathcal{O}^* : L^2(\mathcal{G}) &\to L^\infty(\mathcal{G} \times \hat{\mathcal{G}}) \\ x &\mapsto \mathcal{O}^* x = \Upsilon_{\overline{V_g s}} \circ V_\gamma = \overline{V_g s} V_\gamma x. \end{aligned}$$

Concluding we want to state that the linear operator $\mathcal{O}$ depends on the source $s$ and does not have to be invertible in general. Considering now the problem of the identification of the system, we want to estimate the transfer function of the optimal multiplier which connects our input and output signal. This optimality is given by the

minimization of a functional and in that case we want to minimize a kind of regularized quadratic error, see equation (3.3). This problem is now stated as a regularized least squares problem and will be reformulated in the following as a linear inverse problem. The model we are going to look at has the form:

$$z = \mathcal{O}m + \varepsilon,$$

where $\varepsilon$ is a perturbation, kind of a white Gaussian noise. What we want to solve is a problem of the form

$$\arg\min_{\mathrm{m}} \|z - \mathcal{O}m\|_2^2. \tag{3.3}$$

An existing solution does not have to be unique. Only the Moore-Penrose pseudoinverse of $\mathcal{O}$ will give us the solution of the minimal norm, but computing this is very expensive. Searching a Gabor mask as solution of regularized inverse problem can yield to existence and uniqueness of a solution. We therefore have to minimize the expression

$$\Phi(m) = \|z - \mathcal{O}m\|_2^2 + \lambda r(m), \tag{3.4}$$

where $r(m) : L^\infty(\mathcal{G} \times \hat{\mathcal{G}}) \to \mathbb{R}^+$ is a regularization term and $\lambda \in \mathbb{R}^+$ is a regularization parameter. This parameter controls the balance between the regularization term which is supporting the functional with an additive a priori knowledge of the solution and the reconstruction properties of the Gabor multiplier. $\lambda$ is in general difficult to choose, because it depends on the given signals $s$ and $z$.
Addressing now the finite dimensional case $\mathcal{H} = \mathbb{C}^L$ and $\mathcal{G} = \mathbb{C}^L$, we obtain a linear problem and in case of $r(m) = \|m\|_2^2$ we can write $m$ as the solution of

$$(\mathcal{O}^*\mathcal{O} + \lambda I)m = \mathcal{O}^* z.$$

Here the gradient of the regularization term is chosen as $\nabla r(m) = 2m$. Normally such a solution is impossible to use, because it implies the inversion of a huge matrix system. For example, if we use a signal of length $L = 2^{15}$, which corresponds to a signal of duration $0,75$ seconds sampled with 44100 Hertz. Furthermore, if we choose for the Gabor parameters which we use in the algorithm of Chapter 5, $M = 1024$ and $a = 256$, we get a Matrix $\mathcal{O}$ of size $L \times MN \approx 2^{15} \times 2^{18}$.
In the next section we discuss some possibilities, such that we obtain an explicit solution for equation (3.4). One form of simplification we can do, is an approximation of the solution, by considering only the diagonal entries of the matrix $\mathcal{O}^*\mathcal{O}$. In the following we are going to look at such a special case of matrix, where we can discuss different choices of the regularization term $r(m)$.

# 4 Diagonal Approximation

For the sake of simplicity and to get a better understanding, we are now looking at the diagonal case which brings us to closed-form solutions. These solutions can be of acceptable quality for example in experiments on audio signals. Later on (Chapter 6) we will use some iterative shrinkage algorithms that converge to the exact solution of equation 3.4.
For some further information we refer to [10], [18], [19] and [11]. Back to theory we can reformulate (3.4) directly in the Gabor domain

$$\Phi(m) = \|V_g^*(V_g z - mV_g s)\|_2^2 + \lambda r(m).$$

Because $V_g^*$ is not one to one, this expression cannot be uniquely minimized. Alternatively we can do a first approach and use the minimizer of the functional

$$\Phi(m) = \|(V_g z - mV_g s)\|_2^2 + \lambda r(m).$$

Defining $S$ as the Gabor transformation of $s$ and $Z$ as the transformation of $z$ we get

$$\Phi(m) = \|Z - m \cdot S\|_2^2 + \lambda r(m). \tag{4.1}$$

If the source and the target are the same, the mask $m$ is equal to 1, i.e. each vector entry is 1 and the regularization term at the end disappears. But if they are different, we can use different terms of regularization. The choice of $r(m)$ depends on the variables that are given and on the problem we want to solve, because the regularization terms have different properties which are useful in different tonal situations, discussed in Remark 4.2 and moreover in Chapter 8. Furthermore this regularization term helps us to indicate some a priori information in the shape of the solution (the transformed signal). The parameter $\lambda$ helps balancing between these a priori information of the form and the properties of reconstructing the mask [18]. We are now going to present different choices of regularization terms by stating the following theorem.

**Theorem 4.1.** *Let $\Phi : \mathbb{C}^L \to \mathbb{R}$ be a functional of the form*

$$\Phi(m) = \|V_g z - mV_g s\|_2^2 + \lambda r(m), \tag{4.2}$$

*where $\lambda \in \mathbb{R}^+$ and $r : \mathbb{C}^L \to \mathbb{R}$ is a regularization term. Minimizing this functional yields different solutions for different regularization terms (consider $Z = V_g z$ and $S = V_g s$):*
*a) $r(m) = \|m - 1\|_2^2$ leads to the solution*

$$\tilde{m_\ell} = \frac{\overline{S_\ell} Z_\ell + \lambda}{|S_\ell|^2 + \lambda} \qquad \forall \ell \in \{0, ..., L\}.$$

*b) $r(m) = \| \, |m| - 1\|_2^2$ leads to the solution*

$$\tilde{m_\ell} = \frac{|Z_\ell S_\ell| + \lambda}{|S_\ell|^2 + \lambda} \cdot e^{i \arg(Z_\ell \overline{S_\ell})} \qquad \forall \ell \in \{0, ..., L\} \quad \textit{if } |Z_\ell S_\ell| > \lambda.$$

*c)* $r(m) = \|m-1\|_1$ *leads to the solution*

$$\tilde{m_\ell} = \frac{|S_\ell|\,|Z_\ell - S_\ell| - \frac{\lambda}{2}}{|S_\ell|^2} \cdot e^{i\arg(\overline{S_\ell}(Z_\ell - S_\ell))} + 1$$

$\forall \ell \in \{0, ..., L\}$ *if* $|S_\ell|\,|T_\ell - S_\ell| > \frac{\lambda}{2}$.

*d)* $r(m) = \|\,|m| - 1\|_1$ *leads to the solution*

$$\tilde{m_\ell} = \begin{cases} \frac{|Z_\ell S_\ell| - \frac{\lambda}{2}}{|S_\ell|^2} e^{i\arg(\overline{S_\ell} Z_\ell)} & \text{if } \frac{|Z_\ell S_\ell|}{|S_\ell|^2} > 1 + \frac{\lambda}{2|S_\ell|^2} \\ \frac{|Z_\ell S_\ell| + \frac{\lambda}{2}}{|S_\ell|^2} e^{i\arg(\overline{S_\ell} Z_\ell)} & \text{if } \frac{|Z_\ell S_\ell|}{|S_\ell|^2} < 1 - \frac{\lambda}{2|S_\ell|^2} \\ 1 & \text{else} \end{cases}$$

$\forall \ell \in \{0, ..., L\}$.

Before giving the proof of this theorem we discuss some properties of the different regularization terms. In the next chapter we will visualize these properties by analyzing an example for diagonal approximation with different regularization terms.

*Remark* 4.2. Let $\Phi : \mathbb{C}^L \to \mathbb{R}$ be the function from Theorem 4.1 above which will be minimized. Then the different regularization terms have the following properties:

a) $r(m) = \|m-1\|_2^2$ helps us to control the total energy. Moreover, if we use normed tight frame bounds, i.e. $A = B = 1$, we are going to favor a Gabor multiplier close to the identity operator. This regularization term is producing spurious oscillations in the Gabor mask which is caused by a bad estimation of the phase. A simple calculation shows the reason of the oscillations. Let $(j, k)$ be a point of the time-frequency plane and let the input and the output signal have a phase difference of $\pi$, i.e. $Z = Se^{i\pi}$. Then the Gabor mask at the point $(j, k)$ is given by

$$\tilde{m} = \frac{\overline{S}Z + \lambda}{|S|^2 + \lambda} = \frac{|S|^2 e^{i\pi} + \lambda}{|S|^2 + \lambda}.$$

This short calculation shows the presence of amplitude modulations of the mask due to the diagonal approximation [18, p. 43 et seq.].

b) $r(m) = \|\,|m| - 1\|_2^2$ gives us the possibility of avoiding spurious oscillations of the amplitude of the Gabor mask, apart from that fact it has the same properties as the previous regularization term in a).

c) $r(m) = \|m-1\|_1$ yields sparsity, where the mask is forced to stay close to 1 which corresponds to "no transformation". This regularization term also produces spurious oscillations.

d) $r(m) = \|\,|m| - 1\|_1$ forces the Gabor mask to sparsity of the deviation from the absolute value 1 and also avoids the oscillations of the previous regularization term in c). For some more information on this regularization term consider [11].

*Proof of Theorem* 4.1.

a) Considering $r(m) = \|m-1\|_2^2$ as regularization term, (4.2) becomes

$$\Phi(m) = \|Z - m \cdot S\|_2^2 + \lambda \cdot \|m-1\|_2^2. \tag{4.3}$$

Our aim is to compute the "best" fitting mask $m$ as $\tilde{m} = \min_m \Phi(m)$. We now rewrite the $L_2$-norm as the sum of all entries squared, then our functional $\Phi(m)$ from (4.3) can be expressed as

$$\Phi(m) = \sum_{\ell \in \Lambda} \Phi^\ell(m_\ell) = \sum_{\ell \in \Lambda} \Big( |Z_\ell - m_\ell S_\ell|^2 + \lambda \, |m_\ell - 1|^2 \Big). \tag{4.4}$$

Because we are in a complex vector space, we can rewrite the last equation (4.4) as

$$\sum_{\ell \in \Lambda} \Phi^\ell(m_\ell) = \sum_{\ell \in \Lambda} \Big( (Z_\ell - m_\ell S_\ell)\overline{(Z_\ell - m_\ell S_\ell)} + \lambda (m_\ell - 1)\overline{(m_\ell - 1)} \Big).$$

To get the minimum of this functional, we have to set the first derivative equal to zero. Since $m$ can be a complex vector, we need to use the following approach. Our function $\Phi(m)$ has its solution in a real vector space and so we can split $m$ as:

$$m = m^r + i \, m^i,$$

where $m^r$ and $m^i$ are real-valued vectors. This yields the expression

$$\begin{aligned} \sum_{\ell \in \Lambda} \Phi^\ell(m_\ell) = \sum_{\ell \in \Lambda} \Big( & (Z_\ell - S_\ell(m_\ell^r + i m_\ell^i))\overline{(Z_\ell - S_\ell(m_\ell^r + i m_\ell^i))} + \\ & + \lambda (m_\ell^r + i m_\ell^i - 1)\overline{(m_\ell^r + i m_\ell^i - 1)} \Big). \end{aligned}$$

We can now look at $(m^r, m^i) \mapsto \Phi(m) = \Phi(m^r, m^i)$ mapping from $\mathbb{R}^{2L} \mapsto \mathbb{R}^L$. This leads to the approach to understand the derivation as a derivative in two variables. We can then use the formula

$$\frac{\partial \Phi(m)}{\partial m} = \frac{1}{2} \left( \frac{\partial \Phi(m^r, m^i)}{\partial m^r} - i \frac{\partial \Phi(m^r, m^i)}{\partial m^i} \right) \tag{4.5}$$

and therefore get the derivative for holomorphic functions. For the sake of simplicity we fix one $\ell$, because if we take the derivative componentwise, the other components will vanish. So we erase the sum over all elements from $\Lambda$ and start the derivative with respect to the first variable $m_\ell^r$ :

$$\begin{aligned} \frac{\partial \Phi^\ell(m_\ell^r, m_\ell^i)}{\partial m_\ell^r} = & (-S_\ell)\overline{(Z_\ell - S_\ell(m_\ell^r + i m_\ell^i))} + \overline{(-S_\ell)}(Z_\ell - S\ell(m_\ell^r + i m_\ell^i)) \\ & + \lambda \overline{(m_\ell^r + i m_\ell^i - 1)} + \lambda (m_\ell^r + i m_\ell^i - 1) \\ = & (-S_\ell \overline{Z_\ell}) + |S_\ell|^2 (m_\ell^r - i m_\ell^i) + (-\overline{S_\ell} Z_\ell) + |S_\ell|^2 (m_\ell^r + i m_\ell^i) \\ & + \lambda (m_\ell^r - i m_\ell^i - 1 + m_\ell^r + i m_\ell^i - 1). \end{aligned}$$

Now we use that for complex numbers the equation $\frac{z + \overline{z}}{2} = \mathfrak{Re}(z)$ is valid and so we get

$$\frac{\partial \Phi^\ell(m_\ell^r, m_\ell^i)}{\partial m_\ell^r} = -2\mathfrak{Re}\Big(S_\ell \overline{Z_\ell}\Big) + 2 \, |S_\ell|^2 \, m_\ell^r + \lambda (2 m_\ell^r - 2).$$

The whole thing works similar for the derivative with respect to $m_\ell^i$ with one small difference. There appears the complex number $i$ in each term and therefore we

obtain

$$\begin{aligned}\frac{\partial \Phi^\ell(m_\ell^r, m_\ell^i)}{\partial m_\ell^i} =& (-iS_\ell)\cdot\overline{(Z_\ell - S_\ell(m_\ell^r + im_\ell^i))} + i\overline{(S_\ell)}(Z_\ell - S_\ell(m_\ell^r + im_\ell^i)) \\ &+ i\lambda\Big(\overline{(m_\ell^r + im_\ell^i - 1)} - (m_\ell^r + im_\ell^i - 1)\Big) \\ &= (-iS_\ell\overline{Z_\ell}) + i\left|S_\ell\right|^2(m_\ell^r - im_\ell^i) + i(\overline{S_\ell}Z_\ell) \\ &- i\left|S_\ell\right|^2(m_\ell^r + im_\ell^i) + i\lambda(m_\ell^r - im_\ell^i - 1 - m_\ell^r - im_\ell^i + 1).\end{aligned}$$

It should be noted that there are several sign differences to the case before. Applying now the fact that $\frac{z-\overline{z}}{2i} = \mathfrak{Im}(z)$, the whole computation turns into

$$= 2\cdot\mathfrak{Im}\Big(S_\ell\overline{Z_\ell}\Big) + 2\left|S_\ell\right|^2 m_\ell^i + 2\lambda m_\ell^i,$$

where the sign change is due to the multiplication of the imaginary identity with itself. Applying now equation (4.5) we obtain

$$\begin{aligned}\frac{\partial \Phi^\ell(m_\ell)}{\partial m_\ell} =& \frac{1}{2}\Bigg(-2\mathfrak{Re}\Big(S_\ell\overline{Z_\ell}\Big) + 2\left|S_\ell\right|^2 m_\ell^r + \lambda(2m_\ell^r - 2) \\ &- 2i\mathfrak{Im}\Big(S_\ell\overline{Z_\ell}\Big) - 2i\left|S_\ell\right|^2 m_\ell^i - 2i\lambda m_\ell^i\Bigg).\end{aligned}$$

Because we want to compute the minimum, we have to set this equation to zero

$$\frac{\partial \Phi^\ell(m_\ell)}{\partial m_\ell} = -S_\ell\overline{Z_\ell} + \left|S_\ell\right|^2\overline{m_\ell} + \lambda\overline{m_\ell} - \lambda = 0.$$

Solving with respect to $\overline{m_\ell}$, we get

$$\overline{m_\ell} = \frac{S_\ell\overline{Z_\ell} + \lambda}{\left|S_\ell\right|^2 + \lambda}.$$

But we want the solution with respect to $m_\ell$, therefore we have to take the conjugate. Then the solution of the problem (4.3) reads for every $\ell$

$$\tilde{m}_\ell = \frac{\overline{S_\ell}Z_\ell + \lambda}{\left|S_\ell\right|^2 + \lambda}.$$

In this solution we can see that the parameter $\lambda$ is going to balance the role of the numerator and the denominator.

b) With regularization term $r(m) = \|\,|m| - 1\|_2^2$ we state the computation for

$$\Phi(m) = \|Z - m\cdot S\|_2^2 + \lambda\cdot\|\,|m| - 1\|_2^2. \tag{4.6}$$

Because some things are similar to the case a) before we skip a few computational steps. After rewriting the norms as sums of absolute value squared and considering the case only for one fixed $\ell$, we get

$$\begin{aligned}\Phi^\ell(m_\ell) &= (Z_\ell - m_\ell S_\ell)\overline{(Z_\ell - m_\ell S_\ell)} + \lambda(|m_\ell| - 1)\overline{(|m_\ell| - 1)} \\ &= \left|Z_\ell\right|^2 - 2\mathfrak{Re}(\overline{Z_\ell}S_\ell m_\ell) + \left|S_\ell\right|^2\left|m_\ell\right|^2 + \lambda(\left|m_\ell\right|^2 - 2\left|m_\ell\right| + 1)\end{aligned}$$

As in the case before, our function $\Phi(m)$ has its solution in a real vector space and so we can split $m$ into $m = m^r + i\,m^i$ and obtain

$$\begin{aligned} &= |Z_\ell|^2 - 2\mathfrak{Re}(\overline{Z_\ell}S_\ell)m_\ell^r + 2\mathfrak{Im}(\overline{Z_\ell}S_\ell)m_\ell^i \\ &\quad + |S_\ell|^2 |m_\ell|^2 + \lambda(|m_\ell|^2 - 2|m_\ell| + 1), \end{aligned}$$

where $|m_\ell|^2 = ((m_\ell^r)^2 + (m_\ell^i)^2)$ and $|m_\ell| = \sqrt{(m_\ell^r)^2 + (m_\ell^i)^2}$. Then we can compute the derivative with respect to $m^r$ and $m^i$:

$$\frac{\partial \Phi^\ell(m_\ell^r, m_\ell^i)}{\partial m_\ell^r} = -\mathfrak{Re}(\overline{Z_\ell}S_\ell) + 2|S_\ell|^2 m_\ell^r + 2\lambda m_\ell^r - 2\frac{\lambda m_\ell^r}{|m_\ell|}$$

$$\frac{\partial \Phi^\ell(m_\ell^r, m_\ell^i)}{\partial m_\ell^i} = \mathfrak{Im}(\overline{Z_\ell}S_\ell) + 2|S_\ell|^2 m_\ell^i + 2\lambda m_\ell^i - 2\frac{\lambda m_\ell^i}{|m_\ell|}.$$

We now have to pay attention, because we divide by $|m_\ell|$ and so we have to consider the restriction

$$|m_\ell| > 0 \tag{4.7}$$

Applying the Formula (4.5) we obtain

$$\frac{\partial \Phi^\ell(m_\ell)}{\partial m_\ell} = -\overline{Z_\ell}S_\ell + |S_\ell|^2 \overline{m_\ell} + \lambda\overline{m_\ell} - \frac{\lambda\overline{m_\ell}}{|m_\ell|}$$

which should be zero. Solving this equation with respect to $\overline{m_\ell}$ we have

$$\overline{m_\ell} = \frac{\overline{Z_\ell}S_\ell}{|S_\ell|^2 + \lambda - \frac{\lambda}{|m_\ell|}}.$$

Since there is still a term containing $|m_\ell|$ in the right hand side of this solution, we are multiplying $\overline{m_\ell}$ with its conjugate and obtain

$$\overline{m_\ell}m_\ell = |m_\ell|^2 = \frac{|Z_\ell S_\ell|^2}{\left(|S_\ell|^2 + \lambda - \frac{\lambda}{|m_\ell|}\right)^2}.$$

Solving this equation with respect to the modulus of $m_\ell$ we get

$$|m_\ell| = \frac{|Z_\ell S_\ell| + \lambda}{|S_\ell|^2 + \lambda}.$$

Finally we are using the fact that a complex number can be written the following way

$$z = |z|\, e^{i\arg(z)}.$$

Then the solution of our functional for every $\ell$ is

$$\tilde{m_\ell} = \frac{|Z_\ell S_\ell| + \lambda}{|S_\ell|^2 + \lambda} \cdot e^{i\arg(Z_\ell\overline{S_\ell})} \qquad \text{as long as} \quad |Z_\ell S_\ell| > \lambda.$$

Here the restriction is due to (4.7).

c) The regularization term $r(m) = \|m-1\|_1$ yields the minimization

$$\tilde{m} = \min_m \|Z - m \cdot S\|_2^2 + \lambda \cdot \|m-1\|_1.$$

But now we are going to do a substitution and write instead of $m-1=\mu$ which deduces the reformulation

$$\Phi(\mu) = \|Z - (\mu+1)\cdot S\|_2^2 + \lambda \cdot \|\mu\|_1.$$

Then, we can again write the norms as sums of absolute values and get

$$\Phi(\mu) = \sum_{\ell\in\Lambda} \Phi^\ell(\mu_\ell) = \sum_{\ell\in\Lambda} \Big( |Z_\ell - (\mu_\ell+1)S_\ell|^2 + \lambda\,|\mu_\ell| \Big). \tag{4.8}$$

As in the calculations before, we now skip the sum and keep one $\ell$ fixed:

$$\Phi^\ell(\mu_\ell) = |Z_\ell|^2 - 2\mathfrak{Re}((\mu_\ell+1)\overline{S_\ell}Z_\ell) + |S_\ell|^2\,|(\mu_\ell+1)|^2 + \lambda\,|\mu_\ell|\,.$$

As before, our function $\Phi(\mu)$ assumes its solution in a real vector space and so we can split $\mu$ in $\mu = \mu^r + i\,\mu^i$ and obtain

$$\begin{aligned}\Phi^\ell(\mu_\ell^r,\mu_\ell^i) = {}& |Z_\ell|^2 - 2\mathfrak{Re}(\overline{S_\ell}Z_\ell) - 2\mathfrak{Re}(\overline{S_\ell}Z_\ell)\mu_\ell^r + 2\mathfrak{Im}(\overline{S_\ell}Z_\ell)\mu_\ell^i \\ & + |S_\ell|^2\left((\mu_\ell^r+1)^2 + (\mu_\ell^i)^2\right) + \lambda\sqrt{(\mu_\ell^r)^2+(\mu_\ell^i)^2}\end{aligned}$$

from which we compute the derivatives. The derivative with respect to the first variable is

$$\frac{\partial\Phi^\ell(\mu^r,\mu^i)}{\partial\mu^r} = -2\mathfrak{Re}(\overline{S_\ell}Z_\ell) + 2\,|S_\ell|^2\,(\mu_\ell^r+1) + \lambda\frac{\mu_\ell^r}{|\mu_\ell|} \tag{4.9}$$

and with respect to the second variable, we get

$$\frac{\partial\Phi^\ell(\mu^r,\mu^i)}{\partial\mu^i} = 2\mathfrak{Im}(\overline{S_\ell}Z_\ell) + 2\,|S_\ell|^2\,\mu_\ell^i + \lambda\frac{\mu_\ell^i}{|\mu_\ell|}. \tag{4.10}$$

Combining (4.9) and (4.10) in the formula of holomorphic functions (4.5) and setting zero yields

$$\frac{\partial\Phi^\ell(\mu_\ell)}{\partial\mu_\ell} = -\overline{S_\ell}Z_\ell + |S_\ell|^2\,\overline{\mu_\ell} + |S_\ell|^2 + \frac{\lambda}{2}\frac{\overline{\mu_\ell}}{|\mu_\ell|} = 0.$$

We are solving this equation for $\overline{\mu_\ell}$ and have

$$\overline{\mu_\ell} = \frac{\overline{S_\ell}Z_\ell - |S_\ell|^2}{|S_\ell|^2 + \frac{\lambda}{2|\mu_\ell|}}.$$

But this equation is still including the modulus of $\mu_\ell$, therefore we are going to multiply it with its conjugate

$$\overline{\mu_\ell}\mu_\ell = |\mu_\ell|^2 = \frac{(\overline{S_\ell}Z_\ell - |S_\ell|^2)(S_\ell\overline{Z_\ell} - |S_\ell|^2)}{\left(|S_\ell|^2 + \frac{\lambda}{2|\mu_\ell|}\right)^2}.$$

Solving this with respect to $|\mu_\ell|$ we obtain

$$|\mu_\ell| = \frac{|S_\ell|\,|Z_\ell - S_\ell| - \frac{\lambda}{2}}{|S_\ell|^2}. \tag{4.11}$$

As before we can rewrite $\mu = |\mu|\, e^{i\arg(\mu)}$ and using equation (4.11), we get

$$\mu_\ell = \frac{|S_\ell|\,|Z_\ell - S_\ell| - \frac{\lambda}{2}}{|S_\ell|^2} \cdot e^{i\arg(\overline{S_\ell}(Z_\ell - S_\ell))}.$$

The next step is to undo the substitution ($m = \mu + 1$) and so the solution looks like

$$\tilde{m_\ell} = \frac{|S_\ell|\,|Z_\ell - S_\ell| - \frac{\lambda}{2}}{|S_\ell|^2} \cdot e^{i\arg(\overline{S_\ell}(Z_\ell - S_\ell))} + 1. \tag{4.12}$$

One thing that we ignored so far is the division by $|\mu_\ell|$ in equation (4.9) and (4.10). So we have to apply a threshold argument which means that $|\mu_\ell| > 0$ or, without substitution $|m_\ell - 1| > 0$. If we insert $\tilde{m_\ell}$, i.e. what we have computed so far, we may use the solution $\tilde{m_\ell}$ (from last equation (4.12)) as long as

$$|S_\ell|\,|T_\ell - S_\ell| > \frac{\lambda}{2}.$$

d) For $r(m) = \|\,|m| - 1\|_1$ we have the function

$$\Phi(m) = \|Z - m \cdot S\|_2^2 + \lambda \|\,|m| - 1\|_1$$

which should be minimized. This function can also be written as the sum over all elements of $\Lambda$ instead of the norm and if we fix one $\ell$ (because of the same argument as in the proofs of the other regularization terms before) we obtain

$$\Phi^\ell(m_\ell) = |Z_\ell - m_\ell S_\ell|^2 + \lambda\,||m_\ell| - 1|\,.$$

In the following we have to make a distinction considering the term $||m_\ell| - 1|$. We can omit the modulus outside making the distinction between

- $|m_\ell| > 1$ yields
$$||m_\ell| - 1| = |m_\ell| - 1 \tag{4.13}$$

and

- $|m_\ell| < 1$ yields
$$||m_\ell| - 1| = -\,|m_\ell| + 1. \tag{4.14}$$

During the following computation the upper sign refers to the first bullet and the lower sign refers to the second point. So we have the following functional

$$\begin{aligned}\Phi^\ell(m_\ell) &= |Z_\ell - m_\ell S_\ell|^2 \pm \lambda(|m_\ell| - 1) \\ &= |Z_\ell|^2 - 2\mathfrak{Re}(\overline{Z_\ell} S_\ell m_\ell) + |S_\ell|^2\,|m_\ell|^2 \pm \lambda\,|m_\ell| \mp \lambda.\end{aligned}$$

Splitting the Gabor mask into real and imaginary part $m = m^r + i\,m^i$ we get

$$\begin{aligned}\Phi^\ell(m_\ell) = |Z_\ell|^2 &- 2\mathfrak{Re}(\overline{Z_\ell} S_\ell) m_\ell^r + 2\mathfrak{Im}(\overline{Z_\ell} S_\ell) m_\ell^i \\ &+ |S_\ell|^2\,|m_\ell|^2 \pm \lambda\,|m_\ell| \mp \lambda,\end{aligned}$$

where again $|m_\ell|^2 = ((m_\ell^r)^2 + (m_\ell^i)^2)$ and $|m_\ell| = \sqrt{(m_\ell^r)^2 + (m_\ell^i)^2}$. After computing the derivative with respect to $m_\ell^r$ and $m_\ell^i$ we obtain the following two equations

$$\frac{\partial \Phi^\ell(m_\ell^r, m_\ell^i)}{\partial m_\ell^r} = -2\mathfrak{Re}(\overline{Z_\ell} S_\ell) + 2\,|S_\ell|^2\, m_\ell^r \pm \lambda \frac{m_\ell^r}{|m_\ell|} \tag{4.15}$$

and

$$\frac{\partial \Phi^\ell(m_\ell^r, m_\ell^i)}{\partial m_\ell^i} = 2\mathfrak{Im}(\overline{Z_\ell} S_\ell) + 2\,|S_\ell|^2\, m_\ell^i \pm \lambda \frac{m_\ell^i}{|m_\ell|}. \tag{4.16}$$

We have to notice here that we are going to divide through $|m_\ell|$. Because of the restrictions from (4.13) and (4.14) it follows that $|m_\ell| \neq 0$. Inserting the equations (4.15) and (4.16) into (4.5) we obtain

$$\frac{\partial \Phi^\ell(m_\ell)}{\partial m} = -\overline{Z_\ell} S_\ell + |S_\ell|^2\, \overline{m_\ell} \pm \frac{\lambda}{2} \frac{\overline{m_\ell}}{|m_\ell|}.$$

After setting the last term equal to zero and solving it with respect to $\overline{m_\ell}$ we get

$$\overline{m_\ell} = \frac{\overline{Z_\ell} S_\ell}{|S_\ell|^2 \pm \frac{\lambda}{2|m_\ell|}}.$$

Again we have to use the argument $|m_\ell|^2 = \overline{m_\ell} \cdot m_\ell$ and this yields

$$|m_\ell| = \frac{|Z_\ell S_\ell| \mp \frac{\lambda}{2}}{|S_\ell|^2}.$$

This time we can plug the modulus of $m_\ell$ into our two constraints (4.13) and (4.14) and obtain a threshold condition. We also use the fact $z = |z|\, e^{i \arg(z)}$ and get for all $\ell$

$$\tilde{m}_\ell = \begin{cases} \frac{|Z_\ell S_\ell| - \frac{\lambda}{2}}{|S_\ell|^2} e^{i \arg(\overline{S_\ell} Z_\ell)} & \text{if } \frac{|Z_\ell S_\ell|}{|S_\ell|^2} > 1 + \frac{\lambda}{2|S_\ell|^2} \\ \frac{|Z_\ell S_\ell| + \frac{\lambda}{2}}{|S_\ell|^2} e^{i \arg(\overline{S_\ell} Z_\ell)} & \text{if } \frac{|Z_\ell S_\ell|}{|S_\ell|^2} < 1 - \frac{\lambda}{2|S_\ell|^2} \\ 1 & \text{else.} \end{cases}$$

□

After this theoretical chapter, we are looking at a MATLAB implementation which confirms Theorem 4.1 and its Remark 4.2.

# 5 Example: Diagonal Approximation between two Tones of Musical Instruments

In this chapter we want to give an example showing some properties of the regularization terms discussed in the previous Section 4. We use some sounds of the Vienna Symphonic Library [1] which are sampled with 44100 Hertz. The sounds are of different length, and therefore we use theMATLAB program `samesize_power2.m` which cuts or fades out the signals with a decrease of $e^{-x}$. For a better understanding view Figure 5.1, where we generated a sinus signal on which we applied the exponential decrease. The length of the signals is chosen properly as a power of two, because it makes things easier in the Gabor transformation of the following programs.

```
function [s, z]=samesize_power2(f_1,f_2,sec,cut)
%Input:
%                       f_1 : signal
%                       f_2 : signal
%                       sec : cuts off after the next power of 2 ...
    after 'sec' seconds
%                       cut : decreas of e^-x
% Output:
%                      s : signal on one channel with same lenght as
%                      z : signal on one channel
%
%going from stereo- to mono channel
m=size(f_1);
n=size(f_2);
if m(1,2)==2
f_1=(f_1(:,1)+f_1(:,2))/2;
end
if n(1,2)==2
f_2=(f_2(:,1)+f_2(:,2))/2;
end
%find the non-zero entries, to delete leading zeros
a=find(f_1(:,1));
b=find(f_2(:,1));
%delete the leading zeros
i=a(1)-1;
f_1(1:i)=[];
j=b(1)-1;
f_2(1:j)=[];
%with sample size 44100 we compute the number of entries of 'sec' ...
    seconds
samp=44100*sec;
%t computes the next power of 2 of 'samp' - this is the length of our
%output signal
t=2^nextpow2(samp);
%this block of if-loops tells us, whether our signals are long enough,
%otherwise an error will appear and we have to make our input argument
%'sec' smaller
if length(f_1)<length(f_2)
    l=length(f_1);
  if  t<l;
   else
      errordlg('Signals are too short, please try again with less ...
```

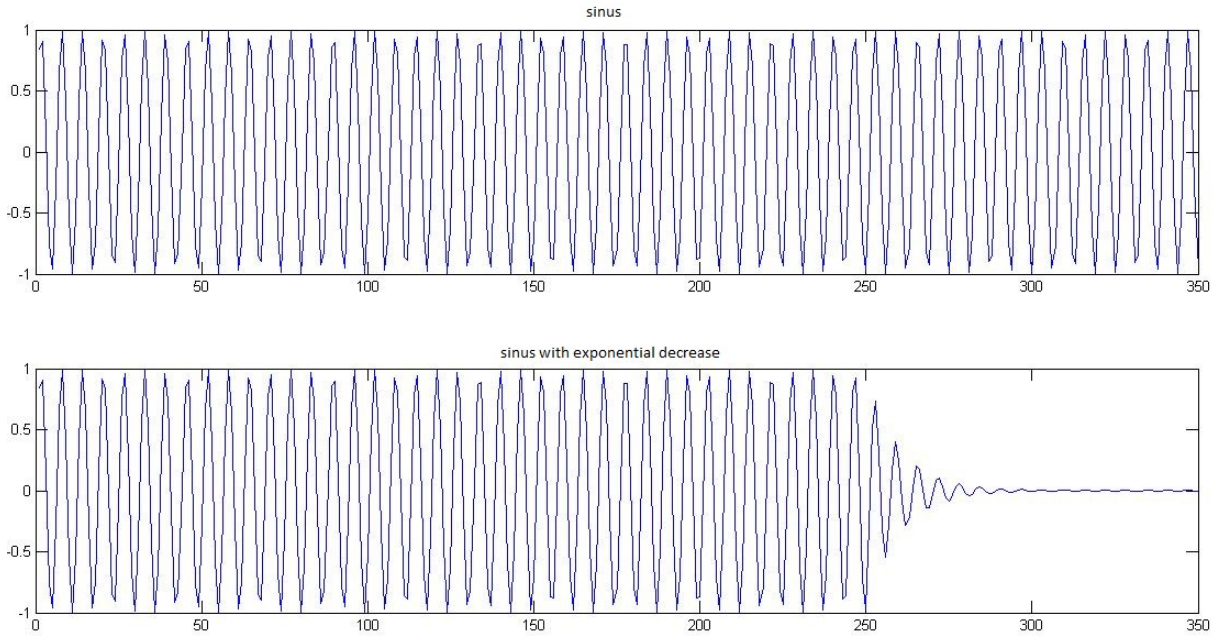


Figure 5.1: The exponential decrease applied on a generated sinus signal.

```
            seconds!')
41        a= num2str(l);
42        b=num2str(t);
43        X=[b,' is bigger than ',a];
44        disp(X);
45    end
46  else
47     l=length(f_2);
48          if  t<l;
49      else
50        errordlg('Signals are too short, please try again with less ...
              seconds!')
51        a= num2str(l);
52        b=num2str(t);
53        X=[b,' is bigger than ',a];
54        disp(X);
55       end
56
57  end
58  %we want to generate a decrease with e^-x  until 'sec' seconds
59  k=[zeros(t-cut,1);(1:cut)'];
60      f_1(t+1:length(f_1))=[];
61      f_2(t+1:length(f_2))=[];
62  s=f_1(:).*exp(-k*0.1);
63  z=f_2(:).*exp(-k*0.1);
64  end
```

For demonstration we use as source signal the violin, because its sound is rich on overtones. As target signal we use the flute which has fewer overtones, especially when it is played very high and loud. Because the two signals should be similar enough,

we use as common tone pitch $C6$. In this entire and also in the following sections we are going to use a *logarithmic scale* for our figures. A logarithmic scale helps us to show things better, if the data is very spread, i.e. from huge values to very small values which would normally not be visible. Another connection we can mention, is the relation of a logarithmic scale to our ear. The way of perception of our ear is logarithmic too, means that the detection of impulses grows logarithmically with the strength of these impulses and that implies the usage of Dezibel [dB] in a logarithmic scale. The MATLAB program we are using for our images, does the following computations for an input vector $X$

$$X_{i,j} = \begin{cases} |X_{i,j}| & \text{if } |X_{i,j}| > \text{threshold} \\ \text{threshold} & \text{if } |X_{i,j}| \leq \text{threshold} \end{cases}$$

$$\tilde{X}_{i,j} = 10 \cdot \log_{10}\left(\frac{X_{i,j}}{\text{threshold}}\right).$$

Here $\tilde{X}$ is the output which will be displayed in the spectrogram. This means that we take the logarithmic scale of all values, where we set the values smaller than a given threshold equal to 0 (this is due to the fact that $\log_{10}(1) = 0$). The threshold depends on the maximum value of the input vector and therefore has influence on the intensity relating to the colorbar of the whole image. As intensity scale we use a scale from 0 to 100. This range allows us to observe a lot of details in our images and that is what is shown with the colorbar on the right hand side of the images.
In Figure 5.2 we show the spectrogram of the musical sounds that we are going to use. One thing that is noticeable, is the different length of the two sounds (and that is why we are going to use `samsize_power2.m` for the further steps). One could also see, that the violin is stroke two times. The richness of the overtone spectrum of the violin is also recognizable in contrast to the less overtones in the spectrogram of the flute. For computing the diagonal approximation of the target signal (in our case we want to do a transformation from the violin to the flute), we use a Hann-window and the parameters $a = 256$, $M = 1024$ for the Gabor transformation with MATLAB.
Before we are going to explain the properties of the different regularization terms mentioned in Remark 4.2 of the previous section, we are going to state the following MATLAB code `diagapprox.m`. It computes the Gabor mask and the approximated transformation of the target signal from which we obtain the data for our figures.

```
1  function [Diag, m]=diagapprox(f_1, f_2,sec,cut, normw,lambda)
2  %'diagapprox' is a program which computes the Gabor mask 'm' with ...
       respect
3  %to different norms and approximates the tranformation to the ...
       target signal
4  %Input:
5  %       f_1 : source signal
6  %       f_2 : target signal
7  %       sec : cuts off after the next power of 2 after 'sec' seconds
8  %       cut : decrease of e^-x
9  %       normw  : regularization norm which is used: 1, abs1, 2, abs2
10 %       lambda  : regularization parameter (should be between 0 ...
       and 1)
11 %Output:
```

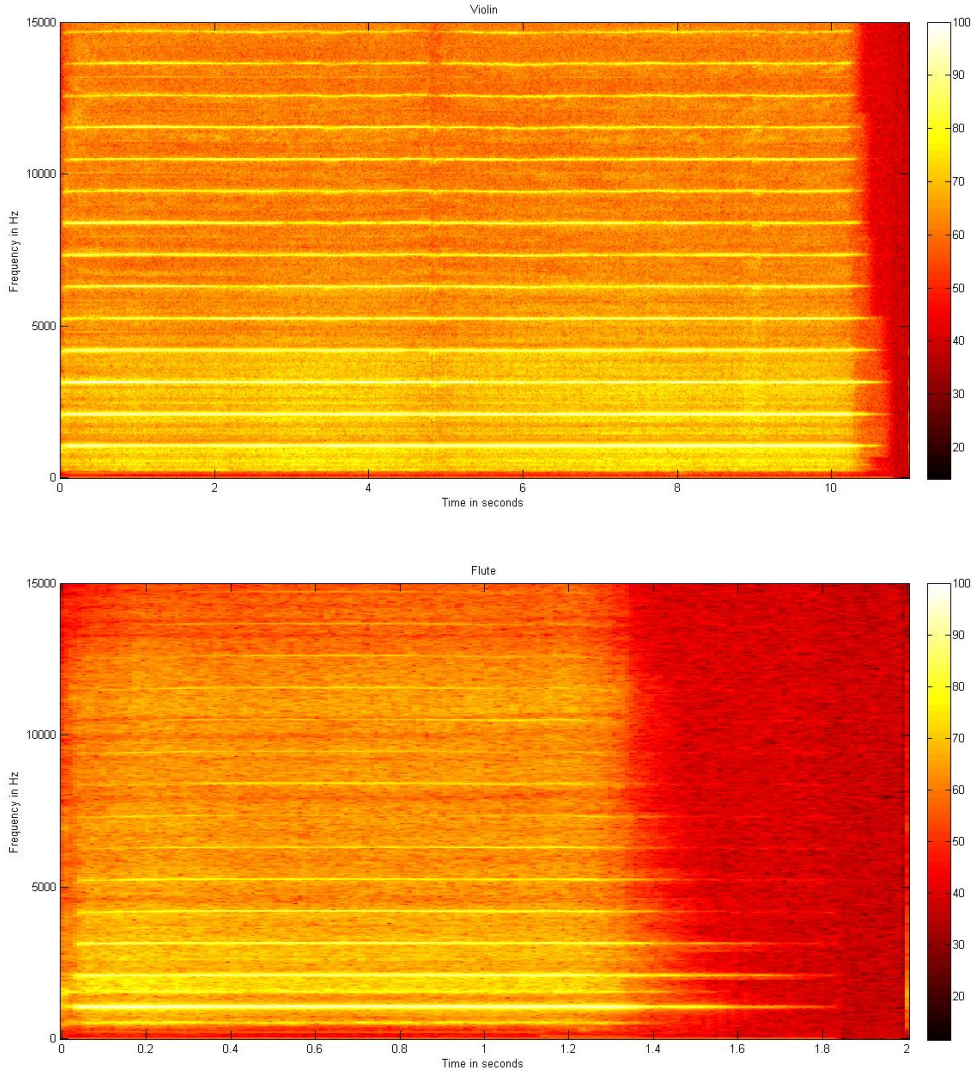


Figure 5.2: Original sounds with original length [1]. The first signal corresponds to the violin and the second to the flute.

```
%       Diag : approximates the transformation to the target ...
    signal through computed mask 'm'

%parameters used for Gabor transformation
a=256;
M=1024;
g=firwin('hann',M);
h=gabdual(g,a,M);
% to have the same size of input and output, we use the following ...
    program:
[s, z]=samesize_power2(f_1,f_2,sec,cut);
%want the same length of input and output
if length(s) ~= length(z)
    error('s and z must have same length')
end

%generate column vectors
if size(s,1)<size(s,2)
    s=transpose(s);
end
if size(z,1)<size(z,2)
    z=transpose(z);
end

Ls=length(s);
%Gabor transformation
S=dgt(s,g,a,M,Ls);
Z=dgt(z,g,a,M,Ls);

%diagonal masks for different norms
%1-norm of (m-1)
if normw == '1'
temp = thresh(abs(Z-S).*abs(S),lambda/2,'soft');
temp = temp./(abs(S).^2 );
temp = temp.*exp(1i*angle(conj(S).*(Z-S)));
m = temp+1;

%1-norm of (|m|-1)
elseif normw == 'abs1'
temp = thresh(abs(Z.*S)-abs(S).^2,lambda/2,'soft');
temp=temp./(abs(S).^2 )+1;
m= temp.*exp(1i*angle(Z.*conj(S)));

%2-norm of (m-1)
elseif normw == '2'
temp = conj(S).*Z  + lambda ;
m = temp ./ (abs(S).^2 + lambda);

%2-norm of (|m|-1)
elseif normw == 'abs2'
temp =  thresh(abs(Z.*S),-lambda, 'soft');
temp = temp./(abs(S).^2 +lambda);
m = temp.*exp(1i*angle(Z.*conj(S)));
end
% reconstruct z through given sigmatild - output in diagonal case
   Diag =idgt(m.*S,h,a,Ls);
end
```

For our figures and demonstrations we use $\lambda = 0.1$. This parameter, as mentioned in the previous sections, controls the regularization term and allows the interpolation between source and target signal, i.e. the transformation of the signal is neither the violin nor the flute and therefore something in between. With this choice of parameter $\lambda$ we obtain a reconstructed signal which sounds almost as the flute shown in Figure 5.2. In Figure 5.3 we can see the diagonal approximation of the transformation to the target signal, computed with different regularization terms.

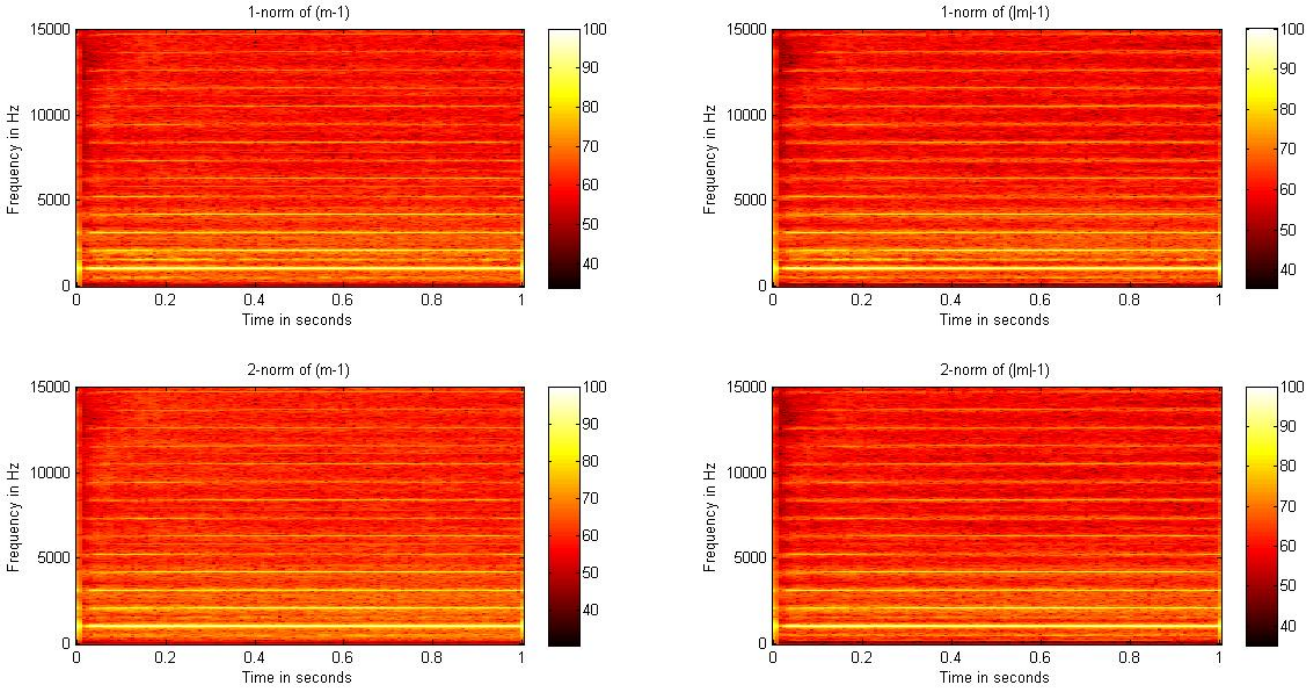


Figure 5.3: Diagonal approximation of the transformation to the flute with different regularization terms.

Since all approximated signals corresponding to different regularization terms sound almost the same (close to the sound of the flute), they have also nearly the same spectrogram, view Figure 5.3. One thing that could be recognized are the overtones which are better visible in the lower figure on the right side of the regularization term $\| |m| - 1\|_2^2$.
In the following we want to discuss a few comments of Remark 4.2 of the previous section. Figure 5.4 shows the Gabor mask of the 1-norm and compares the regularization terms $\|m - 1\|_1$ and $\| |m| - 1\|_1$. Spurious oscillations are visible in the upper figure between 13kHz and12kHz. These oscillations are extinct in the lower figure. This shows that the oscillation can be avoided by taking the modulus of the Gabor mask $m$. Figure 5.5 compares the Gabor masks of the regularization terms $\|m - 1\|_2^2$ and $\| |m| - 1\|_2^2$. As mentioned in Remark 4.2 and similar to the example of the 1-norm above, we also recognize some oscillation in the regularization term $\|m - 1\|_2^2$ between 12kHz and 13kHz. There we can see some oscillating color changes, respectively some oscillating distinctions. In the lower figure we see the mask of the regularization term $\| |m| - 1\|_2^2$, which avoids the oscillations by taking the modulus of $m$.

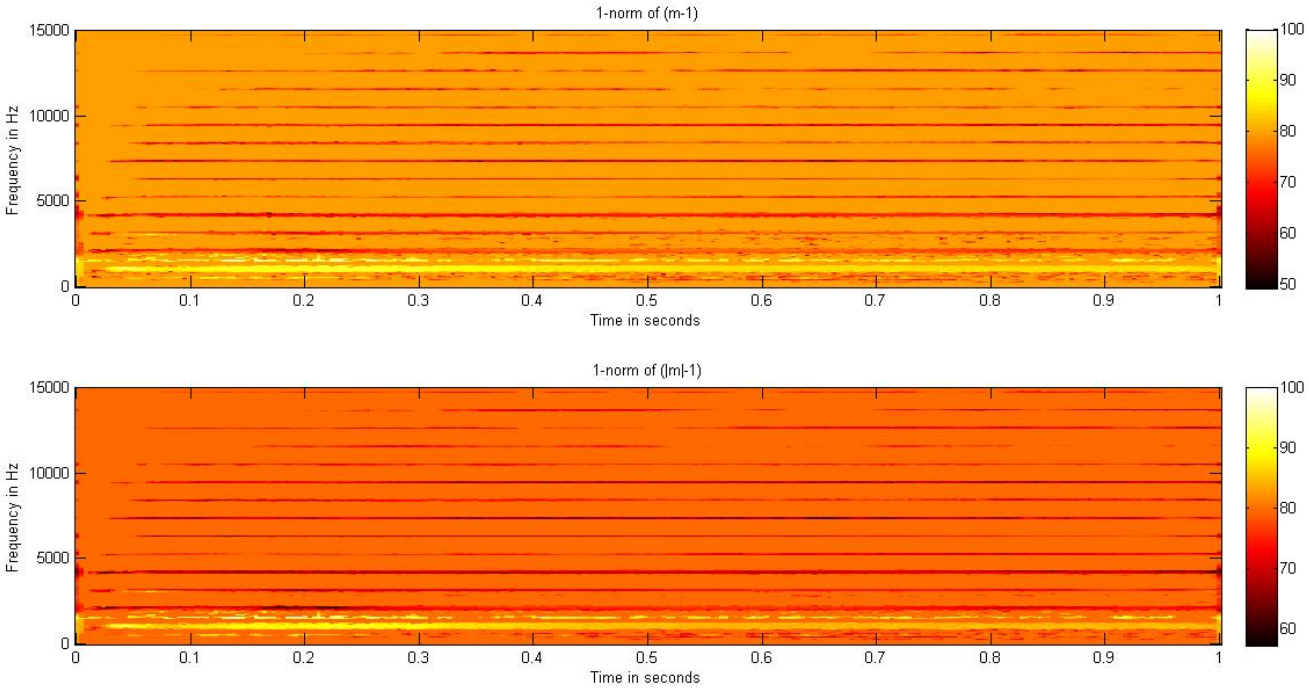


Figure 5.4: Gabor mask of the diagonal approximation comparing the regularization terms of the 1-norm.

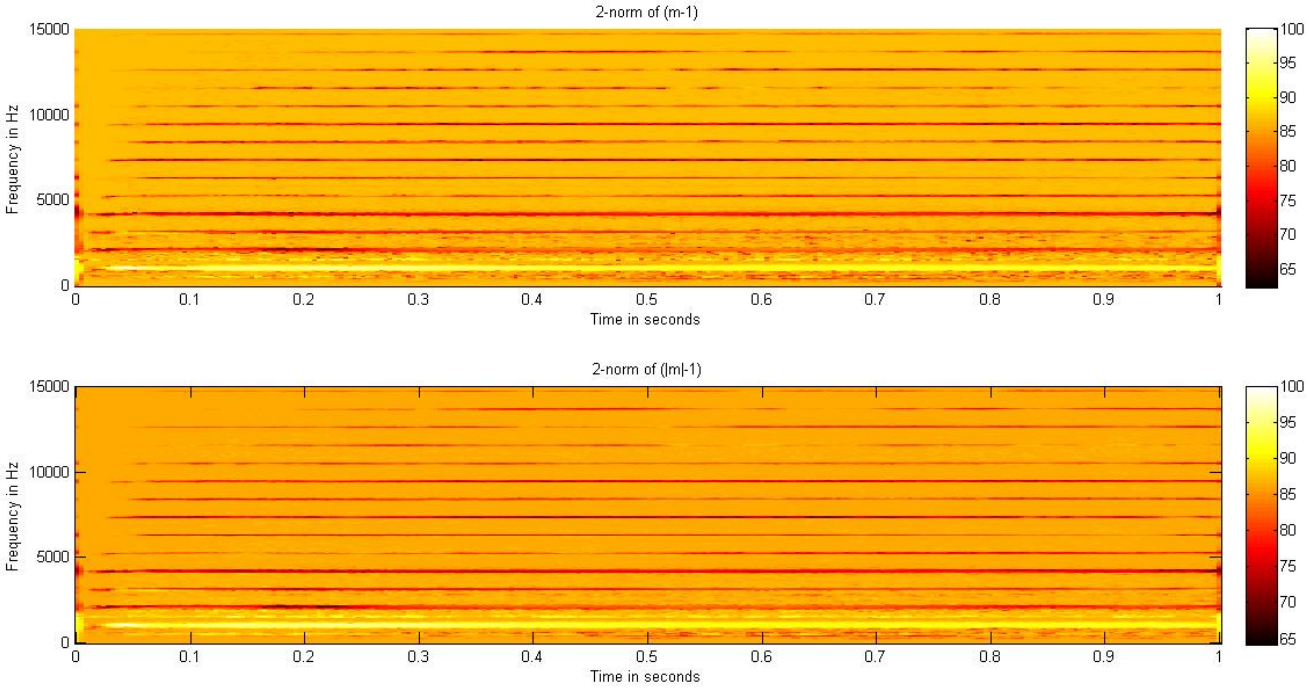


Figure 5.5: Gabor mask of the diagonal approximation comparing the regularization terms of the 2-norm.

For some further experiments and information consider [18, p. 45–48] and [19, p. 7 et seq.]. In the following section we are going to introduce shrinkage and threshold algorithms to get a solution of our signal transformation in the general, i.e. the non-diagonal case. After preparing the theory of the algorithms, we are going to establish a chapter introducing a MATLAB code corresponding the developed theory and doing some experiments considering the sound of the flute and the violin again.

# 6 Iterative Algorithms

In the preceding two chapters we focused on solving the diagonal approximation, i.e. find the minimum of $\Phi(m) = \|(V_g z - m V_g s)\|_2^2 + \lambda r(m)$. Now we want to find a solution of

$$\tilde{m} = \arg\min_m \|z - \mathcal{O}m\|_2^2 + \lambda\, r(m). \tag{6.1}$$

For this problem an explicit solution does not exist. Therefore we have to develop an iterative method which is satisfactory for solving inverse problems. In the following we prepare a model for general regularization terms and we give a MATLAB code for concrete solutions for different terms of $r(m)$ in Chapter 7. Nevertheless the regularization terms that we discussed in Chapter 4, should be kept in mind due to some distinctions. Further details to this chapter can be found in [9], [4], [3], [8], [17] and [18].
Writing (6.1) from above in a more general way we have to find the minimum $\tilde{m}$ by minimizing

$$\Phi(m) = f(m) + \lambda r(m), \qquad f(m) = \|z - \mathcal{O}m\|_2^2. \tag{6.2}$$

A general simplification would be, solving the following problem (considering $r(m) = 0$)

$$\tilde{m} = \arg\min_m f(m).$$

This could be done iteratively using a gradient method which computes a converging sequence $(m^n)_{n\geq 0}$ as

$$m^n = m^{n-1} - t^n \nabla f(m^{n-1}). \tag{6.3}$$

This method searches in the negative gradient direction of the current point which is the direction of the "steepest descent". The parameter $t^n$ denotes the step size that the algorithm goes in direction $-\nabla f(m^{n-1})$ in the $n$-th step. Unfortunately, this method cannot be applied in the more general case of equation (6.2), i.e. $r(m) \neq 0$. Following the paper of Amir Beck and Marc Teboulle [4] we develop now an approximation model. The strategy will be prepared in the following steps. We start by formulating a Taylor expansion of the function $f(m)$ of equation (6.2). Therefore we take the function $f$ at a given point $y$ and regularize by a "local error" measuring quadratic term [4, p. 7]. The remainder for the second derivative of the expansion is given by

$$R_2(f) = \frac{\mathrm{Hess}_f(\xi) - \mathrm{Hess}_f(y)}{2!}\|m - y\|^2 \qquad \xi \in (m, y).$$

Here $\mathrm{Hess}_f(x)$ denotes the Hesse-matrix which corresponds to $f''(x)$ in the one-dimensional case. The step size $t$ should now satisfy the following

$$\mathrm{Hess}_f(\xi) - \mathrm{Hess}_f(y) \leq \frac{1}{t}.$$

Later on we express the step size through the Lipschitz constant, cf. equation (6.9). The function $r(m)$ is added untouched and the approximation model looks as follows

$$q_t(m, y) = f(y) + \langle m - y, \nabla f(m)\rangle + \frac{1}{2t}\|m - y\|^2 + \lambda r(m). \tag{6.4}$$

A straightforward calculation shows

$$q_t(m, y) = \frac{1}{2t}\|m - (y - t\nabla f(y))\|^2 - \frac{t}{2}\|\nabla f(y)\|^2 + f(y) + \lambda r(m). \tag{6.5}$$

In order to compute the minimum of equation (6.2), we have to minimize formula (6.5). Normally, if we want to compute a minimum, we would set the first derivative equal to zero. Due to the derivative the constant terms vanish. Hence we can omit the constant terms of (6.5) and get a method which is valid for cases when differentiation is not possible by setting

$$m^n = \underset{m}{\operatorname{argmin}}\Big\{\lambda r(m) + \frac{1}{2t^n}\|m - (m^{n-1} - t^n \nabla f(m^{n-1}))\|^2\Big\}. \tag{6.6}$$

This is somehow similar to the fundamental proximal operator, a general class of algorithms which also minimizes non-differentiable functions. Solution for special cases of $r(m)$ are stated in Chapter 7, but before we have to introduce proximal algorithms, a machinery for more general problems, i.e. non-convex problems.

## 6.1 Proximal Algorithms

We follow the considerations of paper [4], [3] and [8] by introducing the fundamental proximal operator as an approximation of the gradient based model shown in the previous subsection. The proximal operator

$$\operatorname{prox}_t(r) : \mathbb{C}^L \to \mathbb{C}^L$$

for any step size $t > 0$ is defined as

$$\operatorname{prox}_t(r)(z) = \underset{m}{\operatorname{argmin}}\Big\{r(m) + \frac{1}{2t}\|m - z\|_2^2\Big\}. \tag{6.7}$$

It compensates with a proximal step the calculation of the gradient of $r(m)$. Our iteration (6.6) can be expressed in terms of the proximal gradient operator as

$$m^n = \operatorname{prox}_{t^n}(\lambda r)(m^{n-1} - t^n \nabla f(m^{n-1})). \tag{6.8}$$

Following the approach of Anaïk Olivero in her thesis [18], we define the step size $t$ to be constant for all $n \geq 1$,

$$t := \frac{1}{L(f)}. \tag{6.9}$$

Here $L(f) > 0$ is the Lipschitz constant of $\nabla f$, i.e.

$$\|\nabla f(x) - \nabla f(y)\| \leq L(f)\|x - y\|,$$

for every $x, y \in \mathcal{G}$. One has to be careful since $L(f)$ is not always easy to compute. In the following chapters we are going to do some experiments and also show one possible calculation of $L(f)$ in the discrete case, see (7.2).
Now we consider $f(m)$ and $r(m)$ to be convex, because then we obtain a convex problem.

**Definition 6.1. (Convex problem )**
*A convex optimization problem has the form*

$$\min_{m \in \mathcal{D}} \Phi(m),$$

*where $\Phi : \mathbb{C}^L \to \mathcal{D}$ is a convex function and $\mathcal{D} \subseteq \mathbb{C}^L$ is a convex set.*

*Remark* 6.2 ([6, p. 138]). A fundamental property of a convex optimization problem is, that any locally optimal point is also globally optimal.

We therefore discuss first the convex case. Later on we make some comments to the non-convex case. The parameter $L(f)$ is assumed to be known, such that we can use a constant step size. For the case where $L(f)$ is unknown or cannot be computed, a backtracking step size rule exists (cf. [4, p. 18]). For the sake of completeness we introduce the following structure of the backtracking step size rule (BSR).

**Data** : $L_0 > 0$ and $\eta > 1$

$$\left.\begin{array}{l} \text{search smallest} \quad i_n > 0 \ \text{ for } \ L = \eta^{i_n} L_{n-1} \\ \text{s.t.} \quad \Phi(m^n) \leq q_{\frac{1}{L}}(m^n, m^{n-1}) \\ \text{set} \quad L_n = \eta^{i_n} L_{n-1} \end{array}\right\} \qquad \text{(BSR)}$$

**Algorithmus 1 :** Backtracking step size rule.

Here $m^n$ and $m^{n-1}$ are computed through the proximal operator as in Algorithm 2 equation (6.10). Then the proximal operator could compute the next iteration value $m^n$ with the new found parameter $L_n$, by using the algorithm below, equation (6.10). The method of the backtracking step size is optional and could be inserted in the Algorithm 2 at the place, marked with (BSR). Then the algorithm for proximal gradient methods without backtracking step size rule generally looks as follows:

**Data** : $L = L(f) > 0$ and $m^0$
**while** *the relative error is bigger than some threshold and* $n \geq 1$ **do**

If required, perform (BSR).

$$m^n = \text{prox}_{\frac{1}{L}}(\lambda r)(m^{n-1} - \frac{1}{L}\nabla f(m^{n-1})) \qquad (6.10)$$

**end**

**Algorithmus 2 :** Proximal Gradient Method (PGM) with constant step size.

The advantage of this algorithm lies in its simplicity. On the other hand it converges very slowly. We are going to see the order of its convergence and some associated properties in the following theorem.

**Theorem 6.3.** *Let* $f : \mathbb{C}^L \to \mathbb{R}^+$ *be a convex, Lipschitz-differentiable function with* $L(f) > 0$ *and* $r : \mathbb{C}^L \to \mathbb{R}^+$ *be a convex and continuous function. Assume that the minimization problem*

$$\min_{m \in \mathbb{C}^L} \Phi(m) = f(m) + \lambda r(m),$$

*where* $\lambda > 0$ *admits a solution* $\tilde{m} \in \mathbb{C}^L$*. Then for* $n \geq 1$ *the sequence of the following iteration*

$$m^n = \text{prox}_{\frac{1}{L(f)}}(\lambda r)(m^{n-1} - \frac{1}{L(f)}\nabla f(m^{n-1})), \qquad m^0 \in \mathbb{C}^L$$

*satisfies:*

*(i) For every* $n \geq 1$ *the sequence* $(\Phi(m^n))_n$ *is non-increasing, i.e.*

$$\Phi(m^n) \leq \Phi(m^{n-1}).$$

*(ii) For every* $n \geq 1$

$$\Phi(m^n) - \Phi(\tilde{m}) \leq \frac{L(f)\|m^0 - \tilde{m}\|^2}{2n}$$

*and therefore we have a method of order* $\mathcal{O}(\frac{1}{n})$.

*(iii) For every solution* $\tilde{m}$ *and* $n \geq 1$ *the Fejer monotonicity holds*

$$\|m^n - \tilde{m}\| \leq \|m^{n-1} - \tilde{m}\|.$$

*Thus we get convergence, i.e.* $\lim_{n\to\infty} m^n = \tilde{m}$.

*Proof.* See [4, p. 18–20]. □

*Remark* 6.4. If we have to find the minimum of a convex optimization problem, every solution $\tilde{m}$ of Theorem 6.3 is a unique minimizer.

A problem occurs for the usage of the regularization term $r(m) = \| |m| - 1\|_p^p$ for $p = 1, 2$. These regularization terms clearly avoid the oscillation which we observed in the diagonal case for the terms without the modulus of $m$. Unfortunately, $r(m) = \| |m| - 1\|_p^p$ for $p = 1, 2$ are *non-convex* regularization terms:
We can observe this regularization term as a concatenation of functions like $h(x) = |x|$ and $g(y) = \|y - 1\|_p^p$, where we obtain

$$r(m) = g(h(m)).$$

Due to [6, p. 86] this regularization function would be convex, if $g$ is convex and non-decreasing and $h$ is convex. Both functions $g$ and $h$ are convex. However, $g$ is unfortunately not non-decreasing everywhere which means that $r(m)$ does not necessarily have to be convex, see the following counterexample.

**Counterexample 6.5.** Assume that $r(m) = \||m| - 1\|_p^p$ is convex and $m \in \mathbb{C}$ (for simplicity we choose the dimension $L = 1$). Convexity of $r$ implies

$$r(tm_1 + (1-t)m_2) \leq tr(m_1) + (1-t)r(m_2) \quad \text{for all } t \in [0, 1].$$

Because this should be valid for all $m_1, m_2 \in \mathbb{C}$ and for all $t$ we can insert special values. Let $t = \frac{1}{2}$, we obtain

$$r(\frac{m_1 + m_2}{2}) \leq \frac{r(m_1) + r(m_2)}{2}.$$

Choose $m_1 = 1, m_2 = -1$ we have $1 \leq 0$ a contradiction.

For the non-convex case of $\Phi(m)$ the convergence is of weaker order $\mathcal{O}(\frac{1}{\sqrt{n}})$ (cf. [4, p. 21]). Moreover, convergence to the global minimum is not guaranteed. This means that we can find local solutions, but not necessarily global ones and therefore the uniqueness of the solution gets lost. Nevertheless, in the case of non-convex regularization terms the proximal gradient method is also defined (see [18, p. 54 et seq.]) and the following section will show that we also could use these regularization terms and obtain useful solutions.
In the following we want to keep the simplicity of Algorithm 2 which we have stated

above, but we want to improve the convergence rate by introducing the fast proximal gradient method (FPGM) found by Amir Beck and Marc Teboulle in [3, p. 193]. The essential ingredient is now that the proximal step is done by using a linear combination of the last two steps $m^{n-2}$ and $m^{n-1}$. The algorithm looks as follows:

**Data** : $L = L(f) > 0$ and $y^1 = m^0$, $t^1 = 1$
**while** *the relative error is bigger than some threshold and* $n \geq 1$ **do**

If required, perform (BSR).

$$m^n = \text{prox}_{\frac{1}{L}}(\lambda r)(y^n - \frac{1}{L}\nabla f(y^n))$$

$$t^{n+1} = \frac{1 + \sqrt{1 + 4(t^n)^2}}{2}$$

$$y^{n+1} = m^n + \left(\frac{t^n - 1}{t^{n+1}}\right)(m^n - m^{n-1})$$

**end**

**Algorithmus 3 :** Fast Proximal Gradient Method (FPGM) with constant step size.

In this algorithm a backtracking step size rule could be used if $L(f)$ is unknown. One has to insert Algorithm 1 for (BSR). Because the fast version and the normal version are both using the proximal operator, the main effort remains the same and the simplicity stays, but now we have a slightly faster algorithm.

**Theorem 6.6.** *Let $f(m)$ and $r(m)$ fulfill the assumptions of the previous Theorem 6.3. Then the sequence*

$$m^n = \text{prox}_{\frac{1}{L(f)}}(\lambda r)(y^n - \frac{1}{L(f)}\nabla f(y^n)), \qquad m^0 \in \mathbb{C}^L,$$

*where*

$$t^{n+1} = \tfrac{1+\sqrt{1+4(t^n)^2}}{2}, \qquad t^1 = 1,$$
$$y^{n+1} = m^n + \left(\tfrac{t^n-1}{t^{n+1}}\right)(m^n - m^{n-1}), \qquad y^1 = m^0$$

*satisfies the following:*

$$\Phi(m^n) - \Phi(\tilde{m}) \leq \frac{2L(f)\|m^0 - \tilde{m}\|^2}{(n+1)^2} \qquad \text{for every } n \geq 1.$$

*Proof.* See [4, p. 25]. □

Although we should be satisfied with the new found fast gradient method, there are still problems occurring. The convergence of the computed sequence $(m^n)_{n\geq 0}$ to the solution is not guaranteed and it could happen that the cost function is non-decreasing and in the worst case diverging. This yield the definition of a further improvement called monotone fast proximal gradient method (MFPGM). In the general class of minimization algorithms, monotonicity is indeed a desirable property, but not needed for convergence. The new algorithm which is about to be introduced, does not manipulate the theoretical convergence rate and inherits the same properties as the non-monotone

method. For monotonicity the now improved algorithm contains term (6.11) which takes the previously calculated value ($m^{n-1}$), if the new value ($z^n$) causes an increase. Otherwise it proceeds with $z^n$. The algorithm stated in [4, p. 21] looks as follows:

**Data** : $L = L(f) > 0$ and $y^1 = m^0$, $t^1 = 1$
**while** *the relative error is bigger than some threshold and* $n \geq 1$ **do**

If required, perform (BSR).

$$z^n = \text{prox}_{\frac{1}{L}}(\lambda r)(y^n - \frac{1}{L}\nabla f(y^n))$$

$$t^{n+1} = \frac{1 + \sqrt{1 + 4(t^n)^2}}{2}$$

$$m^n = \text{argmin}\{\Phi(m) : m = z^n, m^{n-1}\} \tag{6.11}$$

$$y^{n+1} = m^n + \left(\frac{t^n}{t^{n+1}}\right)(z^n - m^n) + \left(\frac{t^n - 1}{t^{n+1}}\right)(m^n - m^{n-1})$$

**end**

**Algorithmus 4 :** Monotone Fast Proximal Gradient Method (MFPGM) with constant step size.

(BSR) again marks, where a backtracking step size rule, alias Algorithm 1, could be inserted for unknown step size $L(f)$.
After introducing the algorithms for general regularization terms $r(m)$, we have a short section about the calculation of the proximal operator for the different regularization terms that we have introduced in Chapter 4 in Theorem 4.1. We also give a step by step introduction of the MATLAB code which we use for applications of Chapter 8.

# 7 MATLAB Code for Gabor Multiplier Transformation

In this section we are going to introduce the algorithm which we will use for our experimental Chapter 8. We successively discuss the steps of our MATLAB code `gabmult.m` which is based on a MATLAB code provided by Anaïk Olivero[2]. It is also important to mention that LTFAT (Large Time-Frequency Analysis Toolbox)[3] is necessary for this MATLAB code, for further information view [21] and [22].
We start with some initial properties and the initialization of the Gabor multiplier which is then used and continued through a proximal gradient method, cf. equations (7.4) and (7.5) and Line 124 and following in the MATLAB code. It is important to choose the initialization parameter $\lambda$ carefully. Gradient algorithms are in general fast for big values of $\lambda$, i.e. $\lambda$ close to 1. Therefore such a $\lambda$ yields "no transformation". For $\lambda$ very small, i.e. $\lambda = 10^{-6}$ (almost zero), we obtain a transformation of the input signal into the target signal. If we choose $\lambda$ between 0 and 1, we obtain a transformed signal which sounds neither as the input nor as the output signal. Moreover, we are going to use the diagonal approximation of Chapter 4 as a possible initialization, cf. MATLAB code Line 62 and following.
**Initialization:**

```
1  function  [rs,sigmatild,Crec,eval,init] = gabmult(s, z, g, h, a, ...
       M, normw,lambda, algo, tol, iter_max)
2  % This function approximates the Gabor mask between two signals by ...
       proximal
3  % gradient algorithms by minimizing the inverse problem:
4  % Phi(m)=norm(Z-m*S)^2+lambda*regularization term
5  % s and z have same length due to samesize_power2.m
6  % S is the Gabor transform of s
7  % Z is the Gabor transform of z
8
9  % Input parameters:
10 %         s     : Source signal
11 %         z     : Target signal
12 %         g     : Analysis Window
13 %         h     : Synthesis Window (the dual window of g)
14 %         a     : Length of time shift
15 %         M     : Length of frequency shift.
16 %       normw   : regularization norm/term which is used: 1, abs1, ...
       2, abs2
17 %       lambda  : regularization parameter
18 %        algo   : algorithm that should be used; pgm, fpgm or mfpgm
19 %
20 % Output parameters:
21 %       mask    : Gabor mask
22 %       sigmatild   : Gabor mask by diagonal approximation ...
       (initialization of the algorithm)
23 %       rs      : reconstructed signal
24 %       eval    : Cost function values
25 %       Crec    : Output in the diagonal case, (computed)
```

[2] Anaïk Olivero: Les Multiplicateurs Temps-Fréquence. Applications à l'Analyse et la Synthèse de Signaux Sonores et Musicaux.

[3] The toolbox can be found on http://ltfat.sourceforge.net/.

```
26 %%
27 % Optional parameters:
28 %
29 %      iter_max : Stopping criterion: maximal number of ...
       iterations. Default value is 100 (cf. franalasso)
30 %       tol     : Stopping criterion: minimum relative difference ...
       between
31 %                 norms in two consecutive iterations. Default ...
       value is
32 %                 1e-2. (cf. franalasso)
33 %
34 % we have to check, if iter_max and tol are valid:
35 if iter_max>100
36     error('iter_max has to be less than 100')
37 end
38
39 if tol<1e-2
40     error('tol has to be bigger than 1e-2')
41 end
42
43 %want the same length of input and output
44 if length(s) ~= length(z)
45     error('s and z must have same length')
46 end
47
48 %generate column vectors:
49 if size(s,1)<size(s,2)
50     s=transpose(s);
51 end
52 if size(z,1)<size(z,2)
53     z=transpose(z);
54 end
55
56 Ls=length(s);
57 %discrete Gabor transformation (dgt)
58 S=dgt(s,g,a,M,Ls);
59 Z=dgt(z,g,a,M,Ls);
60 %%
61
62 % Initialization: diagonal mask for different norms
63 %1-norm of (m-1)
64 if normw == '1'
65 m = thresh(abs(Z-S).*abs(S),lambda/2,'soft');
66 m = m./(abs(S).^2 );
67 m = m.*exp(1i*angle(conj(S).*(Z-S)));
68 m = m+1;
69 p=1;
70
71 %1-norm of (|m|-1)
72 elseif normw == 'abs1'
73 m = thresh(abs(Z.*S)-abs(S).^2,lambda/2,'soft');
74 m=m./(abs(S).^2 )+1;
75 m= m.*exp(1i*angle(Z.*conj(S)));
76 p=1;
77
78 %2-norm of (m-1)
79 elseif normw == '2'
```

```
m = conj(S).*Z  + lambda ;
m = m ./ (abs(S).^2 + lambda);
p=2;

%2-norm of (|m|-1)
elseif normw == 'abs2'
m =  thresh(abs(Z.*S),-lambda, 'soft');
m = m./(abs(S).^2 +lambda);
m = m.*exp(1i*angle(Z.*conj(S)));
p=2;

end

%does the transformation from s to z through given sigmatild - ...
    output in diagonal case
   Crec =idgt(m.*S,h,a,Ls);
   sigmatild=m;
```

Following the notation of the previous Chapter 6, where

$$\Phi(m) = f(m) + \lambda r(m), \qquad f(m) = \|z - \mathcal{O}m\|_2^2, \tag{7.1}$$

we now obtain as Lipschitz-constant $L(f)$ of the gradient of $f(m)$ :

$$L(f) = 2 \cdot \lambda_{\max}(\mathcal{O}^*\mathcal{O}), \tag{7.2}$$

cf. [3, p. 189]. For the Gabor transformed case we obtain $L(f) = 2 \cdot \lambda_{\max}(\mathcal{S}^*\mathcal{S})$, because $f(m) = \|Z - Sm\|_2^2$. The cost function is computed through $\Phi(m^n) = \|z - V_h^*(m^n S)\|_2^2 + \lambda r(m^n)$ for every iteration $n$, where $S$ is the Gabor transformation of the input signal $s$ and $z$ denotes the target signal.

```
%Lipschitz-Constant
Cest=2*max(abs(eig(S'*S)));
%%
%Initialization of the mask
init = m;
clear m;
%%
% in eval we insert the values that we get for our functional, ...
    i.e. cost function
% first evaluation input
eval = zeros(iter_max,1);
if strcmp(normw,'abs1')==1 || strcmp(normw,'abs2')==1
eval(1) = norm( idgt(S.*init,h,a,Ls) - z )^2 + lambda* norm( ...
    abs(init(:))-1, p )^p ;
else
eval(1) = norm( idgt(S.*init,h,a,Ls) - z )^2 + lambda* norm( ...
    init(:)-1, p )^p ;
end
```

**Proximal Gradient Algorithms:**
Now we introduce the proximal gradient algorithms which we initiate with a while-loop. This loop is performed as long as the stopping criterion of the algorithm is

not met. In our algorithm we have two criteria (Line 122), the first is a maximal number of iterations. It tells us that if we want for example, in the case of the normal proximal gradient method, $\varepsilon$-optimal solution (i.e. $\Phi(m^n) - \Phi(\tilde{m}) \leq \varepsilon$ for $\tilde{m}$ $\varepsilon$-optimal solution), we need at most $\lceil \frac{L(f)\|m^0 - \tilde{m}\|^2}{2\varepsilon} \rceil$ iterations [4, p. 20]. This can be seen as follows:
As stated in Theorem 6.3

$$\Phi(m^n) - \Phi(\tilde{m}) \leq \frac{L(f)\|m^0 - \tilde{m}\|^2}{2n}$$

and we want the right hand side to be smaller than $\varepsilon$. If we do a reformulation of

$$\frac{L(f)\|m^0 - \tilde{m}\|^2}{2n} \leq \varepsilon$$

for $n \geq 1$, we get

$$\frac{L(f)\|m^0 - \tilde{m}\|^2}{2\varepsilon} \leq n.$$

This shows, that for a $\varepsilon$-optimal solution of the PGM (Proximal Gradient Method), we need at most $\lceil \frac{L(f)\|m^0 - \tilde{m}\|^2}{2\varepsilon} \rceil$ iterations. For the sake of FPGM (Fast PGM) and MFPGM (Monotone FPGM) we can use the estimation for $\Phi(m^n) - \Phi(\tilde{m})$ of Theorem 6.6 and proceed analogously as for PGM. Thus we obtain a $\varepsilon$-optimal solution after at most $\lceil \frac{\sqrt{2L(f)\|m^0 - \tilde{m}\|^2}}{\sqrt{\varepsilon}} - 1 \rceil$ iterations. But in the MATLAB code we are going to use the reference value of `franalasso.m` which is 100 (cf. Line 29 in the MATLAB code).
The second stopping criterion is the relative error, cf.[18, p. 54], between two subsequent values of the cost function

$$\text{error} = \frac{\|\Phi(m^n) - \Phi(m^{n-1})\|}{\|\Phi(m^n)\|}. \tag{7.3}$$

Again we take the reference value of the MATLAB program `franalasso.m` which is $10^{-2}$, as stated in Line 32 of the MATLAB code.
For the fast algorithm and for the monotone some further initializations should be done, view Line 117 and 118 of the code. The general gradient step is given by $m^n - \frac{1}{L(f)}\nabla f(m^n)$. Due to (7.1) the gradient is $\nabla f(m) = \mathcal{O}^*(\mathcal{O}m - z)$ and we obtain the gradient step

$$m^n - \frac{1}{L(f)}\mathcal{O}^*(\mathcal{O}m^n - z).$$

We consider here the Gabor transformed case $f(m) = \|Sm - Z\|_2^2$, and use for the subsequent statements of the proximal operator the gradient step

$$k^n = m^n - \frac{1}{L(f)}\mathcal{S}^*(\mathcal{S}m^n - Z).$$

The explicit computation of some proximal operators exist, if we consider regularization terms of the $\ell^p$-norms (which is our case). For supplementary information see also [18] and [19]. Consider $k^n \in \mathbb{C}^L$, then the proximal operator for the 1-norm is

$$\text{prox}_{\frac{1}{L(f)}}(\lambda\|.\|_1)(k^n) = e^{i\arg(k^n)}[|k^n| - \lambda]_+. \tag{7.4}$$

The proximal operator of the 1-norm has a special name, called soft threshold operator. In that case the algorithms are called different, i.e. the proximal gradient method is called iterative shrinkage threshold algorithm (ista), first mentioned in [9]. The fast version is called (fista) and the monotone one is (mfista), see [3]. If we want to take the proximal operator for the 2-norm, we have to use

$$\text{prox}_{\frac{1}{L(f)}}(\lambda\|.\|_2^2)(k^n) = \frac{k^n}{1+\lambda/L(f)}. \tag{7.5}$$

One basic property of these proximal operators is the translation:

$$\text{prox}_{\frac{1}{L(f)}}(\|T_x.\|)(k^n) = x + \text{prox}_{\frac{1}{L(f)}}(\|.\|)(k^n - x)$$

which we have to use for our regularization terms. Since we use $r(m) = \|m-1\|_p^p$ for $p = 1, 2$, we consider

$$\text{prox}_{\frac{1}{L(f)}}(\lambda r)(k^n) = 1 + \text{prox}_{\frac{1}{L(f)}}(\lambda\|.\|_p^p)(k^n - 1).$$

For the regularization terms including the modulus of $m$, i.e. $r(m) = \|\,|m|-1\|_p^p$ for $p = 1, 2$, we can take the same computations of the proximal operator and only have to use the modulus of $k^n$. For our regularization terms we have to use the shifted version $(|k^n| - 1)$, also compare [18, p. 54].

```
111 %initialization of the iteration step and cost difference
112 iter = 1;
113 cost_diff = 1e16;
114
115 % initial values for FPGM and MFPGM
116 if strcmp(algo,'fpgm') == 1 || strcmp(algo,'mfpgm') == 1
117 t0=1;
118 y0=init;
119 end
120
121 %bound the algorithm using a while loop with iter_max and cost_diff
122 while ((iter <= iter_max)  && (cost_diff >= tol))
123
124    %% PGM
125    if strcmp(algo,'pgm') == 1
126
127     % gradient step
128     m = (conj(S).*Z - conj(S).*dgt( idgt(S.*init,h,a,Ls) ,g,a,M,Ls));
129     m = init + 2*m/Cest;
130
131     % proximal operator
132     if normw == '1'
133     Phase = angle(m-1);
134     m = thresh(abs( m-1 ),lambda,'soft');
135     m = m.*exp(1i*Phase);
136     m = m+1;
137     elseif normw == 'abs1'
138     Phase = angle(abs(m)-1);
139     m = thresh(abs( abs(m)-1 ),lambda,'soft');
140     m = m.*exp(1i*Phase);
```

```
141      m = m+1;
142      elseif normw == '2'
143      m = (m + 2*lambda) ./ (1+2*lambda);
144      elseif normw == 'abs2'
145      m = (abs(m) + 2*lambda) ./ (1+2*lambda);
146      end
147
148     %% FPGM
149     elseif strcmp(algo,'fpgm') == 1
150
151      % gradient step
152      m = conj(S).*Z - conj(S).*dgt( idgt(S.*y0,h,a,Ls) ,g,a,M,Ls) ;
153      m = y0 + 2*m/Cest;
154
155      % proximal operator
156      if normw == '1'
157      Phase = angle(m-1);
158      m = thresh( abs(m-1) ,lambda,'soft');
159      m = m.*exp(1i*Phase);
160      m = m+1;
161      elseif normw == 'abs1'
162      Phase = angle(abs(m)-1);
163      m = thresh( abs(abs(m)-1) ,lambda,'soft');
164      m = m.*exp(1i*Phase);
165      m= m+1;
166      elseif normw == '2'
167      m = (m + 2*lambda) ./ (1+2*lambda);  ;
168      elseif normw == 'abs2'
169      m = (abs(m) + 2*lambda) ./ (1+2*lambda);
170      end
171
172      t = (1+sqrt(1+4*t0^2))*0.5 ;
173      y0 = m + (t0-1)/t*(m-init);
174      t0 = t;
175
176     %% MFPGM
177     elseif strcmp(algo,'mfpgm') == 1
178
179      % gradient step
180      temp = conj(S).*Z - conj(S).*dgt( idgt(S.*y0,h,a,Ls) ...
             ,g,a,M,Ls) ;
181      temp = y0 + 2*temp/Cest;
182
183      % proximal operator
184      if normw == '1'
185      Phase = angle(temp-1);
186      temp = thresh(abs( temp-1 ),lambda,'soft');
187      temp = temp.*exp(1i*Phase);
188      temp=temp+1;
189      elseif normw == 'abs1'
190      Phase = angle(abs(temp)-1);
191      temp = thresh( abs(abs(temp)-1) ,lambda,'soft');
192      temp = temp.*exp(1i*Phase);
193      temp= temp+1;
194      elseif normw == '2'
195      temp = (temp + 2*lambda) ./ (1+2*lambda)  ;
196      elseif normw == 'abs2'
```

```
197       temp = (abs(temp) + 2*lambda) ./ (1+2*lambda)  ;
198       end
199
200       t = (1+sqrt(1+4*t0^2))*0.5 ;
201
202       % Monotonicity :
203       if strcmp(normw,'abs1')==1 || strcmp(normw,'abs2')==1
204       Em = norm( idgt(S.*init,h,a,Ls) - z )^2 + lambda* norm( ...
             abs(init(:))-1, p )^p ;
205       Ez = norm( idgt(S.*temp,h,a,Ls) - z )^2 + lambda* norm( ...
             abs(temp(:))-1, p )^p  ;
206       else
207       Em = norm( idgt(S.*init,h,a,Ls) - z )^2 + lambda* norm( ...
             init(:)-1, p )^p ;
208       Ez = norm( idgt(S.*temp,h,a,Ls) - z )^2 + lambda* norm( ...
             temp(:)-1, p )^p  ;
209       end
210       if Ez < Em
211           m=temp;
212       else
213           m=init;
214       end
215        % Update:
216       y0 = m + t0/t*(temp-m) + (t0-1)/t*(m-init);
217       t0 = t;
218
219      end
220
221       %Update to get into the next step
222       init=m;
223       iter = iter + 1;
224
225       %evaluation of the subsequent iteration step
226       if strcmp(normw,'abs1') == 1 || strcmp(normw,'abs2') == 1
227       eval(iter) = norm( idgt(S.*init,h,a,Ls) - z )^2 + lambda* ...
             norm( abs(init(:))-1, p )^p ;
228       else
229       eval(iter) = norm( idgt(S.*init,h,a,Ls) - z )^2 + lambda* ...
             norm( init(:)-1, p )^p ;
230       end
231       %%
```

**Update:**
In the following, we update the relative error (7.3) of the cost function. It is important to mention that this error could be 0 for the sake of the MFPGM. The Algorithm 4 of Chapter 6 chooses every time the smallest value, i.e. $m^n = \arg\min\{\Phi(m) : m = z^n, m^{n-1}\}$. If the function $\Phi(m)$ increases it would be possible that the same value $m^{n-1}$ is taken more often (stays constant), i.e. $m^n = m^{n-1}$ and therefore the difference between two following steps $\Phi(m^{n-1}) - \Phi(m^n) = 0$. This would lead to a standstill of our MATLAB code, due to Line 122. For convenience we take a constant relative error equal to 1 in the case of MFPGM.

```
232      if strcmp(algo,'mfpgm') == 1
233         % cost_diff of mfpgm can be 0, then while loop would stop
```

```
234         cost_diff = 1;
235     else
236          cost_diff = norm(eval(iter-1)-eval(iter))/ ...
                norm(eval(iter-1)) ; %/norm
237     end
238
239       % displays a warning, if cost function increases
240       if eval(iter-1)<eval(iter)
241           disp(['Warning: Costfunction increases : iter = ...
                ',num2str(iter)])
242       end
243 end
244
245 disp(['cost function of estimated mask with algorithm ',algo,' and ...
      norm ' ,normw, ' : ',num2str(eval(iter-1))])
246
247 eval = eval(1:end);
248 % by Proximal gradient methods transformed signal:
249 rs=idgt(init.*S,h,a,Ls);
250
251 end
```

The last thing we want to mention is the following figure (Figure 7.1), where we want to underline the improvement from proximal gradient method over fast proximal gradient method to monotone fast proximal gradient method. For the compilation of this figure we used $\lambda = 0,0001$ and the 2-norm.

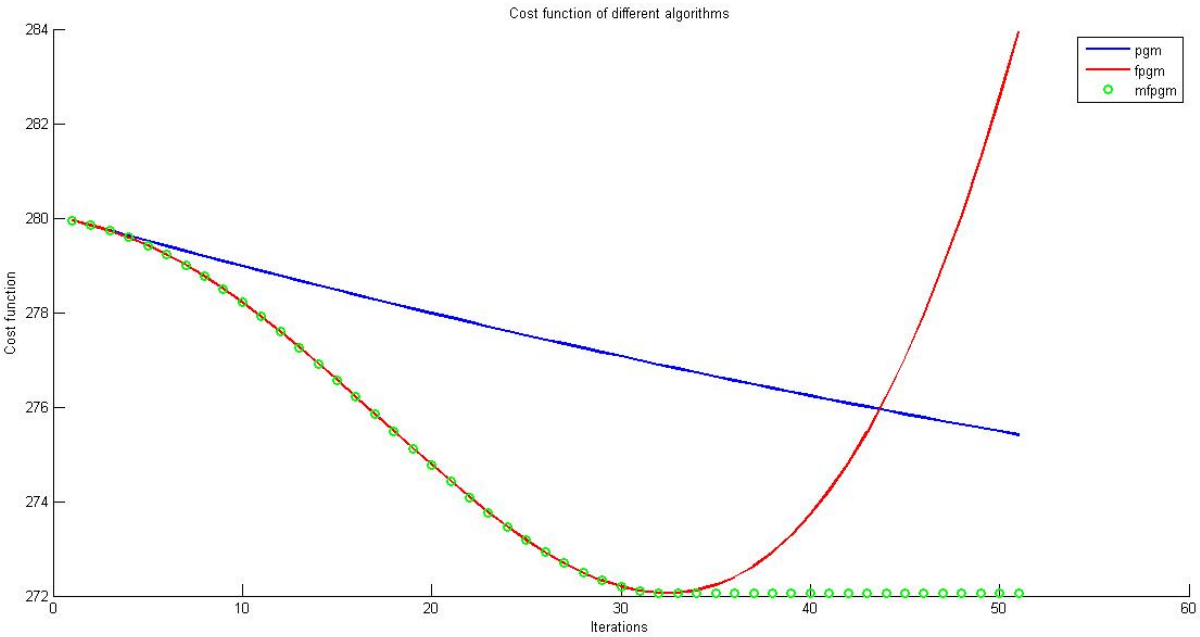


Figure 7.1: Cost function of the different proximal algorithms after 50 iterations.

It is clearly visible that FPGM and MFPGM are initially the same and faster than PGM. Then the fast method starts diverging, whereas the monotone version is still

converging.
After the introduction to this MATLAB code, we are now ready to do some experiments, considering different dynamics of the different signals and also some signal transformation with many and less overtones.

# 8 Applications obtained through MATLAB Code

In this chapter we are going to do some experiments, considering the different regularization terms and the different iterative algorithms of the preceding Chapter 7. We divide this chapter in two subsections, where we focus on different things. On the one hand we consider difference resulting from different dynamics, i.e. how the Gabor multiplier changes when the usage of the same signals, but in different dynamics. On the other hand the difference of masks will be caused due to a swap of source and target signal. First some generalities that hold for both subsections.
Again, as in Chapter 5, we consider the violin and the flute which play the same note (C6). All original sounds are from Vienna Symphonic Library [1]. The spectrograms of the original sounds can be viewed in the chapter named before in Figure 5.2. Some basic parameters for the program `gabmult.m` are:

```
a=256;
M=1024;
g=firwin('hann',M);
h=gabdual(g,a,M);
[s,z]=samesize_power2(f_1,f_2,sec,cut);
lambda=input('Which lambda would you prefer? - Choose lambda ...
    between 0 and 1: ');
if lambda>1 || lambda<0
    error('lambda should be between 0 and 1')
end
tol= 1e-2; %Default value is 1e-2
iter_max= 50; %Default value= 100
```

If we run the program it is obvious, that if we take the regularization parameter $\lambda$ close to 0 (means very small like $10^{-6}$), we are close to reconstruction of the target signal, starting from the source signal. Whereas $\lambda = 1$ causes "no transformation" and stays at the source sound. If we choose $\lambda$ between 0 and 1, we get a sound, which is between the source and target signal. Therefore the parameter $\lambda$ can be viewed as an interpolation parameter between source and target signal.
Choosing $\lambda$ very small, the different norms and algorithms yield almost the same result, even for listening and visualization via MATLAB. For this reason, we are going to choose $\lambda$ between 0 and 1, because we want to describe the occurring differences (audible and visible). We start with the discussion of the behavior of the different directions of transformation, i.e. we swap source and target and show the properties of the masks.

## 8.1 Swap of Source and Target Signal

One point that should be mentioned before we discuss the interchange is, that everything that is mentioned concerning impression of hearing is subjective and should be checked for own processing. First we do an experiment and try to find an inverse of $\tilde{m}$ which solves

$$\tilde{m} = \arg\min_{m} \|Z - mS\|_2^2 + \lambda\, r(m).$$

An inverse would denote obtaining $S$ from the multiplication of a mask with the target signal $Z$. Then we will see and hear through checking with MATLAB that this will not work! Both, multiplying the target signal with the obtained mask, means $\tilde{m}Z$,

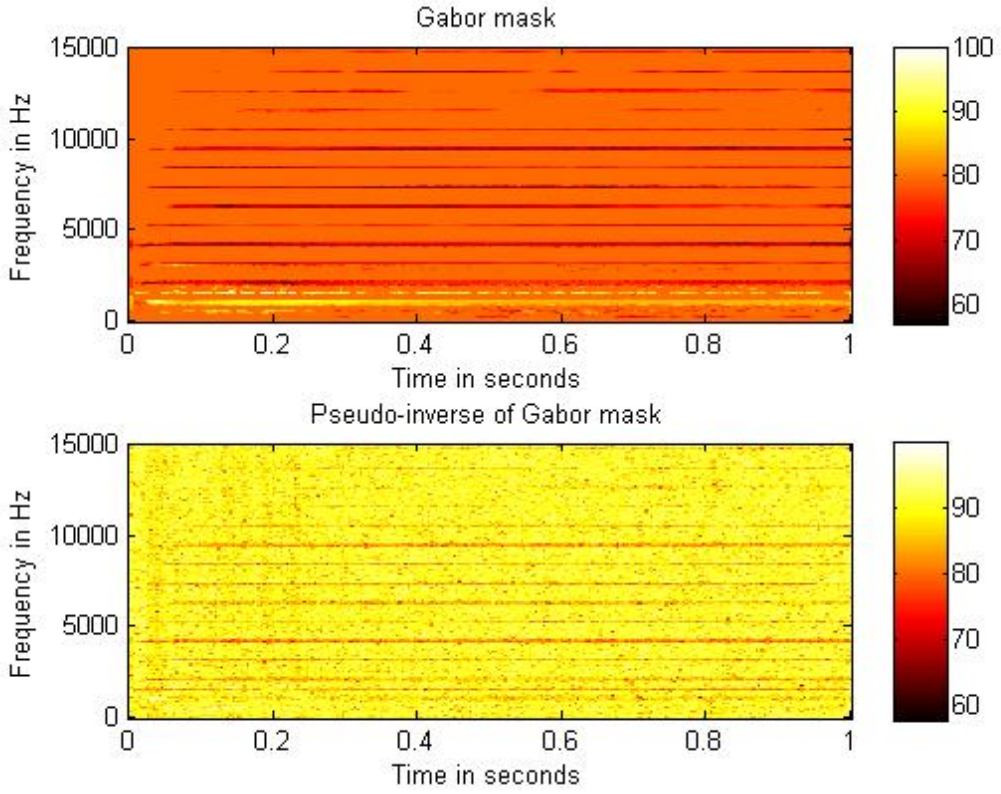


Figure 8.1: Comparison of the Gabor mask and its pseudo-inverse.

leads to a diffuse sound. Also if we compute the pseudo-inverse of $\tilde{m}$, written as $\tilde{m}^\dagger$, we obtain a strange sound computing $\tilde{m}^\dagger Z$. This implies that there exists no "inverse" version of the mask $\tilde{m}$ or no "inverse" computational step which yields the input sound $S$. Therefore we have to do an own computation for the other direction, i.e. the swap of input and output signal. In Figure 8.1 we can compare the mask and its pseudo-inverse (computed from the direction violin to flute with $\lambda = 0.1$, 1-norm and MFPGM) and below in Figure 8.2 the spectrogram of the obtained sound of $\tilde{m}^\dagger Z$.

In the inverted mask of Figure 8.1 we can see a lot of noise, which destroys the subtle structure of the Gabor mask. And now it is conceivable that the spectrogram of this mask (Figure 8.2) multiplied with the target signal yields nothing meaningful. Although Figure 8.2 does not look so bad, the resulting sound is not agreeable to listen to.

Now we do research, where we compute all transformations of both directions, i.e. from violin to flute and vice versa. One observation made so far, considering these transformation is that it plays indeed a role in which direction we transform. Recognized through hearing so far, the transformation from violin to flute is "easier" than the transformation from flute to violin. (Here the word *easier* will be specified with the explanation of Table 1 and 2.) Maybe that is due to the different richness of overtones. In the following we are going to manifest this fact through some examples. First we are going to state tables, where we consider the different norms and algorithms. An associated value of $\lambda$ is inserted, when the transformed signal is audibly identified as the target signal. Here we use the property of $\lambda$ balancing between "no transformation"

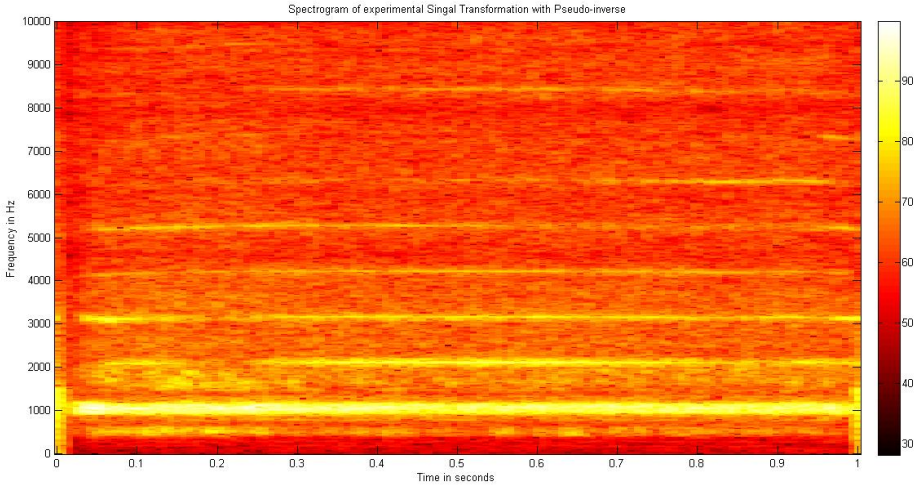


Figure 8.2: Spectrogram of $\tilde{m}^{\dagger}Z$.

($\lambda = 1$) and "perfect reconstruction" ($\lambda = 0$). Reconstruction has to be considered in the point of view that we are doing a transformation from the start signal to the target signal and the smaller we choose $\lambda$ the closer we reach the target signal through transformation. Therefore we say the target signal is reconstructed through the input/start signal. One should keep in mind that this is again a subjective observation and also strongly connected to the instruments we are going to use. Here $p$-norm denotes $\|m-1\|_p^p$ and $|p|$-norm is meant to be $\||m|-1\|_p^p$. The acronyms PGM, FPGM and MFPGM denote the algorithms that we have introduced in Chapter 6. In Table 1 we are transforming the violin into the flute and we insert the values of $\lambda$, where the sound via transformation seems to be the target sound.

| | **PGM** | **FPGM** | **MFPGM** |
|---|---|---|---|
| **1-norm** | 0.01 | 0.01 | 0.009 |
| **\|1\|-norm** | 0.01 | 0.01 | 0.25 |
| **2-norm** | 0.003 | 0.0025 | 0.0025 |
| **\|2\|-norm** | 0.004 | 0.005 | 0.009 |

Table 1: Transformation from violin to flute. Values of $\lambda$ are entered, below which the transformation to the target signal audibly reminds of a flute.

Here (Table 1) it is visible that for norms without absolute value of the mask, we get higher values of $\lambda$ and therefore earlier a good transformation for PGM and FPGM, compared to MFPGM. But if we look at the values of the $|p|$-norms, we earlier, means for a comparatively big $\lambda$, obtain good transformation results for MFPGM. Now we state the table in the direction of flute to violin (cf. Table 2).

| | **PGM** | **FPGM** | **MFPGM** |
|---|---|---|---|
| **1-norm** | 0.2 | 0.2 | 0.1 |
| **\|1\|-norm** | 0.025$^\star$ | 0.05$^\star$ | 0.1 |
| **2-norm** | 0.009 | 0.009 | 0.0025 |
| **\|2\|-norm** | 0.005$^\star$ | 0.002$^\star$ | 0.001 |

Table 2: Transformation from flute to violin. Values of $\lambda$ are entered, below which the transformation to the target signal audibly reminds of a violin.

For this direction we have almost the same observation, concerning the absolute value of the Gabor mask. This means that MFPGM for $p$-norms ($p = 1, 2$) is slower than the other algorithms, whereas for $|p|$-norms ($p = 1, 2$) it shows earlier good results. For the with $\star$ marked $\lambda-$entries we have a very unclean violin sound which doesn't get better, if we try to use very small $\lambda$. Therefore the first value for $\lambda$ is taken, where it reminds of a violin. Hence the value for MFPGM of the |2|-norm is smaller than the values for PGM and FPGM. In following Figure 8.3 we can see that the spectrogram of MFPGM has more clear lines and moreover the blotted sound of the other algorithms (in Table 2 marked with $\star$) is visible.

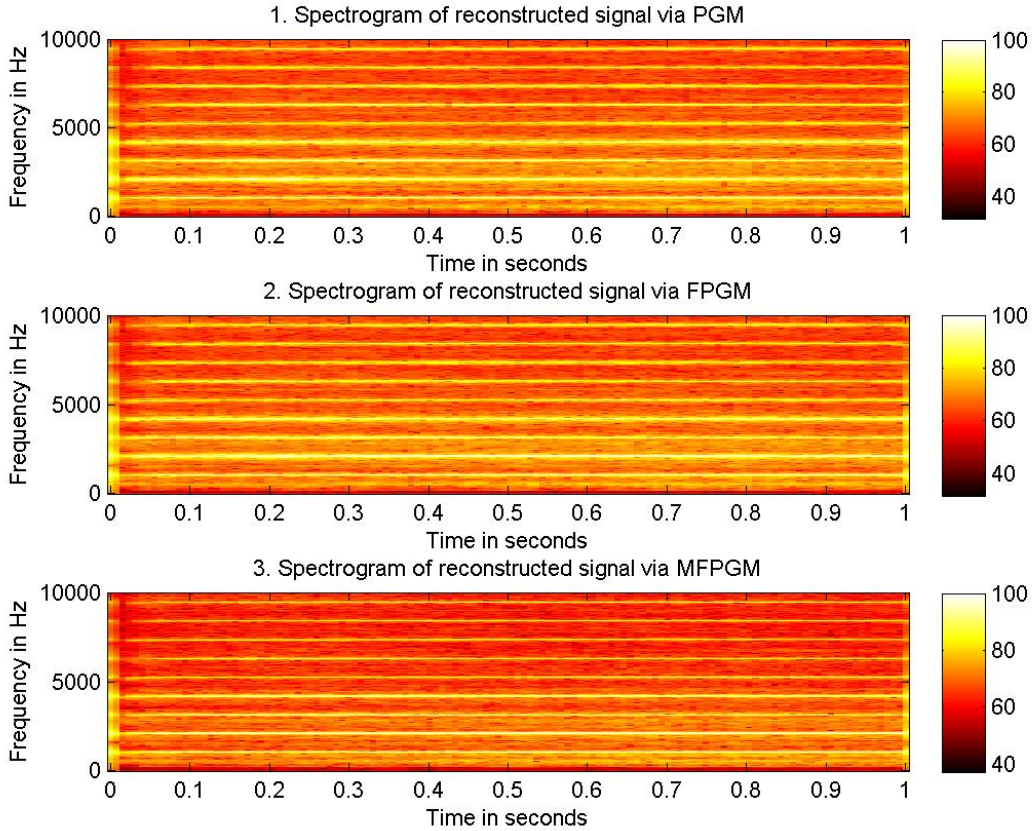


Figure 8.3: Spectrogram of the transformations from flute to violin for all different algorithms using |2|-norm and $\lambda = 0.001$.

A general trend is that the transformation from flute to violin happens slightly faster

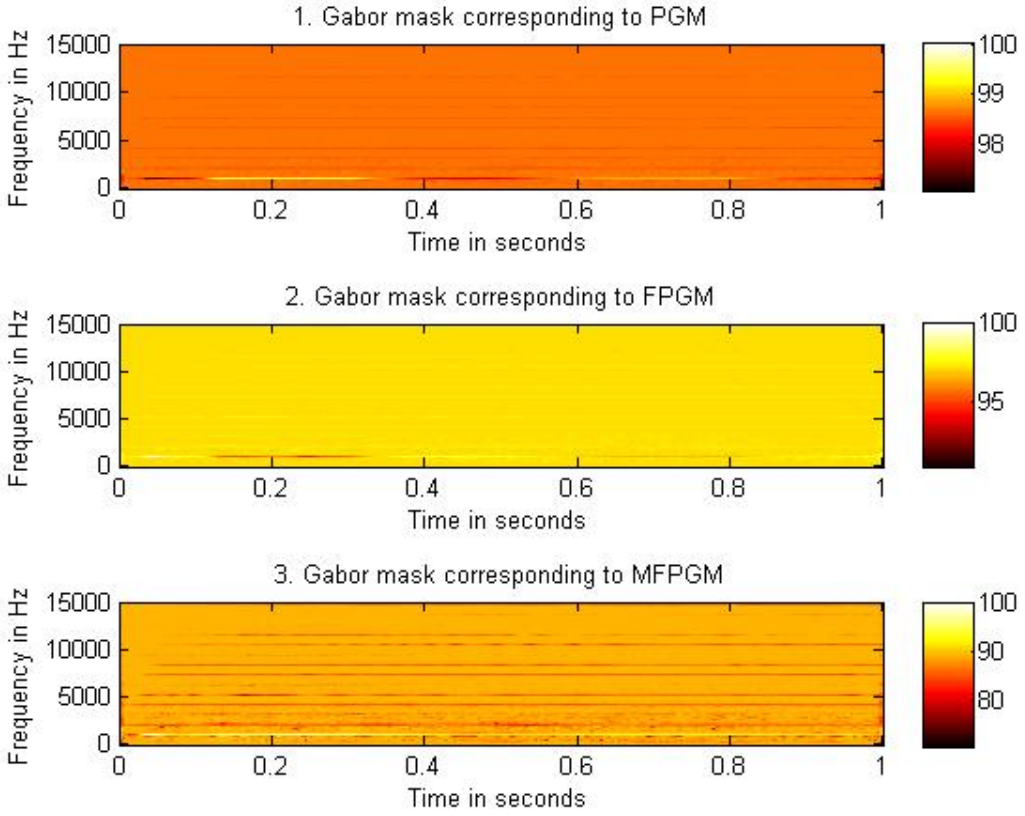


Figure 8.4: Gabor mask of the transformation from violin to flute.

than in the other direction. Slightly faster in the sense of entering the same parameters in the MATLAB code and be audibly closer to the target sound, i.e. the violin is reached earlier (with bigger $\lambda$) than the flute. This means that it is indeed relevant which instruments we use as source and which we are going to use as target. It is unclear if this is due to the overtones. For more research in this direction, there have to be performed more experiments considering more different instruments, but this would go beyond the scope of this thesis.
Another observation, concerning the sound of the transformation is that the oscillations for some $\lambda$ are clearly audible in the case of $p-$norms (no absolute value of the Gabor mask). Of course we state an example, where this oscillations are visible. In Figure 8.4 and 8.5 we considered $\lambda = 0.1$ and the 2-norm to show that there exist of course differences between the direction of transformation, due to the visualization of the oscillations. For the direction of flute to violin, more oscillations are visible than in the other direction. This observation is also recognizable if one listens to the sounds.
Finally we want to show that the MATLAB code works for both directions very well and its efficiency is independent of the used signal. In the following figures we show the sound which we want to reach by transformation followed by the results with MFPGM which worked best (cf. Table 1 and 2). We take an intermediate value for $\lambda = 0.1$. First we want to transform into the flute and therefore show Figure 8.6. The spectrograms of the reconstructed flute is displayed in Figure 8.7. Here the explicit characteristics

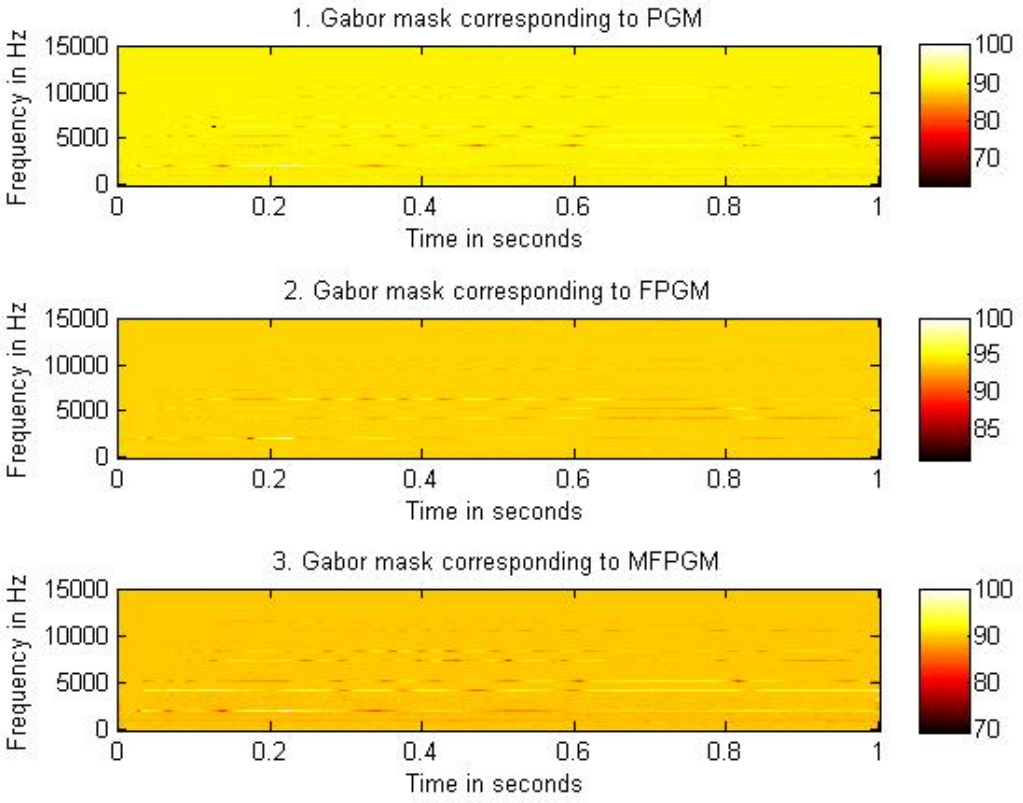


Figure 8.5: Gabor mask of the transformation from flute to violin.

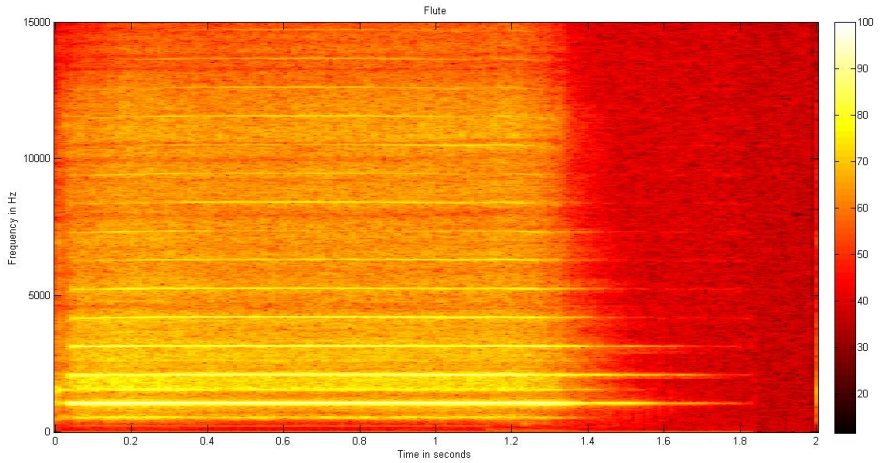


Figure 8.6: Original flute sound with original length.

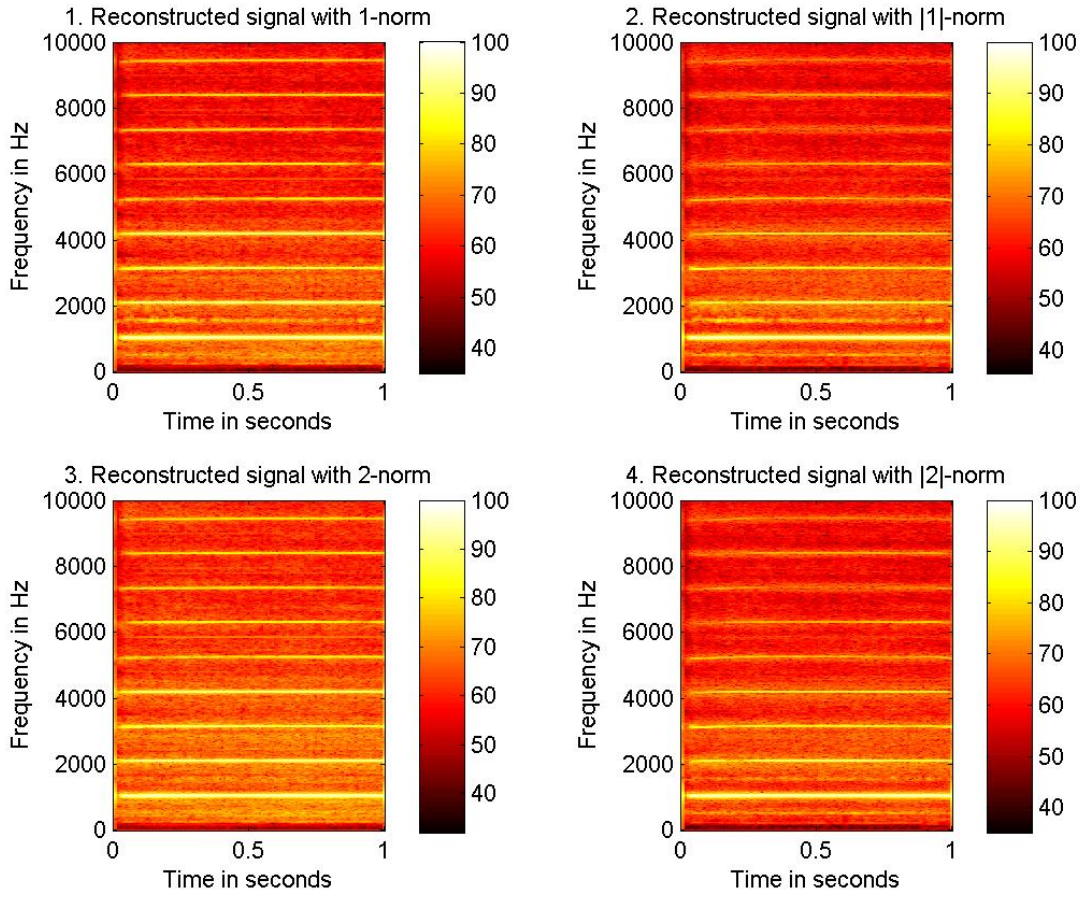


Figure 8.7: Reconstructed flute of source sound violin with MFPGM and $\lambda = 0.1$ for all different norms.

of the overtones becomes visible. For the $p$-norm (for $p = 1, 2$ without modulus), the differences between the overtones are not that visible, because the yellow lines are the partial tones and the area between these tones is also slightly yellow. On the contrary, the absolute value of the mask in the norms yield a clear visible distinction between the overtones, where the area between changes color. This shows the characteristic of a flute played with a high pitch, i.e. less accented overtones and the fundamental frequency accented like a sinus. Considering the associated Gabor masks, Figure 8.8 shows that all the masks have intensity near 100 at the fundamental note. This implies that the fundamental information of a sound, here used from the violin, only has to be slightly modulated for the target sound, the sound of the flute. In other words, the flute inherits almost everything (=intensity near 100) of the fundamental tone of the violin. Because the violin is richer on overtones, the masks have to modify its other overtones to get the sound of a flute. The varying basic color of the different masks are due to the different solutions to our problem solved with different norms and different algorithms. For plotting the function we use a logarithmic scale which sets the values in relation to the maximal entry (cf. Chapter 5 p. 24). If there exists a big maximum concentrated in one point, then the rest is comparatively small and therefore gets less intensity compared to the maximum. Consequently some basic colors of the masks seem very dark or very clear, which has nothing to do with the level of transformation. For comparing the level of "efficiency" of such a transformation (i.e. if one compares the different $\lambda$ of the Tables 1 and 2, where the algorithms have different $\lambda$ where they "turn" into or reach the target signal), we have to take a closer look at the modification of the frequencies, especially the modification of the overtones. Now we are going to look in the "other" direction, i.e. we do the transformation from the flute to the violin. Therefore we show the original sound of the violin again in Figure 8.9. The violin generates a more rough sound due to the high number of overtones and therefore the original sound very clearly points out the usage of more than one partial tone. Very light-colored are the first four partial tones.
In Figure 8.10 we observe now the transformation corresponding to the different norms for $\lambda = 0.1$. Here the visible difference is not that big, but one could see that the area between the partial tones turns more and more the same color as the partial tones themselves, because this generates the roughness of the sound of an violin. This is the total contrary of what we mentioned before, when we transformed into the flute. Considering the associated masks in Figure 8.11, it can be seen that the fundamental note has to be absorbed a bit. This is due the fact that the fundamental note of the flute is very sinusoidal, i.e. very dominating and the violin needs more a vibration of a saw-teeth function and therefore weights the partial tones slightly different. In this figure the different basic colors are because of different maxima of the vectors of the mask which influence the program we are using for the logarithmic spectrogram (cf. Chapter 5 p. 24). Here the 1-norm and the $|1|$-norm have similar maxima, because their basic color is nearly the same. The 2-norm seems to have no big difference between the maximum and the other values and therefore the basic color shows high intensity. Contrary the $|2|$-norm is displayed very dark. This is due to a point maximum, a maximum which is very highly concentrated in one point and comparatively small in other values. Therefore the other values get low intensity and the whole image turns very dark. For comparisons of the efficiency, means which algorithm reaches the target signal with a bigger $\lambda$ of the different algorithms and different norms, we can compare

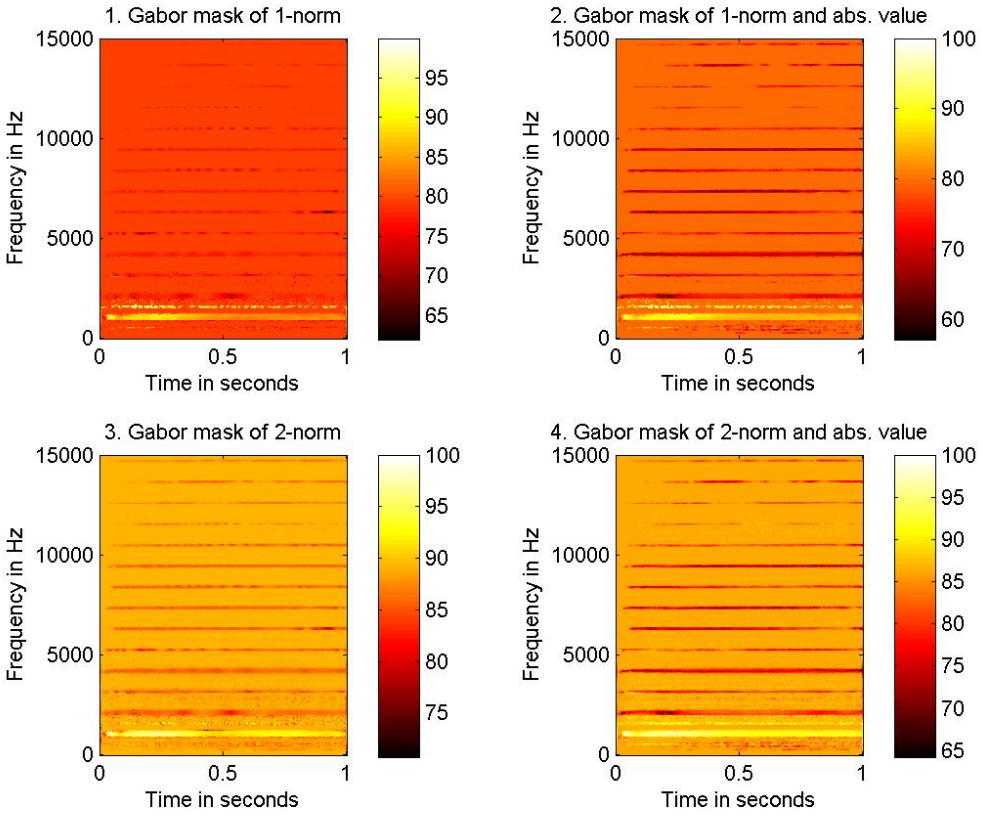


Figure 8.8: Transformation from violin to flute. Gabor mask with MFPGM and $\lambda = 0.1$ for all different norms.

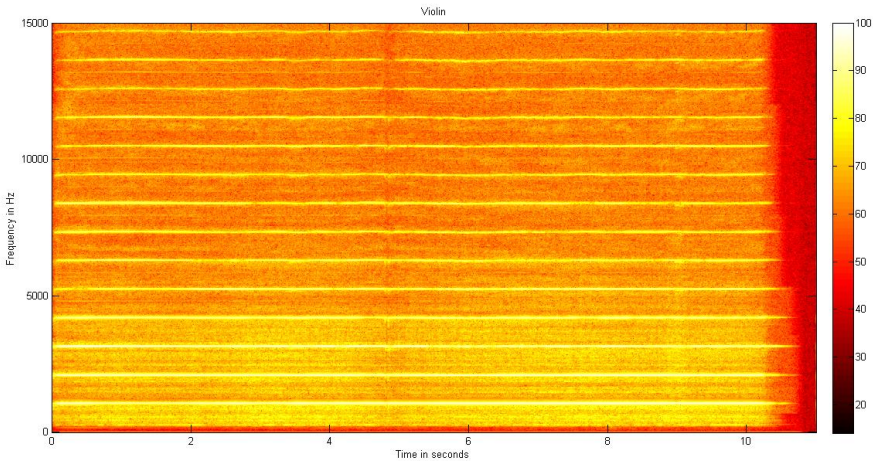


Figure 8.9: Original violin sound with original length.

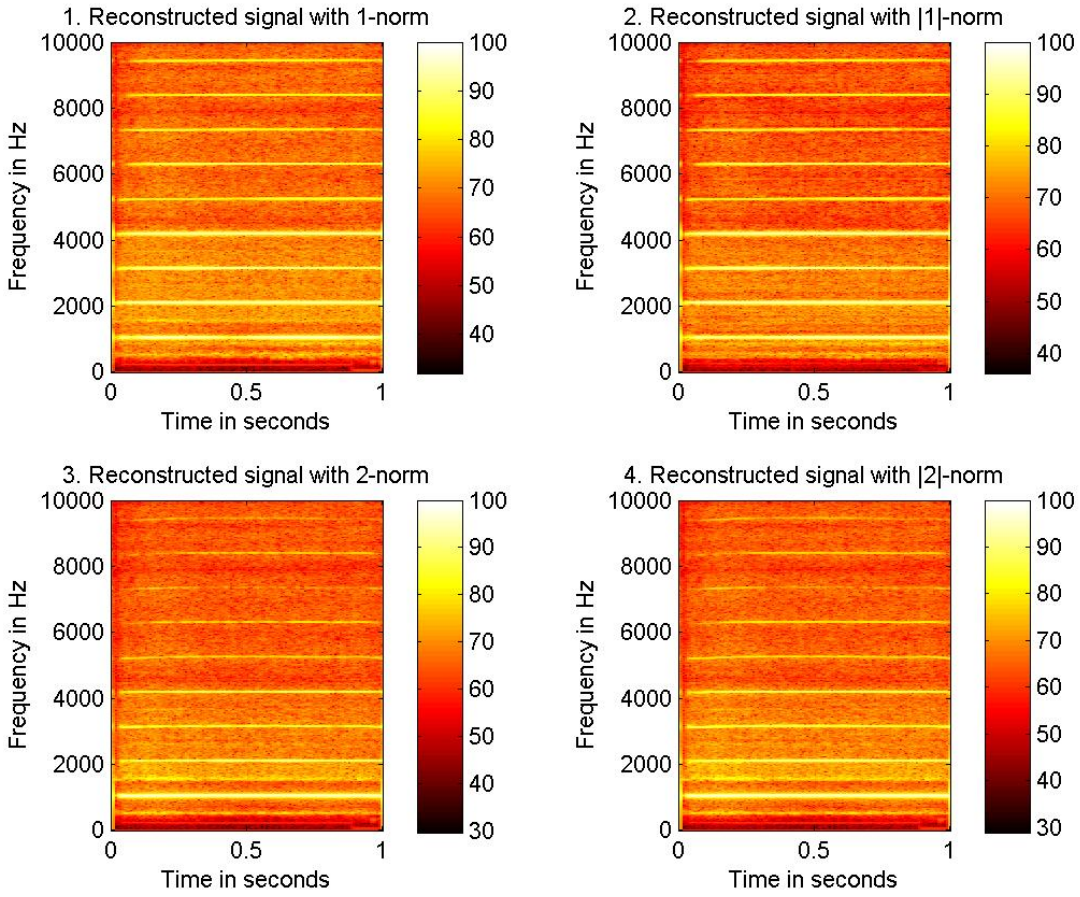


Figure 8.10: Reconstructed violin from source sound flute with MFPGM and $\lambda = 0.1$ for all different norms.

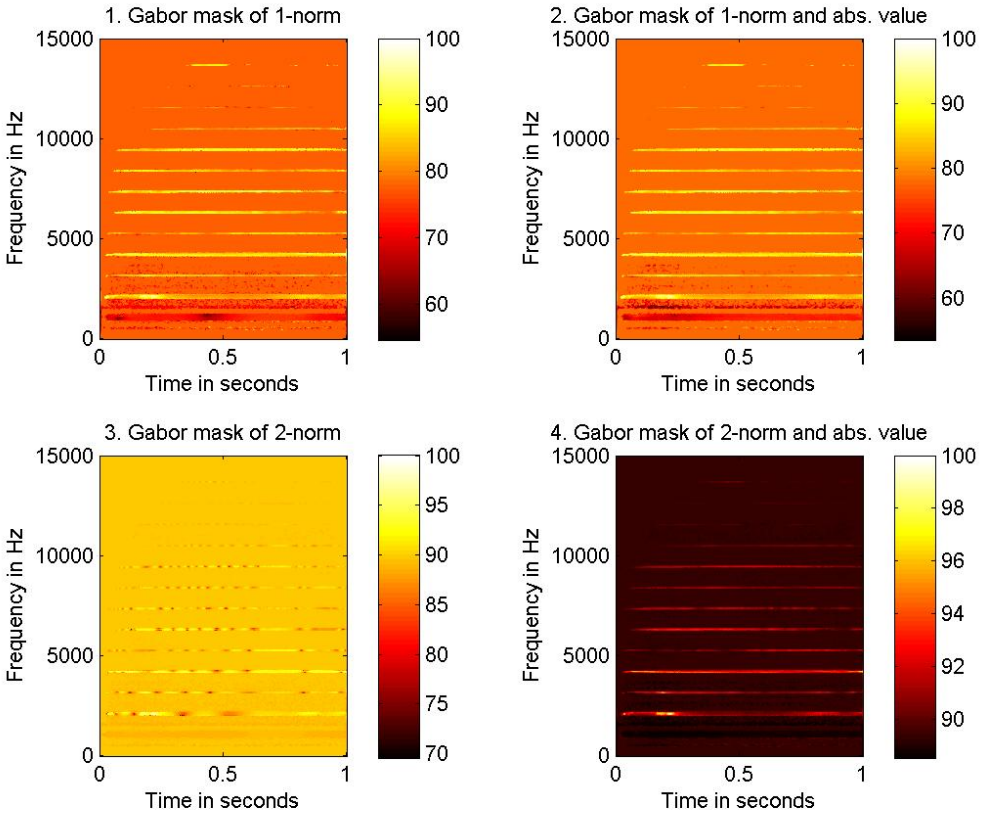


Figure 8.11: Transformation from flute to violin. Gabor mask with MFPGM and $\lambda = 0.1$ for all different norms.

the modification of the overtones. For the 1-norms with and without modulus, it seems that a modification of every overtone took place and therefore we nearly completely transformed the input into the target signal with $\lambda = 0.1$. On the contrary the 2-norms have not reached the target signal with this $\lambda$, cf. Table 2. After discussing the impact of the direction on the parameter $\lambda$, on the mask and showing that it makes a difference which instrument is transformed in the other, we now consider consequence of different dynamics.

## 8.2 Consequences of Different Dynamics

In this section we are discussing the impact of dynamic on the signal transformation. We use the Italian way of denomination, i.e. *piano* stands for playing an instrument soft, whereas *forte* means playing an instrument loudly. We are going to transform the violin into the flute, because in the last section some noise arose when we transformed from the flute to the violin. One thing that should be mentioned is the fact that a musician has to play an instrument in a different manner, if he plays it piano or forte. That is why differences in the discussion of this subsection should and will occur. First we look at the transformation, when the instruments are played in piano. Therefore we show the original sounds in Figure 8.12 and 8.13.

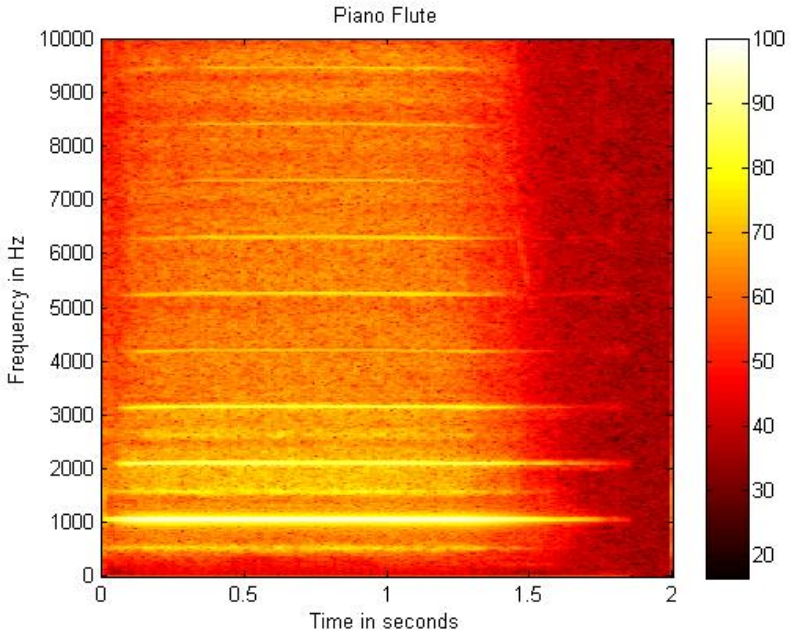


Figure 8.13: Original flute sound, played piano.

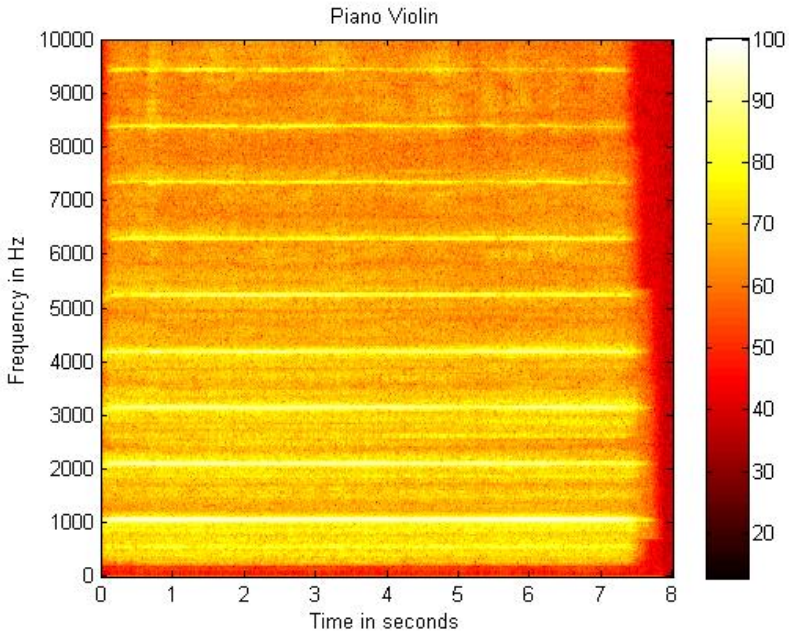


Figure 8.12: Original violin sound, played piano.

It is clearly visible that the flute strongly forms the fundamental tone and that the violin spreads its energy among all overtones. To get an impression of the transformation in piano, we again provide a table, cf. Table 3. This table shows in the columns the different gradient methods and in the rows the different regularization terms which we are using. The entries are the regularization parameters $\lambda$ which should normally move between 0 and 1, where 1 denotes "no transformation" and 0 is "full reconstruction". Here we insert the value of $\lambda$ where the sound of the transformed signal clearly

| | **PGM** | **FPGM** | **MFPGM** |
|---|---|---|---|
| **1-norm** | 0.02 | 0.01 | 0.01 |
| **\|1\|-norm** | 0.01 | 0.01 | 0.3 |
| **2-norm** | 0.02 | 0.015 | 0.015 |
| **\|2\|-norm** | 0.0035 | 0.0035 | 0.15 |

Table 3: Audible transformation from piano violin to piano flute. The values show the balancing parameter $\lambda$ between $(0, 1)$, where a reconstruction of the target signal is audibly obtained.

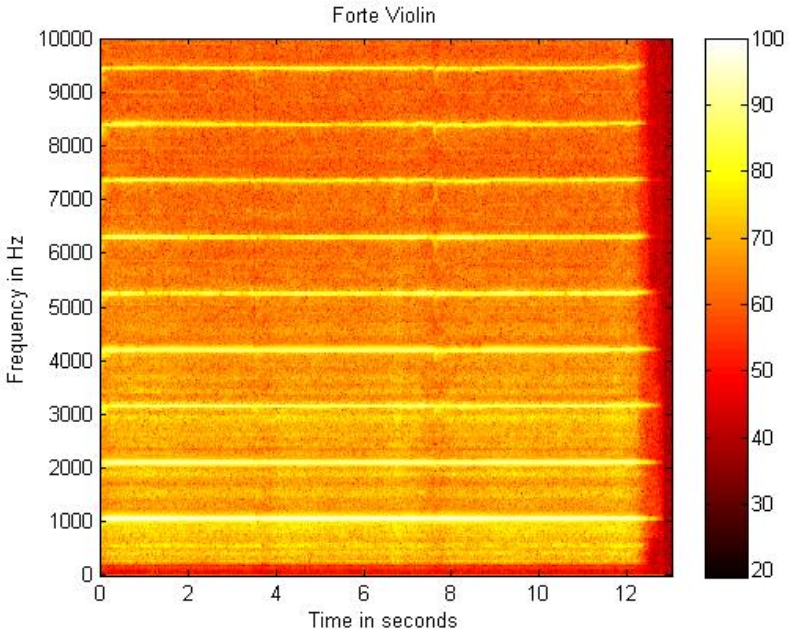


Figure 8.14: Original violin sound, played forte.

reminds of the sound of the target signal. Table 3 shows entries which hardly differ. One observation that we also made in the previous subsection, is that the algorithm MFPGM is faster in the cases of |p|-norms ($p = 1, 2$). In this case faster means that if we try every value of $\lambda$ for our transformation going from 1 to 0, we would reach the target signal (after transformation) with a bigger value for $\lambda$ for $|p|$-norms in the case of MFPGM, as for other norms. A further observation is that the values of $\lambda$ for the transformation with the |2|-norm and the algorithm PGM and FPGM are much smaller than the others. To compare these results with the forte played instruments, we will show the table which we obtain for nearly reconstruction of the forte played flute. But before we show the original sounds in a Spectrogram 8.14 and 8.15.
Again characteristic spectrograms occur, where the energy of the flute focus on the fundamental tone and the forte violin shows all overtones in the same intensity. What could be observed in Figure 8.15 is the input of the tongue that is audible. The tongue has to give some pressure, such that the sound of the flute is loud. Therefore the input is visible as a small wave at the beginning of the sound. Now we show in Table 4 the forte played instruments and discuss the differences we are observing.
The MFPGM is again faster if we use $|p|-$norms. Compared to the Table 3 of the piano

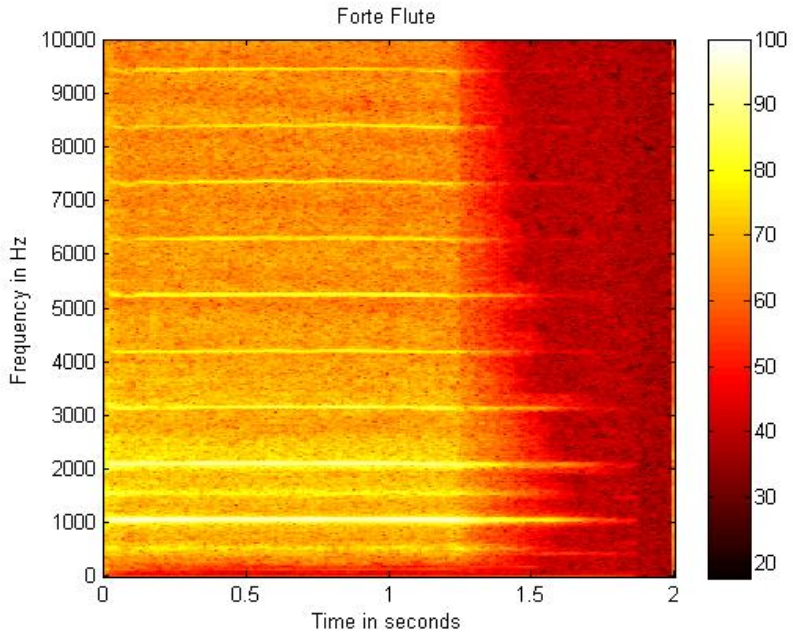


Figure 8.15: Original flute sound, played forte.

| | **PGM** | **FPGM** | **MFPGM** |
|---|---|---|---|
| **1-norm** | 0.015 | 0.01 | 0.007 |
| **\|1\|-norm** | 0.01 | 0.01 | 0.2 |
| **2-norm** | 0.015 | 0.015 | 0.02 |
| **\|2\|-norm** | 0.0025 | 0.0025 | 0.08 |

Table 4: Audible transformation from forte violin to forte flute. The values show the balancing parameter $\lambda$ between $(0, 1)$, where a reconstruction of the target signal is audibly obtained.

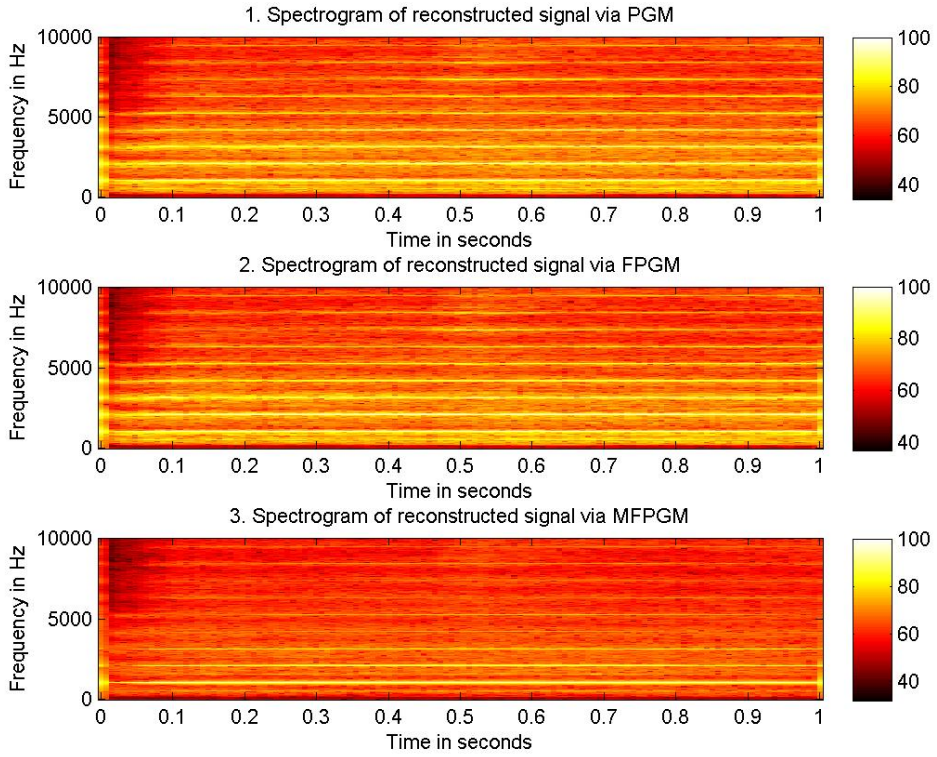


Figure 8.16: Piano transformation from violin to flute, |2|-norm and $\lambda = 0.1$.

transformation there are similar values for lambda in each entry. The differences of each entry in comparison using the same norm and same method are between $0.003 - 0.1$. Therefore we have the same tendencies in general. What should be mentioned is the fact that if one is listening to the original sound of the flute one clearly hears the noise of starting to blow wind in the instrument, because the tongue has to contribute for a sound played in forte. However, this noise vanishes through the transformation. Perhaps this is due to the fact that such a noise cannot be adapted from the violin and for the mask it is too difficult to create such kind of sound.
In the following Figure 8.16, we are going to show the piano transformation of the |2|-norm with $\lambda = 0.1$ where the PGM and FPGM have not reached the target signal yet, but the MFPGM indeed has. If we compare Figure 8.16 with the original sounds of Figures 8.12 and 8.13, then we clearly observe why the sounds of PGM and FPGM have not reached the target sound yet. Their spectrograms clearly resemble the violin, where the frequencies nearly have the same energy. On the other hand, if we look at the lower spectrogram of the MFPGM, we can see the importance and high energy of the first (fundamental) frequency.
Now we also compare the spectrogram of the same settings for the forte transformation, see Figure 8.17. Here it is visible that the lower spectrogram of MFPGM does not remind us of a flute at the first sight. This is due to the fact, that compared to Table 4 the value of $\lambda = 0.08$ is less than the value of $\lambda = 0.1$ of this figure. Therefore we have not reached the target signal yet. The energy of the spectrograms of PGM and FPGM are again distributed very regular, which resembles of a violin. The last spectrogram

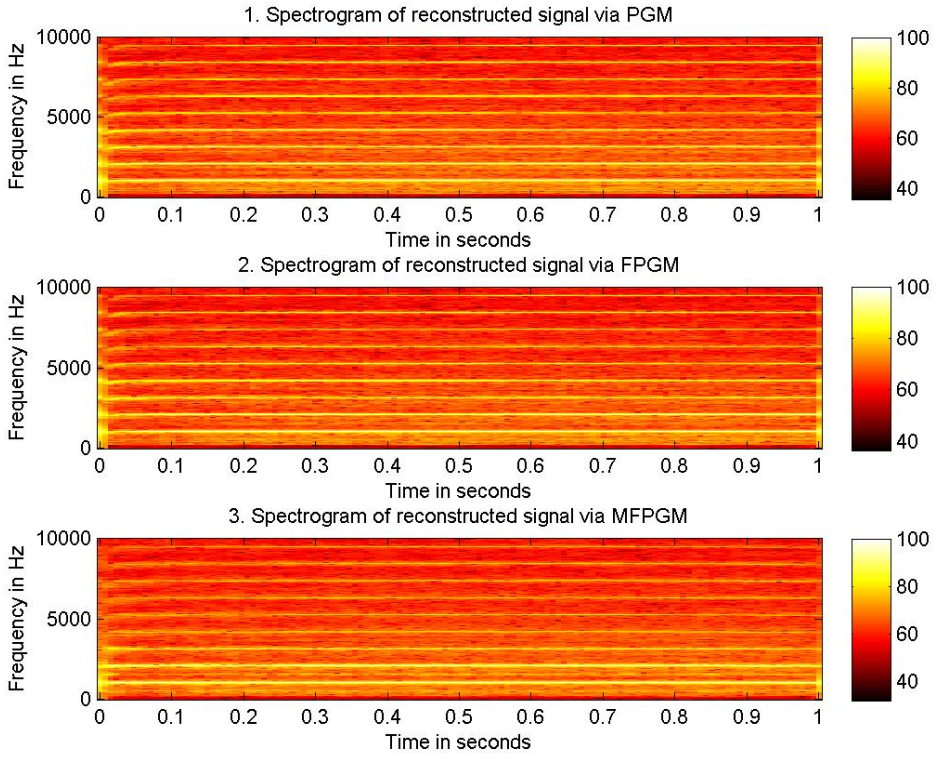


Figure 8.17: Forte transformation from violin to flute, |2|-norm and $\lambda = 0.1$.

(spectrogram of MFPGM) of Figure 8.17 also spreads its energy, but there is a small tendency, that more energy is focused on the first overtones.
Comparing the piano transformation and the forte transformation of |2|-norm with $\lambda = 0.1$, we can conclude that there is indeed a difference between the transformations. Not only for the visible spectrograms, but also for the audible sounds, where the flute has to be played different for a forte sound which gets completely lost in the transformation. What is also visible, is that PGM and FPGM take an earlier value of $\lambda$ (earlier in the sense of a bigger value for $\lambda$ decreasing from 1 to 0) in the forte transformation, view Figure 8.17, because there is not so "much" noise between the overtones which would be characteristic for the rough sound of a violin. On the other side, the MFPGM reaches earlier the target signal in the piano transformation which can be viewed in Figure 8.16. Here, the concentration of the energy on the fundamental tone is clearly stronger and therefore the transformed signal resembles a flute.
In the following Figures 8.18 and 8.19 we are going to compare Gabor masks. Generally it could be said that the masks of piano played instruments really look different from forte played masks. We are going to look at the transformation of violin into flute for the case of the 1-norm (because here we can see some oscillations which would not occur, if we took the modulus). Moreover we choose $\lambda = 0.1$ which is an intermediate value, because $\lambda$ can be chosen between 0 and 1.

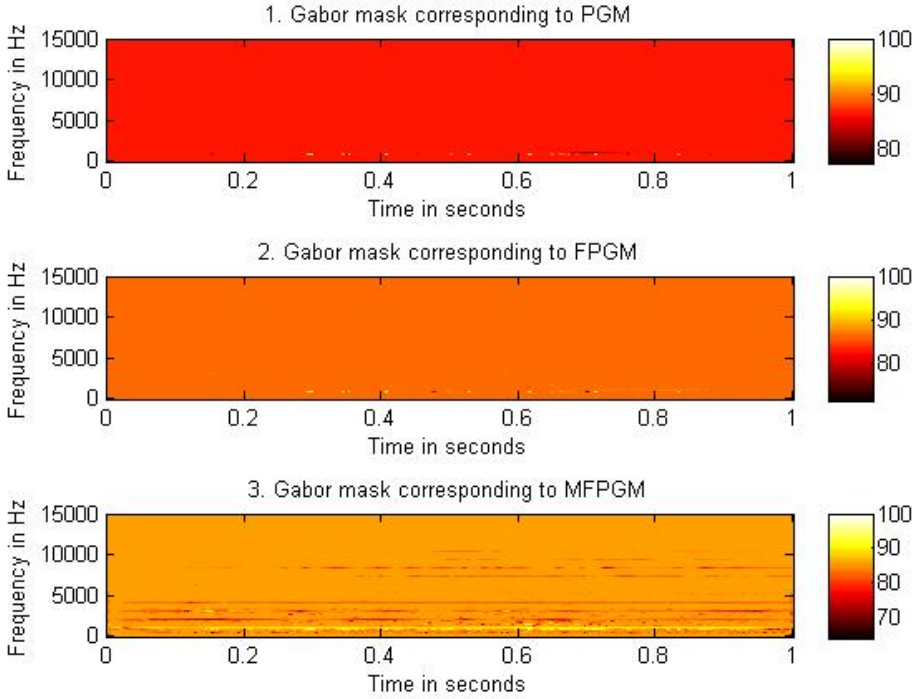


Figure 8.18: Gabor mask of the piano transformation from the violin into the flute with 1-norm and $\lambda = 0.1$.

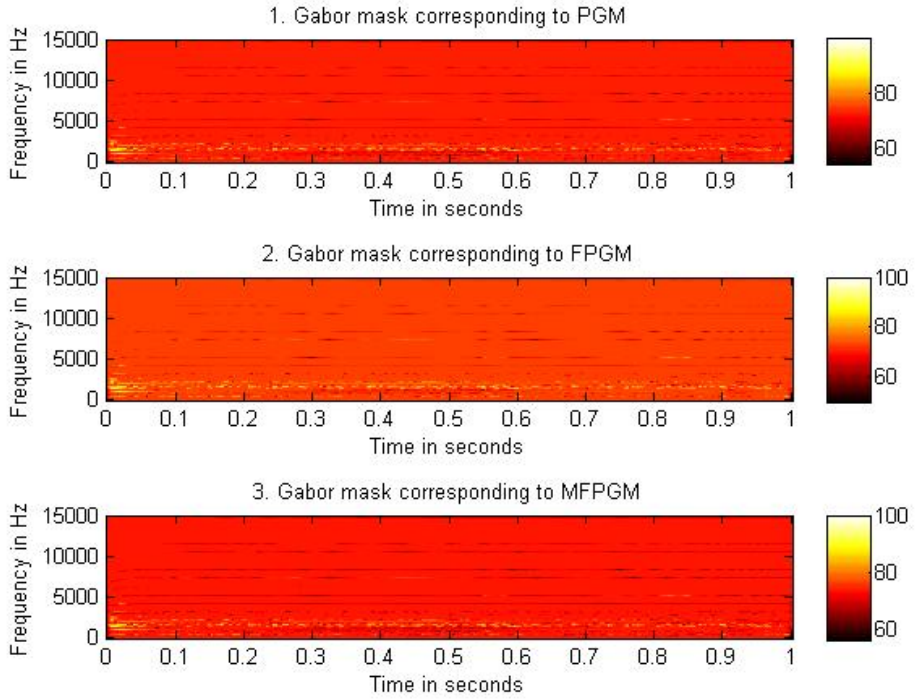


Figure 8.19: Gabor mask of the forte transformation from the violin into the flute with 1-norm and $\lambda = 0.1$.

The basic color of the Figures 8.18 and 8.19 are nearly the same which implies that they have nearly the same range of values, i.e. the maximum values of each mask-vector are similar. But what is obvious is the fact that there are a lot more oscillations in the case of the forte transformation, cf. Figure 8.19. This was always the case, comparing all the masks between forte and piano transformation of the mask $r(m) = \|m-1\|_p^p$ for $p = 1, 2$. All the masks with $\lambda = 0.1$ show results, where the reconstruction of the flute through transformation (cf. Tables 3 and 4) doesn't seem obvious. But the fundamental tone of each mask is accented, because the intensity value is close to 100 compared to the colorbar. This means that the fundamental frequency will be adopted from the violin, whereas the other overtones have intensity values near 0 which means modification of the input sound. Therefore the overtones of the violin have to be modified and prepared, such that they fit in the spectrogram of the flute.
This shows that the commonality of the violin and the flute are in general the fundamental tone and that was what was demanded with "sufficient" similarity. The rough sound and the richness of overtones have to be modified and be adapted for the transformation into the flute.

# Bibliography


[1] Vienna symphonic library: Vienna super package, 2014.

[2] P. Balazs. Basic definition and properties of Bessel Multipliers. *Journal of Mathematical Analysis and Application*, (325):571–585, 2007.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-threshold algorithm for linear inverse problems. *SIAM J.Imaging Sciences*, 2(1):183–202, 2009.

[4] A. Beck and M. Teboulle. *Gradient-Based Algorithms with Applications to Signal Recovery Problems*. Cambridge University Press, 2010.

[5] B. Boashash, editor. *Time Frequency Signal Analysis and Processing. A Comprehensive Reference*. Elsevier Ltd., 2003.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] O. Christensen. *An Introduction to Frames and Riesz Bases*. Birkenhäuser Boston a.o., 2003.

[8] P. Combettes and J. Pesquet. Proximal splitting methods in signal processing. Springer, 2011.

[9] I. Daubechies, M. Defrise, and C. De Mol. An iteratice thresholding algorithm for linear inverse prblems with a sparsity constraint. *Comm. Pure Apl. Math.*, 57(11):1413–1457, 2004.

[10] P. Depalle, R. Kronland-Martinet, and B. Torrésani. Time-frequency multipliers for sound synthesis. *Proceedings SPIE, Wavlets XII*, 6701, 2007.

[11] M. Dörfler and E. Matusiak. Sparse Gabor multiplier estimation for identification of sound objects in texture sound. *Lecture Notes in Computer Science*, 8905:443 – 462, 2014.

[12] G. Evangelista, M. Dörfler, and E. Matusiak. Arbitrary phase vocoders by means of warping. *Musica e Tecnologia*, 7:91 – 118, 2013.

[13] H. G. Feichtinger and T. Strohmer, editors. *Gabor Analysis and Algorithms: Theory and Application*. Birkenhäuser Boston a.o., 1998.

[14] H. G. Feichtinger and G. Zimmermann. A Banach space of test functions for Gabor analysis. In H. G. Feichtinger and T. Strohmer, editors, *Gabor Analysis and Algorithms: Theory and Application*. Birkenhäuser Boston a.o., 1998.

[15] K. Gröchenig. Aspects of Gabor analysis on locally compact abelian groups. Birkenhäuser Boston a.o., 1998.

[16] K. Gröchenig. *Foundations of time-frequency analysis*. Birkenhäuser Boston a.o., 2001.

[17] M. Kowalski, E. Vincent, and R. Gribonval. Beyond the narrowband approximation: Wideband convex methods for under-determined reverberant audio source seperation. *IEEE Transaction on audio, speech and language processing*, 18(7):1818–1829, 2010.

[18] A. Olivero. *Les Multiplicateurs Temps-Fréquence. Application À l'Analyse et la Synthése de Signaux Sonores et Musicaux*. PhD thesis, University of AIX-Marseille, 2012.

[19] A. Olivero, B. Torrésani, and R. Kronland-Martinet. A class of algorithms for time-frequency multiplier estimation. *IEEE Transactions on Audio, Speech and Language Processing*, 21(8):1550–1559, 2013.

[20] A. Olivero, B. Torréssani, and R. Kronland-Martinet. A new method for Gabor multipliers estimation: Application to sound morphing. *Proceedings of EUSIPCO, Aalborg*, 2010.

[21] Zdeněk Průša, Peter L. Søndergaard, Nicki Holighaus, Christoph Wiesmeyr, and Peter Balazs. The Large Time-Frequency Analysis Toolbox 2.0. In Mitsuko Aramaki, Olivier Derrien, Richard Kronland-Martinet, and Sølvi Ystad, editors, *Sound, Music, and Motion*, Lecture Notes in Computer Science, pages 419–442. Springer International Publishing, 2014.

[22] Peter L. Søndergaard, Bruno Torrésani, and Peter Balazs. The Linear Time Frequency Analysis Toolbox. *International Journal of Wavelets, Multiresolution Analysis and Information Processing*, 10(4), 2012.

# Supplements

## Deutschsprachige Zusammenfassung

Wie der Titel dieser Arbeit verrät, beschäftigt sie sich mit der Transformation von akustischen Signalen durch *Gabor multipliers*, auf deutsch Gabor Multiplikatoren.
Im ersten Kapitel wird mit dem Wunsch von gleichzeitiger Darstellung von Zeit und Frequenz die kontinuierliche Kurzzeit-Fourier Analyse eingeführt und auf den Bereich der Gabor Analyse erweitert. Somit werden *Gabor frames* und wichtige zugehörige Eigenschaften erklärt. Im nächsten Kapitel wird eine solche Transformation allgemein durch lineare zeitabhängige Filter eingeführt. Dabei wird ein besonderes Augenmerk auf Gabor Multiplikatoren als solche Filter gelegt.
Durch die Suche eines bestgeeigneten Gabor Multiplikators entsteht ein Inverses Problem, das es zu lösen gilt.
In Kapitel 4 versucht man das entstandene Minimierungsproblem durch Regularisierung zu lösen. Es werden verschiedene Regularisierungsterme vorgestellt und deren Eigenschaften erklärt. Im Spezialfall, dem Diagonalfall, kann man so eine explizite Lösung berechnen, die in diesem Kapitel hergeleitet wird. Im darauffolgenden Kapitel werden diese Lösungen anhand eines Beispiels diskutiert und ihre Vorteile erörtert.
Das 6. Kapitel führt, durch die Einführung einer Gradienten Methode, zur Lösung des Inversen Problems für den allgemeinen Fall. Dazu wird die Theorie von *Proximal Gradient Methods* erarbeitet und auch die zugehörigen Beschleunigungen und Verbesserungen solcher Algorithmen vorgestellt. Zugehörig zu diesem Kapitel ist der im darauffolgenden Kapitel entwickelte MATLAB Code. Dieser fasst die besprochene Theorie in ein Programm zusammen. Nach der Erarbeitung dieses Codes endet die Arbeit mit einem anwendungsorientierten Teil. Es werden Beispiele, die durch den eigenen MATLAB Code erarbeitet wurden, diskutiert. Dabei werden verschiedene Gabor Multiplikatoren in verschiedenen Hinsichten untersucht. Zum einen auf die Existenz eines inversen Gabor Multiplikators, wobei man die Transferfunktion von Start- in Zielsignal und umgekehrt betrachtet. Außerdem wird auch noch ein Experiment mit Blick auf das Verhalten des Multiplikators bei unterschiedlichen Dynamiken erarbeitet.

# Roswitha Sibylle Bammer

*Curriculum Vitae*

## Personal Data

| | |
|---|---|
| Name | **Roswitha Sibylle Bammer**. |
| Date of birth | **May $5^{th}$, 1991**. |
| Place of birth | **Kirchdorf an der Krems**, *Austria*. |
| Nationality | **Austria**. |

## Education

| | |
|---|---|
| 1997–2001 | **Primary School**, *Scharnstein*. |
| 2001–2009 | **Secondary School**, *Gymnasium und ORG des Schulvereins der Kreuzschwestern Gmunden*, Graduation with distinction. |
| 2009–2012 | **Bachelor Mathematics**, *University of Vienna*. |
| since 2012 | **Master Mathematics**, *University of Vienna*. |
| since 2013 | **Bachelor Musicology**, *University of Vienna*. |
| Oct. 2014 | **Doctoral school for computational harmonic analysis**, *CIRM (Marseille)*. |

## Scholarship

| | |
|---|---|
| 2010, 2012–2014 | **Academic Excellence Scholarship**, *University of Vienna*. |

## Languages

| | |
|---|---|
| German | **native** |
| English | **fluent** |
| French | **advanced** |