Real Time Spectrogram Inversion

Diploma Thesis

Clara Hollomey

Institute for Electronic Music and Acoustics

University of Music and dramatic Arts Graz

Supervisor: DI Dr. rer. nat. Franz Zotter

Assessor: O. Univ. Prof. Mag. art. DI Dr. techn. Robert Höldrich

Graz, March 2014

Abstract

The Short Time Fourier Transform (STFT) is not only an important tool for analyzing non-stationary audio signals. As a linear, computational efficient and invertible transformation, it also provides the framework for many other algorithms. In various applications it would be favorable to employ algorithms that are only given the magnitude of the STFT. However, under absence of phase information we cannot invert the spectrogram back to a time signal. Assuming zero phase would leave us with a severely corrupted signal. Thus, magnitude-only spectrogram inversion requires to retrieve the missing phase information.

Until now, phase retrieval is most often based on the algorithm proposed by Griffin and Lim in 1984. The algorithm only works offline and is not suitable for a continuous stream of audio signals.

Primal signal retrieval is meant to provide a solution for multichannel applications requiring to gather spectral signal content from several distributed microphones of unclear phase inter-relations. Negative interference might destroy spectral information when just summing up the microphone signals. It can be entirely circumvented by superimposing magnitude-only spectrograms, yielding the spectrogram magnitude of the primal signal.

What is the spectrogram phase of the primal signal and how can we get it in real time?

Kurzfassung

Abgesehen von seiner wichtigen Funktion als Analysewerkzeug für den Verlauf von Audiosignalen, dient das Betragsspektrogramm auch als Grundlage für viele andere Algorithmen zur Amplitudenmodifikation im Frequenzbereich, für die es vorteilhaft wäre, eine klangtreue Rücktransformation in den Zeitbereich zu erhalten. Klangtreue ist aber ganz ohne Phaseninformation oder mit nur schlechter Phaseninformation nicht erreichbar. Einfache Annahmen über die Phase, wie z.B. Nullphasigkeit in jedem Transformationsfenster, würden das Signal hörbar verändern. Gelungene Spektrogramm-Inversion erfordert die Gewinnung konsistenter Phaseninformation.

Bisher basieren alle Phasengewinnungsalgorithmen auf jenem von Griffin und Lim aus dem Jahr 1984. Dieser Algorithmus arbeitet offline und ist für einen kontinuierlichen Audio-Stream nicht geeignet.

Die Ursignalgewinnung soll es ermöglichen, viele verteilte Mikrofone unklarer Phasenverhältnisse in einem Spektrogramm zusammenzufassen. Durch einfaches Aufsummieren würde man destruktive Interferenzen erzeugen und somit das Ursignal verfälschen. Das kann allerdings vermieden werden, wenn man den Absolutbetrag der einzelnen Mikrofonspektrogramme addiert.

Was ist nun die Phase des Ursignals und wie erhält man sie in Echtzeit?

Statutory Decleration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

_____ _____

date signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

_____ _____

Ort und Datum Unterschrift

Danksagung

Ich bedanke mich bei meinem Betreuer Franz Zotter für seine Unterstützung und den Einführungskurs in strukturierter Programmierung. Danke Christian Schörkhuber für das mentale Aufbauen und deine schnelle Hilfe, wann immer ich sie gebraucht habe.
Generell Danke an das IEM und alle seine Mitarbeiter, es ist nicht nur unglaublich, wie viel ich hier gelernt habe, sondern vor allem auch was. Besonders bedanken möchte ich mich bei Gerhard Nierhaus für die mentalistische Unterstützung und die Gespräche, die mich oft (und dann meist mehr als nur) ein Stückchen weiter gebracht haben. Danke auch an Daniel Mayer für unzählige gemeinsame Konzertbesuche und das Teilen deiner unvoreingenommenen klaren Sicht auf Musik und das Leben drumherum. Danke IOhannes Zmölnig dafür, dass du mich immer wieder bei Laune gehalten hast. Mit der Danksagung an Thomas Musil könnte ich eine eigene Arbeit füllen und darum fange ich hier gar nicht erst an.

Großer Dank geht an meine Eltern und meinen Bruder Marc, die immer für mich da waren und mir klar gemacht haben, dass es wurscht ist, wenn die Dinge genau so schlimm sind, wie sie Anfangs aussehen, wenn man nur irgendwie weitermacht. Meinen Großeltern danke ich für ihren Glauben an mich und ihre bedingungslose Präsenz in meinem Leben.

Jami - mein Lebensmensch.
Danke an Nadine und Rudi für geteilte Freude, Panik, Verzweiflung und Angst. Außerdem bedanke ich mich bei meinen Kolleginnen vom Institut für Elektronik, besonders Reinhard und Sandra für die unterhaltsame Gestaltung der Wartezeit auf "Katzenmann Rising". Danke Oli und Teresa, vor Allem für die Pizza, aber auch dafür, dass ihr immer da seid wenn ich euch brauche und Michi, Stefan sowie allen Leuten von der IAESTE für Zusammenarbeit, Spass, Action, Erfolge, Niederlagen und Freundschaft über viele Jahre - besonders euer Alumni-Service ist super. Und danke Sabrina fürs Erinnern, dass es auch noch etwas anderes im Leben gibt.
Robert und Susi, die unzähligen Gespräche, Kaffees und Zigaretten mit euch zu jeder erdenklichen Tages- und Nachtzeit haben die Inffeldgasse erst so richtig zu meinem Zuhause gemacht.
Das Fertigstellen dieser Arbeit wäre wohl ohne das ausgezeichnete Essen in der "Geidorf-Stub'n" schwer möglich gewesen (Öffnungszeiten: Montag-Samstag, ab 17:00), danke Kathi!

Danke Denise, besser wirds nicht. Carmen, ich weiß nicht, was ich schreiben soll. Der Unterschied ist jedenfalls eklatant. Danke Magdalena, dass ich von dir noch immer so viel lernen kann. Den rudimentären Rest, der über die Jahre von meiner Allgemeinbildung erhalten geblieben ist schreibe ich voll und ganz deinem Einwirken zu. Danke David und die Waupies, wenn ich schon erwachsener werden musste, dann wenigstens mit euch - es war eine großartige Zeit!

Contents

1 Introduction **7**

2 The Spectrogram **11**

2.1 Windows 11

2.2 Hopsize 15

2.3 DFT-Length 16

2.4 The Spectrogram 16

2.5 Notation in Vector Form 17

3 Spectrogram Inversion **22**

3.1 Relevant Properties of the STFT 22

3.1.1 Spectrogram Consistency 23

3.1.2 Unicity of Representation 25

3.1.3 DFT Magnitude-Phase Relations 25

3.2 Spectrogram Inversion 26

3.3 The Griffin and Lim Algorithm 27

4 Real Time Spectrogram Inversion **31**

4.1 Real-Time Iterative Spectrogram Inversion (RTISI) 31

4.2 RTISI Extended 33

4.3 Fundamental Issues 34

5 Implementation **37**

5.1 Basic Considerations 37

5.2 Experiments, Results and first Evaluation 38

5.3 The Real-Time Constraint 39

5.4 RTISI Restructured 40

5.5 Considerations on Realtime Capability 45

6 Outlook and Conclusion **48**

A Characteristics of Audio Signals **51**

B Fast Fourier Transform **53**

C TSM: Operating directly on the STFT **56**

C.1 Time Scale Modification 56

C.2 Phase Vocoder 57

C.3 Phase-locked Vocoder 59

C.4 Overlapp-Add Method 60

1 Introduction

For audio signal analysis it is often convenient and even more often necessary to display a signals evolution in the time and frequency domain at the same time. For discrete-time signals, this can be done by computing the discrete Fourier transform for each time-step n and realigning the thus obtained sequence in time. One usual side-constraint when doing so is preserving equivalence to the standard Fourier transform (and thus invertibility to the time-domain), a goal that can be easily achieved when certain conditions on the STFT's structure are met (for details, see Chapter 2).

The Spectrogram

From an analysis point of view, the STFT literally opens up a new dimension of signal representation: while time and frequency are displayed on the coordinate systems abscissa and ordinate, the z-axis is dedicated to the signals amplitude, usually represented by the intensity or color of each point. Taking the STFTs squared absolute value in dB corresponds to frequency intensity and displays a signal's energy distribution per Hertz. This kind of signal representation is often referred to as the classical (positive and real-valued) Spectrogram.

Due to its computational simplicity, the STFT has become one of the most important analysis tools of nowadays signal processing.

Spectrogram Inversion

Unfortunately, in most cases one will not only want to analyze the audio signal but rather desires to change something about it. Signal modification in the Fourier domain is often simpler and in addition computationally more efficient than in the time-domain. Therefore the standard workflow in audio signal processing usually consists of the following steps:

– the time-domain signal undergoes a Fourier analysis,
– its magnitude is processed,
– it is retransformed to the time-domain and
– played back.

Since audio signals are often non-stationary, the computationally simpler discrete Fourier transform is extended to the STFT. In order to gain a playback signal after the STFT and the processing step, the signal needs to be re-transformed to the time-domain. The audio signal processing workflow using STFT is shown in figure 28.

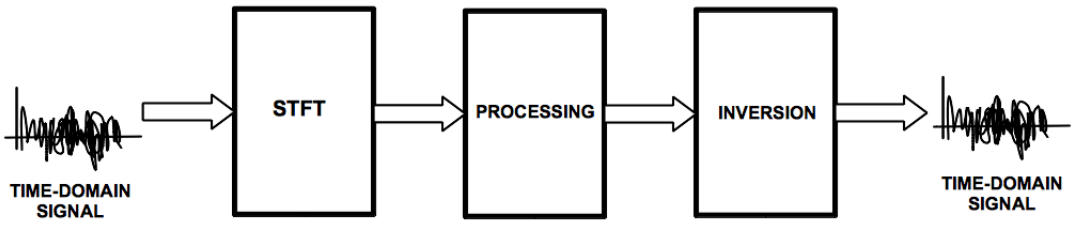


Figure 1: Audio signal processing workflow using STFT

Typical processing steps include operations like time scale modification, pitch shifting, equalizing or denoising. Within this context, one typically aims at preserving as much of the input signal as possible while achieving some improvement on certain features.

However, conventional inversion of such a modified STFT is not guaranteed to yield an audio signal as an output - let alone a signal related in any way to the primary input. Therefore, an estimate for the respective STFTs inversion needs to be found and applied (Chapter 3).

As far as spectrograms are concerned, in order to perform the absolute value computation in the complex domain which is necessary to obtain them, all explicit phase information is discarded. Still, the spectrogram is assumed to contain all or at least most of the information originally present in the signal. However, the question where exactly which information is contained has not been solved yet. Due to this total lack of phase information, spectrogram inversion probably is the worst-case scenario of STFT-inversion.

Reconstructing a phase-shifted signal from its spectrogram by inverse Fourier transform leads to the result shown in figure 2.

This reconstruction may not be called perfect. In fact, an arbitrary signal cannot be reconstructed by means of an inverse Fourier transform when only its magnitude is known. While nowadays storing the phase information presents a minor challenge since most computer systems enable the user to keep all computational results, in some cases being able to operate directly on the spectrogram is still required.

Real-Time Spectrogram Inversion

The motivational example for this thesis is primal signal retrieval (see figure 3). IEM owns a 64-channel spherical microphone array capable of recording the directional sound radiation of any subject placed within it. Playback of a single-channel sound representing the entire set of all 64 channels in terms of an audio signal would be useful to have. If accomplished in a way without destructive interference and in real-time the design of

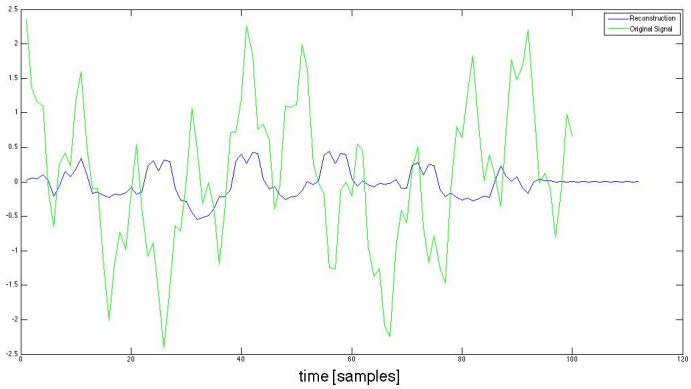


Figure 2: Reconstruction of a signal from its Spectrogram assuming zero-phase (green...input signal, blue...its poor reconstruction)

adaptive filters representing the directivity would largely be simplified. This approach is hoped to improve the audio quality of known SIMO/MIMO identifications approaches.

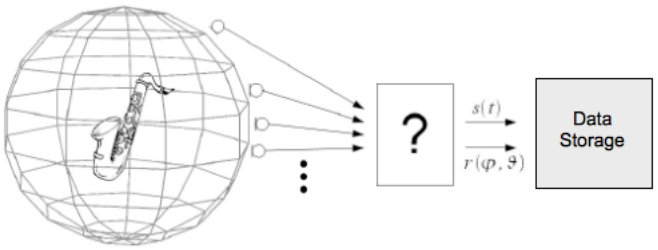
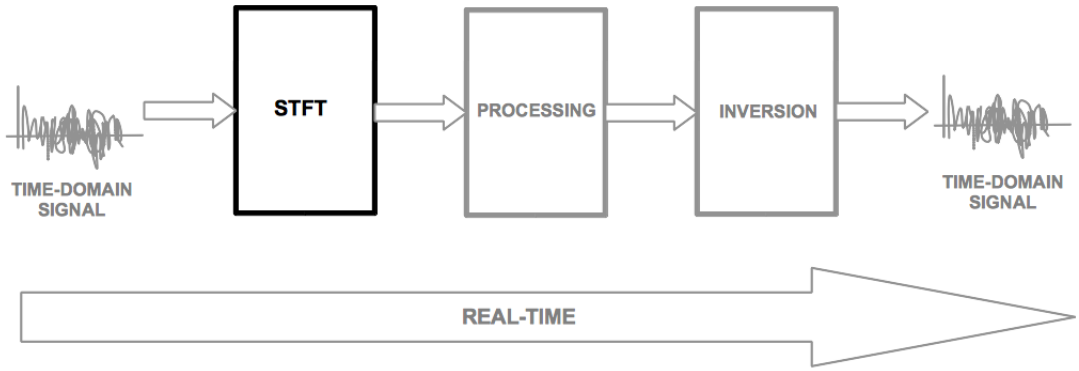


Figure 3: The Concept of Primal Signal Retrieval (from [Sue11])

In order to do so without the risk of interferences destroying or corrupting the audio-data, the magnitudes of all channels are averaged, yielding - a spectrogram.

Luckily, Griffin and Lim developed an algorithm that accomplishes signal reconstruction from STFT-magnitude [Sue11]. However, Griffin and Lim's algorithm opposes real time computation and therefore the question remains open: What is the spectrogram phase of the primal signal and how can we get it in real time?

The Spectrogram



- How to compute the STFT and the Spectrogram?
- Which constraints must be met in order to gain an STFT preserving all information on the time signal?
- Can all this be simplified with regards to applications in audio signal processing?

2 The Spectrogram

Since in this work all signals are assumed being sampled and bandlimited, the STFT can be defined by means of the DFT:

$$X(m, \omega_k) = \sum_{n=0}^{N-1} x[n + mR]w[n]e^{-j\omega_k n} \tag{1}$$

with $\omega_k = \frac{2\pi k}{K}, k \in \mathbb{Z}$. Starting from a time sequence $x[n]$, the STFT can be derived using the following steps:

1. split $x[n]$ into M overlapping sections of length N and let the hopsize R denote the shift between two neighbouring sections

$$\underline{X_m} = \left[\; x_0^T, x_1^T, ..., x_{M-1}^T \; \right]^T \tag{2}$$

2. weight each section $\underline{x_m}$ by a window-function $w[n]$
3. optionally append this windowed x_m by $K - N$ zeros
4. calculate the K-point discrete Fourier transform

So in summary, the STFT depends on the following parameters:

– the window function $w[n]$
– the hopsize R
– the length K of the Fourier transform

Basically, this chapter aims to give more detailed insights on how this parameters need to be chosen in order to obtain an STFT that preserves all parameters necessary for perfect reconstruction when processing audio signals. The constraints derived in this section allow not only the derivation of a more compact vector-formulation of the STFT but will also be exploited later in this work in order to obtain a realtime capable algorithm for spectrogram inversion.

2.1 Windows

In most cases, the period of the signal under investigation will not match the DFTs. In order to smooth discontinuities that arise due to the DFTs circular nature, a window function is applied that then undergoes the Fourier transform the same way as the input signal does.

Audio signals are real signals, yielding a symmetric and therefore redundant Fourier transform. So apart from being non-zero at all coefficients for simplicity, the window-function should be real-valued.

There are many different kinds of windows to choose from and no general assumption can be made on their usefulness as this highly depends on the application. Within a focus on audio signals, it seems impossible to determine a single best suited window. Choosing

the window function involves always a trade-off between rather resolving equal-strength components with similar frequency values as opposed to resolving those with diverging characteristics. Especially some cosine-type windows, namely the Hanning and the Hamming window are situated in the middle of this trade-off and therefore are extensively applied for narrow-band applications (e.g. telephone channels).

Since those cosinge-type windows are widely used, they are also likely to suit within the context of spectrogram inversion and will therefore be dealt with in more depth. For further information on windows for audio signal processing it is referred to [Nut81,Smi14, OS99]. Starting from the basic rectangular window, this section discusses cosine-type windows, their impact on signal representation in the time-frequency domain and the resulting application areas for audio signal processing.

The Rectangular Window

Defined as '1' (or sometimes, constant) from $-\frac{M-1}{2}$ to $\frac{M-1}{2}$ and zero elsewhere, the rectangular window is the simplest case of a window. Its DFT can be derived as:

$$W(\omega_k) = \sum_{n=0}^{N-1} w[n] e^{-j\omega_k n} \tag{3}$$

$$= N \mathrm{sinc_c}(\omega_k), \tag{4}$$

with $\omega_k = \frac{2\pi k}{K}, k \in \mathbb{Z}$ and N the window-length

The Fourier transform of the rectangular window yields an output $sinc_c$ that strongly ressembles a sinc-function but due to the sampling in the time-domain which is necessary for the DFT, aliasing in the frequency domain cannot be prevented. However, assuming the sampling frequency in the time-domain going to infinity, the distorted $\mathrm{sinc_c}(\omega)$ reaches $\mathrm{sinc}(\omega)$.

Investigation of this periodic sinc-function's magnitude- and phase-spectrum shows that not all of the window's energy is concentrated in the center, but there are additional non-zero areas. Being solely dependent on the sharpness of the windows transition to zero-values, the side-lobes are likely to corrupt time-frequency relations since multiple sinusoids at various frequencies can become indistinguishable in the spectral domain.

As far as the centered *main lobe* is concerned, its width

$$d_{ML} = \frac{4\pi}{M} \tag{5}$$

depends on the window-length M only.

Thus, choosing a shorter window generally leads to a higher ambitus in frequency resolution, while on the other hand, a shorter window might not be capable to catch the full

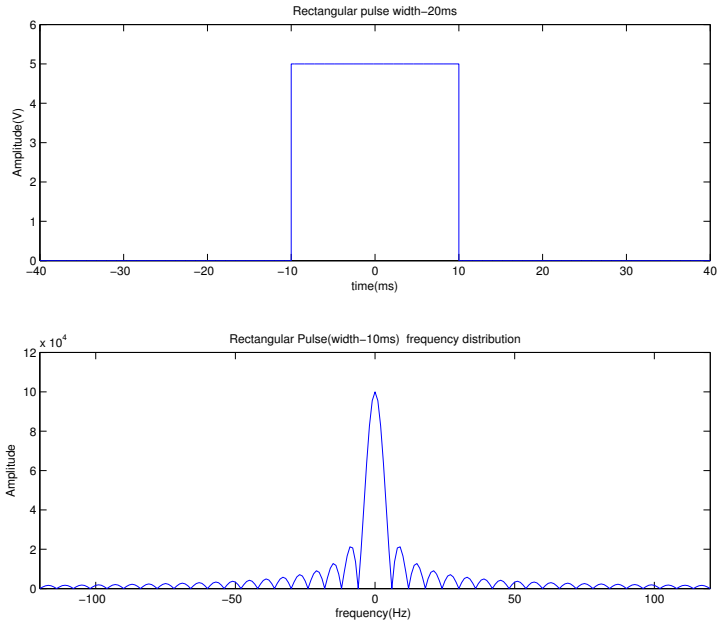


Figure 4: Time and Frequency Distribution of a Rectangular Window

waveform of a lower frequency component which might corrupt the frequency information present within the window's borders.

Spectral leakage is often referred to as a window's dynamic range [Smi14]. While the rectangular window is very capable of resolving sinusoids of equal strength, when amplitudes differ, it is a poor choice since the different side-lobes make it impossible to determine the respective sinusoids origin.

In figure 4, a rectangular window is displayed in time and frequency domain.

Cosine-Type Windows

In order to reduce the side lobe level, the abrupt transition between zero and one of the rectangular window needs to be smoothed. The Hamming window family can be produced by multiplication of the rectangular window with one period of a cosine or, equivalently, superposition of three sinc-functions in the frequency domain.

Coming from this approach, the Hamming window can be expressed as three shifted rectangular windows W:

$$W_{ct}(\omega) = \alpha W(\omega) + \beta W(\omega - \omega_S) + \beta W(\omega + \omega_S) \tag{6}$$

with ω_S being the frequency shift. The shift invariance of the DFT then leads to the

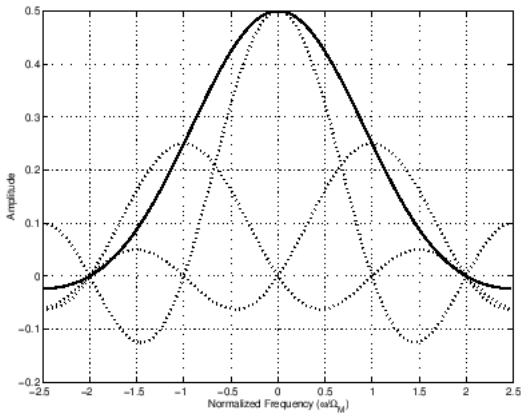


Figure 5: Construction of a hamming window via rectangular windows (from [Smi14])

cosine-type window families standard generating formula:

$$w_{ct} = \alpha w[n] + \beta w[n]e^{-j\omega_S n} + \beta w[n]e^{j\omega_S n} \tag{7}$$

$$= w[n](\alpha + 2\beta\cos(\frac{2\pi n}{M})). \tag{8}$$

Some combinations of α and β have become known under specific names.

$$w_{ct} = w[n](\frac{1}{2} + \frac{1}{2}\cos(\frac{2\pi n}{M})) = w[n]\cos^2(\frac{\pi n}{M}) \tag{9}$$

yields the Hanning window. The magnitude of its side-lobes decreases about three times as fast as that of the rectangular window. Combined with the very smooth transition from one to zero it is a suitable window for most audio-processing purposes.

The Hamming window however cancels the first side-lobe completely as α is chosen as $\alpha = 1 - \beta$. Even though this affects the continuity of the transition from the window's ends to zero, the hamming window often is the window of choice for low-quality audio processing as its side-lobes ("ripples") are all of equal magnitude and lie right below the quantization noise floor of an 8-bit codec (approximately 40 dB lower than the main lobe's peak).

When instead of three shifted rectangular windows an arbitrary number is used, the resulting window type is called Blackman-Harris window. Thus, the cosine-type window family can be generalized (with $w[n]/W$ the rectangular window) via:

$$w_{ct}[n] = w[n]\sum_{l=0}^{L-1}\alpha_l\cos(l\frac{2\pi}{M}n) \tag{10}$$

or, equivalently in the frequency domain:

$$W_{ct}(\omega) = \sum_{k=-(L-1)}^{L-1} \alpha_k W(\omega k \frac{2\pi}{M}). \tag{11}$$

This generalization is usually referred to as the Blackman-Harris window-type. Each added rectangular window provides another degree of freedom for either side-lobe minimization providing a possible improvement to windows in Hamming style, or optimization of the magnitude decrease - the roll-off factor, as is the goal in the Hanning window case. This goes at the expense of frequency resolution, as the window's main lobe widens with each rectangular window added.
Generally spoken, the more elaborated windows tend to support a larger spectral dynamic range -a property that is sometimes necessary in audio processing in order to enforce especially higher frequency components.

In summary, the used window-function is real-valued for simplicity (in order to preserve the symmetric output spectrum of the DFT inherent to real signals) , of the same length as x_m and without zero coefficients. Hamming and Hanning windows exhibit a relatively fine frequency resolution and an agreeable dynamic range in the frequency domain and are therefore extensively used in audio signal processing.

2.2 Hopsize

For deriving an AIP-STFT, one single constraint on hopsize and window function must be met:

If for the window function

$$\sum_{j=0}^{\infty} w[n - jR] = 1, \forall n \in \mathbb{Z}, \tag{12}$$

then the sum of all STFT-frames corresponds to the DFT of the whole signal and the window is said to have the *constant overlap-add* property. A signals retransformation into the time-domain while not fulfilling this requirement results in a different weighting of the output signals' amplitude. Although for some applications, especially when mere signal analysis is required, the constant overlap-add constraint may be relaxed, such corruption of the signal leads to perceptible artifacts (amplitude modulation) and thus is usually undesired in audio signal processing.

The exact amount of overlap in samples is determined by the used window-function.

It can be concluded that for windows other than the rectangular window which exhibits far too many drawbacks for practical usage, a hopsize $0 < R \leq N$ is required and therefore, an overlap must exist. Also, for simplicity, the overlap is assumed to be constant.

2.3 DFT-Length

In practice however, the DFT is replaced by the computational more efficient FFT, usually applying the Cooley-Tukey algorithm (for more information, see Appendix) which operates in its optimum at a signal length $K = 2^p, p \in \mathbb{Z}$. Therefore, most STFT-implementations either perform internal zero-padding up to the next power of two or require the input signals $\underline{x_m}$ being of length 2^p in the first place.
In addition, the DFT exhibits the already mentioned redundancy for real signals permitting to keep only the first $\frac{N}{2} + 1$ values for input signals of even length.

So for further considerations the DFT-length can be assumed to exactly match the length of the input section $\underline{x_m}$ with $N = K = 2^k$, thus preventing the need for zero-padding.

2.4 The Spectrogram

Invented at Bell Laboratories during World War II [Smi14, KC92], the spectrogram has become the most important tool for displaying signals in a mixed time- and frequency-domain. Its entries refer to the (pointwise) absolute value of the STFT X according to

$$P_{mk} = \sqrt{Re(X_{mk})^2 + Im(X_{mk})^2}$$

for $k = 1, 2, ..., K$ and $m = 1, 2, ..., M$, with P the spectrogram and Re/Im its real and imaginary components. It can be seen as a usually dB-scaled intensity plot of the STFTs' magnitude and is defined by the parameters:

Parameter	Effect
window-type	side-lobe suppression
window-length	frequency resolution
hopsize R	sampling factor time-domain
FFT-length N	sampling factor frequency domain

Table 1: Summary of Spectrogram Parameters and their effects

Since the window type controls the side-lobe suppression, it is also responsible for the often-mentioned time-frequency smearing. While the hopsize R determines the down-sampling in the time-domain, its minimum is a sliding FFT with $R = 1$. In summary for the choice of R, on the one hand the perfect reconstruction criterion with regard to the used window type has to be fulfilled, while on the other hand, in order to avoid redundancy one will usually try to make it as large as possible.

2.5 Notation in Vector Form

In [Yan08], a framework for general-purpose STFT and ISTFT vectorization is proposed. With regards to the conditions and constraints derived in this chapter, this framework is compactly represented here to fit the application of spectrogram inversion.

Describing the transformation itself is straight forward using the DFT-matrix:

$$\boldsymbol{F} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & ... & ... & 1 \\ 1 & \phi & \phi^2 & \phi^3 & ... & \phi^{N-1} \\ 1 & \phi^2 & \phi^4 & \phi^6 & ... & \phi^{2(N-1)} \\ 1 & \phi^3 & \phi^6 & \phi^9 & ... & \phi^{3(N-1)} \\ : & : & : & : & & : \\ 1 & \phi^{N-1} & \phi^{2(N-1)} & \phi^{3(N-1)} & ... & \phi^{(N-1)(N-1)} \end{bmatrix} \tag{13}$$

with $\phi = e^{-\frac{j2\pi}{N}}$.

In a next step, the window function can be written as a diagonal matrix with the window coefficients w_n. Clearly, both matrices are square and for analysis of a signal $x[n]$ with length N, of the dimension $[N \times N]$.

$$\boldsymbol{W} = \begin{bmatrix} w_1 & 0 & 0 & ... & ... & 0 \\ 0 & w_2 & 0 & 0 & ... & 0 \\ 0 & 0 & w_3 & 0 & ... & 0 \\ 0 & 0 & 0 & w_4 & ... & 0 \\ : & : & : & : & & : \\ 0 & 0 & 0 & 0 & ... & w_N \end{bmatrix} \tag{14}$$

The remaining question now is how to denote the overlap-procedure. The overlap matrix $\mathbf{O}$ consists from identity matrices shifted against each other according to the respective hopsize:

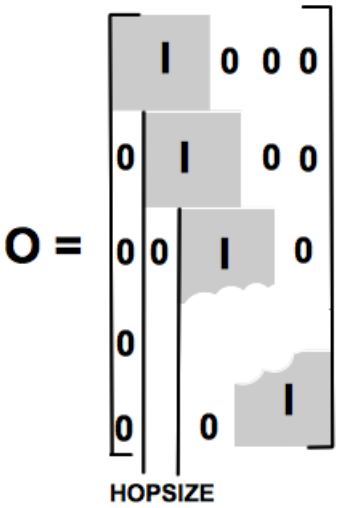


Figure 6: The overlap matrix

Assuming the input signal $x[n]$ to be split into M sections of length N (and therefore being of length $(M-1)R+N$) gives an overlap matrix $\boldsymbol{O}$ (see figure 6) of dimensionality $[MN \times (M-1)R+N]$ and covers the whole signal while the Fourier transform matrix $\boldsymbol{F}$ and the window matrix $\boldsymbol{W}$ of dimensionality $N \times N$ cover only one section $\underline{x_m}$. Figure 7 shows the relation between $x[n]$ and its overlapped version.

After splitting $x[n]$, the sections $\underline{x_m}$ needs to be windowed and subsequently undergo a Fourier transform. The M sets of decoupled equations can be summarized within two matrices $\boldsymbol{W}_{Block}$ and $\boldsymbol{F}_{Block}$. The Kronecker tensor product of a matrix and an identity matrix

$$\boldsymbol{C} = \boldsymbol{A} \circ \boldsymbol{B} = (a_{ij} \cdot \boldsymbol{B}) = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \dots & \dots & \dots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix} \tag{15}$$

which need not necessarily be of the same dimension gives a matrix in the desired block structure. Considering the Kronecker tensor product of two matrices $\boldsymbol{I}$ and $\boldsymbol{Q}$ with $\boldsymbol{I}$ the identity matrix gives

$$\boldsymbol{I} \circ \boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 \cdot \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 \cdot \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \cdot \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix} \end{bmatrix}$$

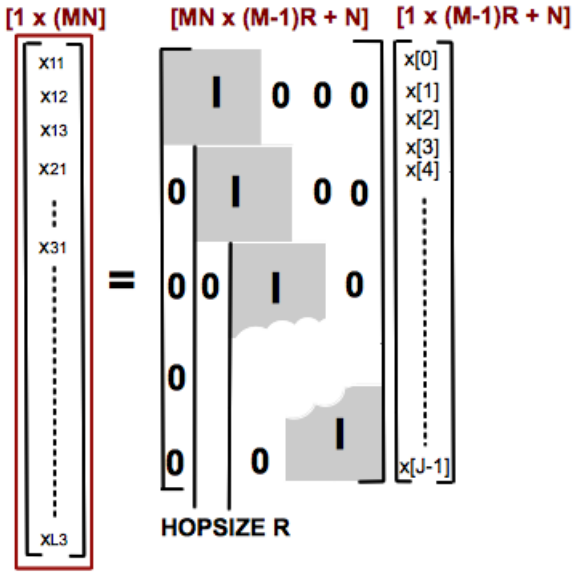


Figure 7: Splitting a signal $x[n]$ into M overlapping sections of length N

and the desired block diagonal structure is obtained. Applying this operation on $\boldsymbol{F}$ and $\boldsymbol{W}$ yields the desired block-diagonal matrix $\boldsymbol{F}_{Block}$ and the diagonal matrix $\boldsymbol{W}_{Block}$, both of dimension $[MN \times MN]$.

The overall STFT is resumed in figure 8. Notably the STFTs output $\underline{X}$ in the frequency domain is obtained as a vector rather than in the usual matrix form. In short:

$$\underline{X} = \boldsymbol{F}_{Block}\boldsymbol{W}_{Block}\boldsymbol{O}\underline{x} \tag{16}$$

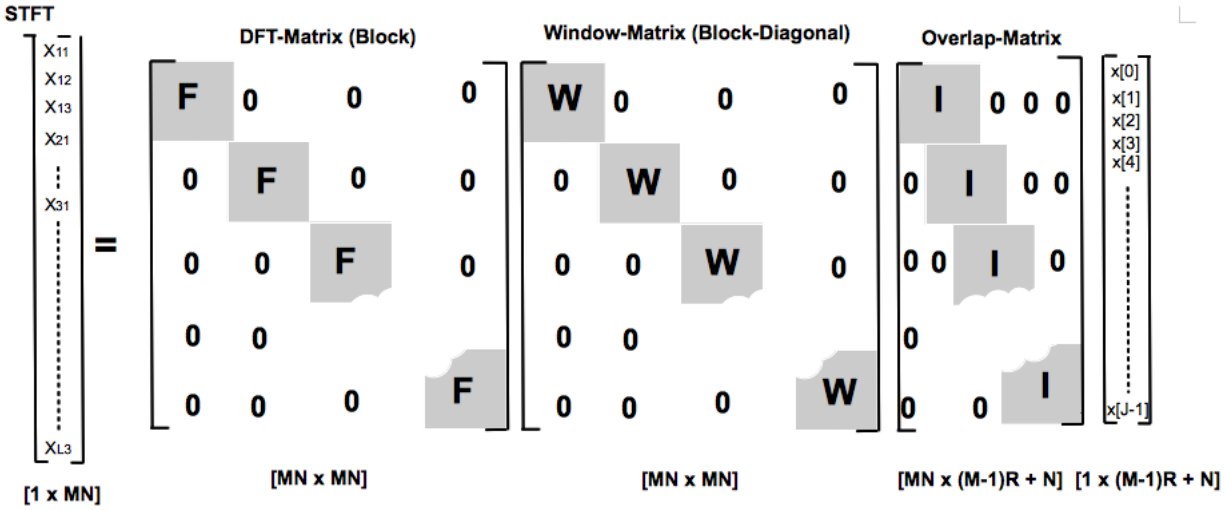
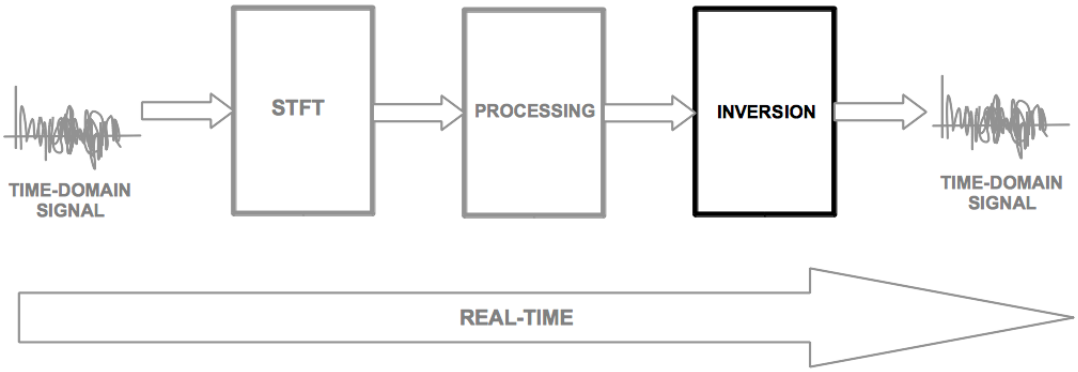


Figure 8: The STFT

Spectrogram Inversion



This chapter deals with STFT inversion in general.
– Which conditions must be met in order for an STFT or Spectrogram to be invertible?
– And what is the role of phase?
In addition, equivalently to chapter 2, a vector notation for inversion of the Short-Time Fourier Transform will be given and the baseline for any subsequent work on spectrogram inversion, the Griffin and Lim algorithm, will be introduced.

3 Spectrogram Inversion

3.1 Relevant Properties of the STFT

When dealing with spectrogram inversion, we need to answer some basic questions concerning the STFT:

- Is there a time-domain signal x for any array of complex numbers? And subsequently: What are the conditions for a set of real non-negative numbers being called a spectrogram?
- Does spectrogram inversion generally lead to a unique time-domain representation of the signal? If not so, which constraints must be met?
- Are there any known magnitude-phase relations that can be applied in order to simplify reconstruction?
- Is the spectrogram phase-variant at all? Does phase really matter?

Apart from the fact that there are many works [Pal03, CSD10, NQL83], suggesting the perceptional importance of phase, the third question can easily be answered by conducting a simple experiment: If spectrograms of real signals were phase-invariant, considering a cosine-signal $s = A\cos(wt + \phi)$ and varying the phase ϕ from 0 to 2π should always give the same resulting spectrogram provided this spectrogram is always computed for the same parameters. A first non-informed visual inspection of the following two spectrograms where N designates the FFT-length and M refers to time reveals that this obviously is not the case:

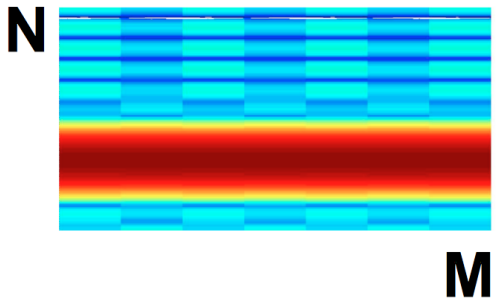


Figure 9: Normalized Spectrogram for $\phi = 0$, with overlap=4, fft-bins=64

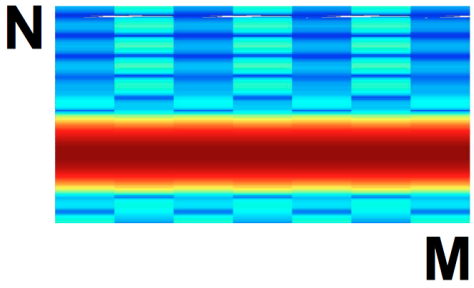


Figure 10: Normalized Spectrogram for $\phi = \pi/4$, with overlap=4, fft-bins=64

Another interesting observation is that the difference introduced by the phase shift is not concentrated on the concerned frequency band but rather affects the whole spectrogram. To describe the differences more precisely, the difference between the spectrograms was computed as $||S| - |S_{\pi * d}||$ with $d = 0 : \pi/4 : 2\pi$. The resulting errors' energy is far from negligible and can be seen in figure 11.

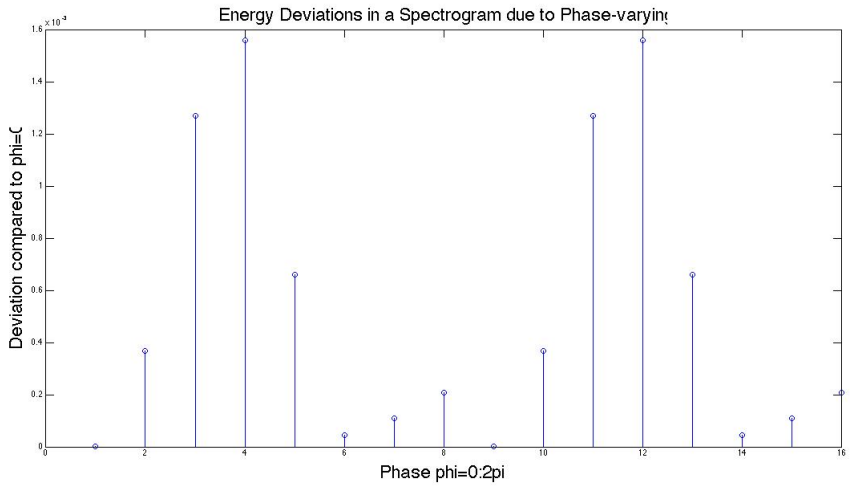


Figure 11: Gathered magnitude of two spectrograms due to phase varying from $\phi = 0$ to $\phi = 2\pi$

Thus, phase does matter and this section addresses the three remaining questions giving a summary on what is known about the mathematical conditions under which a spectrogram inversion is theoretically possible.

3.1.1 Spectrogram Consistency

In the previous chapter, the STFT was defined as

$$X(m, \omega_k) = \sum_{n=0}^{N-1} x[n + mR]w[n]e^{-j\omega_k n}$$

where R is the hopsize of adjacent frames. Its reciprocal, $STFT^{-1}$ will further be defined as $ISTFT$ (Inverse Short-Time Fourier Transform).

Given a set of real non-negative numbers $P \in \mathbb{R}_+^{N\times M}$, this set is a spectrogram if $P = \sqrt{XX^*}$, where $X^*(\omega_k)$ is the complex conjugate of the subset of STFTs $X(\omega_k) = STFT(ISTFT(X(\omega_k)))$. Thus for reconstruction, two equations must be verified:

$$P = \sqrt{X(\omega_k)X^*(\omega_k)} \tag{17}$$

$$C = X(\omega_k) - STFT(ISTFT(X(\omega_k))) \tag{18}$$
$$= X - FWO(FWO)^+X = 0 \tag{19}$$

For many applications the spectrogram P might not be consistent in the above defined sense due to interference from other sources (source separation), corruption by noise

(denoising) or simply bad interpolation algorithms (pitch shifting) [SD11]. In such cases, P can be approximated by minimization of $C(X|\hat{X})$ with regard to some norm (usually the Froebenius norm) in order to find another STFT-subset $\hat{X}(\omega_k)$ verifying both $P = \sqrt{\hat{X}\hat{X}^*}$ and the consistency criterion.

The following graphic (modified from [SD11] and [LKOS10]) illustrates the inter-domain relations for STFT-processing:

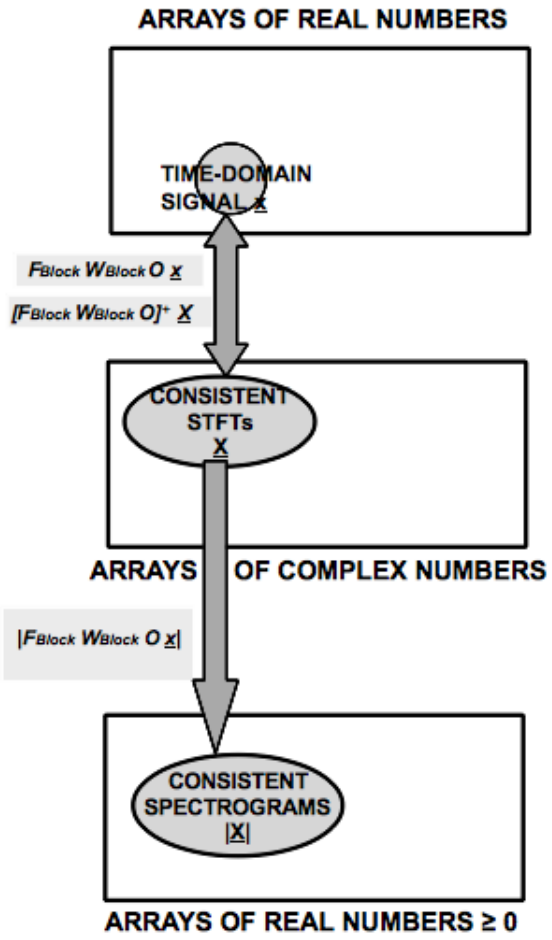


Figure 12: Domains involved for Spectrogram Computation

In summary, only consistent STFTs lead to specific real-valued time-domain signals $\underline{x_i}$ and an array of complex numbers can only be referred to as a consistent STFT $\underline{X}$, if it was computed from a time-domain signal $\underline{x}$ according to

$$\underline{X} = \boldsymbol{F_{Block} W_{Block} O} \underline{x}$$

whereas a consistent spectrogram can only be derived from a consistent STFT via

$$\underline{P} = \sqrt{\underline{X} \underline{X*}}$$

or

$$\underline{P} = |\boldsymbol{F_{Block}} \boldsymbol{W_{Block}} \boldsymbol{O} \underline{x}| \tag{20}$$

3.1.2 Unicity of Representation

The spectrogram corresponds to the absolute value of the $STFT$ (e.g. $|STFT[x]| = |STFT[-x]|$). Therefore, there are always at least two signals fulfilling the consistency criteria on the spectrogram potentially preventing the spectrogram inversion algorithm from convergence. In literature, this phenomenon is called stagnation [FW86].

Extensive research on the conditions under which a unique signal reconstruction is possible has been done by Nawab et al. [NQL83]. According to them, for a hopsize $R > 1$ unique signal reconstruction can be achieved under the following conditions:

1. since each window changes a signals time-frequency relation in a different way, the spectrograms window function $w(n)$ must be known
2. $R \leq N$, so the STFT contains at least the same amount of information as the original signal does
3. the window function is required not to contain any zeros within [0; N-1], otherwise information on the signal might get lost
4. one of the signals' boundaries must be defined (e.g. one-sided signal), which is not of practical importance to audio signals since there would be too many changes in-between the boundaries to predict any phase-related behavior [SD11] just from knowing a starting point
5. the first R samples of the signal must be known (starting from the first non-zero sample)
6. there must not be more than R consecutive zero-samples within the signal

In [SD11], Sturmel et al. invalidated the latter two conditions stating that they resulted from Nawab's goal to successively interpolate the signal and that there are many cases where they are neither necessary nor sufficient.

3.1.3 DFT Magnitude-Phase Relations

Even though the Bode-Theorem uniquely links phase and magnitude under minimum-phase conditions [DH04], no satisfying connection for mixed-phase signals has been provided so far, even though many works exploited magnitude-phase relations trying to reconstruct intelligible signals from one of the two. Hayes [HLO80] already stated in 1980, that:

A sequence which is known to be zero outside the interval $0 \leq n \leq (N-1)$ *is uniquely specified to within a scale factor by (N - 1) distinct samples of its phase spectrum in the interval* $0 < w < \pi$ *if it has a z-transform with no zeros on the unit circle or in conjugate reciprocal pairs.*

This statement has been verified by others (e.g. [AKP07], [YSK84], [OS99]) and thus it can be concluded that under the presence of phase information, especially when the sign is known, a signal can be reconstructed perfectly up to a scale factor. Unluckily, for the case where only magnitude information is available, things are getting more complex due to the ambiguities and stagnation issues already outlined.
For some signal-classes, such as speech and images, being able to regain the correct phase seems to be of less importance, as the frequency spectrum of speech on the one hand, is relatively predictable as compared to music and although additional phase information is likely to increase intelligibility, the aesthetical requirements as far as the reconstructed signals' sound quality is concerned are different. Furthermore, it has been shown [Pal03] that modelling speech as an Auto-Regressive process provides a lot of information on the signals' nature facilitating computation. (for more details on the spectral differences between speech and music see: Appendix). As far as images are concerned, the signal under reconstruction is positive per definition [LKOS10] and thus, phase-sign indeterminacies vanish.
Yegnanarayana et al. [YSK84], explored the importance of group delay as a linking factor from magnitude to phase and derived algorithms for different amounts of previous knowledge on the signal to reconstruct, but for arbitrary (i.e. mixed-phase) signals, according to them, full knowledge of phase and magnitude is required in order to provide a reconstruction.

3.2 Spectrogram Inversion

An intuitively comprehensible formulation for the ISTFT can be easily derived solving the STFTs matrix notation for $\underline{x}$:

$$\begin{aligned} \underline{X} &= \boldsymbol{F_{Block}} \boldsymbol{W_{Block}} \boldsymbol{O} \underline{x} \\ \boldsymbol{O}^T (\boldsymbol{W_{Block}} \boldsymbol{F_{Block}})^{-1} \underline{X} &= \boldsymbol{O}^T (\boldsymbol{W_{Block}} \boldsymbol{F_{Block}})^{-1} (\boldsymbol{F_{Block}} \boldsymbol{W_{Block}}) \boldsymbol{O} \underline{x} \\ \underline{x} &= \boldsymbol{O}^T (\boldsymbol{W_{Block}} \boldsymbol{F_{Block}})^{-1} \underline{X} \end{aligned}$$

Still, as $\underline{x}$ is clearly overdetermined by $\underline{X}$, there is no exact inverse to the STFT operation matrix

$$\boldsymbol{H} = (\boldsymbol{F_{Block}} \boldsymbol{W_{Block}}) \boldsymbol{O}. \tag{21}$$

A very common approach to such problems is the application of the least squares solution in order to derive an estimate:

$$\begin{aligned} \underline{X} &= \boldsymbol{H} \underline{x} \\ \underline{x}_{LS} &= (\boldsymbol{H}^H \boldsymbol{H})^{-1} \boldsymbol{H}^H \underline{X} \\ &= (\boldsymbol{O}^T \boldsymbol{W}_{Block} \boldsymbol{F}_{Block}^H \boldsymbol{F}_{Block} \boldsymbol{W}_{Block} \boldsymbol{O})^{-1} \boldsymbol{O}^T (\boldsymbol{W}_{Block} \boldsymbol{F}_{Block}^H) \underline{X} \end{aligned}$$

and since for the DFT-matrix $\boldsymbol{F}^H = \boldsymbol{F}^{-1}$,

$$\underline{x}_{LS} = (\boldsymbol{O}^T \boldsymbol{W}_{Block} \boldsymbol{W}_{Block} \boldsymbol{O})^{-1} \boldsymbol{O}^T (\boldsymbol{W}_{Block} \boldsymbol{F}_{Block}^{-1}) \underline{X}. \tag{22}$$

This solution corresponds to the expression derived for the ISTFT,

$$\underline{x} = \boldsymbol{O}^T(\boldsymbol{W}_{Block}\boldsymbol{F}_{Block}^{-1})\underline{X}$$

except for the additional term

$$(\boldsymbol{O}^T\boldsymbol{W}_{Block}\boldsymbol{W}_{Block}\boldsymbol{O})^{-1}.$$

Closer investigation of this latter expression (here assuming $N = 2$ for simplicity) yields

$$(\boldsymbol{O}^T\boldsymbol{W}_{Block})(\boldsymbol{W}_{Block}\boldsymbol{O})^{-1} = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_2 & w_1 & 0 \\ 0 & 0 & 0 & w_2 \end{bmatrix} \cdot \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & w_1 & 0 \\ 0 & 0 & w_2 \end{bmatrix} =$$
$$= \begin{bmatrix} w_1 \cdot w_1 & 0 & 0 \\ 0 & w_2 \cdot w_2 + w_1 \cdot w_1 & 0 \\ 0 & 0 & w_2 \cdot w_2 \end{bmatrix}.$$

So the Least Squares solution for $\underline{x}$ can be interpreted as an $ISTFT$ weighted by the overlapping sum of the squared window function.

Based on this least squares solution, in 1984 Griffin and Lim [GL84] presented the first global approach to phase estimation from spectrograms. Intended as an improvement for Time-Scale-Modification, due to its simplicity and perceptually good results it has established itself as the baseline for any subsequent work.

3.3 The Griffin and Lim Algorithm

Generally spoken, the Griffin and Lim algorithm relies on a two-domain constraint: while on the one hand, the given absolute values of the estimated $STFT$ are kept fixed in the spectral domain, phase coherence is enforced in the time-domain. The basic aim is finding a time-domain counterpart for the modified -and therefore probably inconsistent- spectrum $Y(\omega)$ by a least squares approach.

In other words, an update rule that minimizes the Euclidean Distance

$$D[x[n]|Y[m,k]] = \sum_{m=-\infty}^{\infty} \frac{1}{2\pi} \int_{\omega=-\pi}^{\pi} |X[m,k] - Y[m,k]|^2 d\omega \tag{23}$$

between two frequency spectra is proposed according to:

$$x[n] = \frac{\sum_{m=-\infty}^{\infty} w[mR-n]y[mR,n]}{\sum_{m=-\infty}^{\infty} w^2[mR-n]} = ISTFT_m \tag{24}$$

which leads to a slightly modified $ISTFT_m$ quite similar to that derived in the previous section where y[n] explicitly needs to be windowed before the overlap-add procedure and additionally, a squared window normalization term needs to be applied which corresponds to the solution derived in the previous section.

Apart from that, using

$$\boldsymbol{H} = (\boldsymbol{F_{Block}} \boldsymbol{W_{Block}}) \boldsymbol{O}.$$

the algorithm is quite straight-forward, as it basically consists of repeated ISTFT and STFT sequences that enforce the given spectrograms magnitude.

Algorithm 1 Phase Reconstruction Algorithm derived by Griffin and Lim

1: X_0 : $|\boldsymbol{FWO}\underline{x}|$; Phase: arbitrary
2: **for** $i = 1$ to I **do**
3: $\quad \underline{X}_{i,m} \leftarrow \boldsymbol{H}[(\boldsymbol{H}^H\boldsymbol{H})^{-1}\boldsymbol{H}_1^H]\underline{X}_{i-1,m}$
4: $\quad \underline{X}_i \leftarrow X_0 e^{j\angle X_i}$...element-wise
5: **end for**

The given spectrogram (Step 1) is transformed to the time-domain and then again a fourier transform is applied (Step 3) in order to combine the phase-information with the initial spectrogram $\underline{X}_0$ (Step 4). When applied for a certain number of iterations [GL84] the minimum mean squared error (minMSE) between $\underline{X}_0$ and $\underline{X}_i$ decreases. A summary on the vector spaces involved for Griffin and Lims algorithm is shown in figure 13.

As the gradient of D is strictly monotonically decreasing [GL84], this algorithm is capable of reducing the distance at every iteration step though convergence to a global minimum cannot be guaranteed.

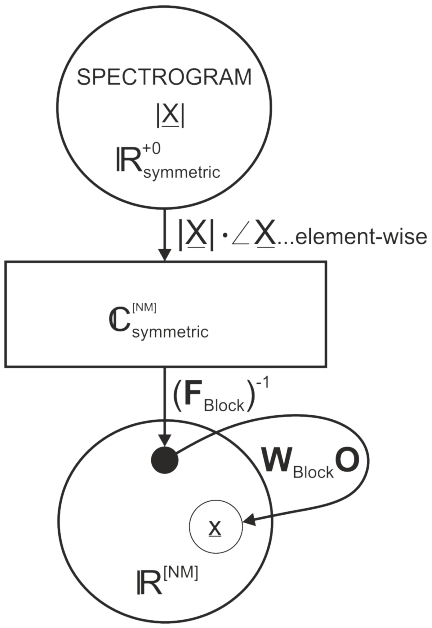


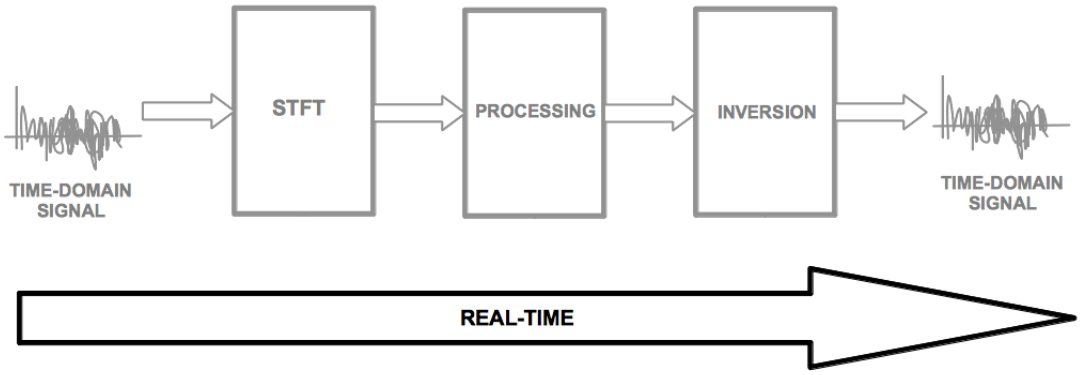
Figure 13: Vector spaces involved in the computation of the Griffin and Lim algorithm

Even though the obtained signals usually are perceptually close to their originals', as far as their structure in the time-domain is concerned, they often look different . In addition, the algorithm has three major drawbacks:

– at every iteration step, ISTFT and STFT need to be computed for the whole signal; this requires offline processing and increases computation time
– depending on the material, it can take a long time (i.e. a huge number of iterations) until convergence is reached
– the algorithm does not take any local information into account, this often results in a corrupted output signal (i.e. pre-echos, additional delays, phasyness)

Nonetheless, the algorithm provided by Griffin and Lim remains the core of nearly every state-of-the-art phase recovery process.

Real Time Spectrogram Inversion



The Griffin and Lim algorithm has one major drawback: in each iteration a DFT and an IDFT need to be performed over the whole signal. This not only increases computational cost but also requires knowledge of the whole signal in advance. Therefore this algorithm is not suited for many real-time applications. Some of it extensions , real-time capable at least structurally, will be presented in this chapter.

4 Real Time Spectrogram Inversion

4.1 Real-Time Iterative Spectrogram Inversion (RTISI)

The main drawback of the Griffin and Lim algorithm is the necessity of performing both a Fourier synthesis and analysis over the whole signal at each iteration step. Based on the assumption of spectrogram consistency in overlapping areas (a constraint that must be given in order to make perfect reconstruction possible), Zhu et al. [ZBW07] restructured the basic algorithm in order to make it real-time capable by restricting computations to one frame per iteration rather than transforming the whole signal.

Regarding one single frame $\underline{x}_m$ of length N within the context of matrix-notation $\boldsymbol{H} = \boldsymbol{F}_{Block}\boldsymbol{W}_{Block}\boldsymbol{O}$ enlightens why the naive approach of simplifying $\boldsymbol{F}_{Block}$ to $\boldsymbol{F}$ (with dimension [N x N]) and $\boldsymbol{W}_{Block}$ to $\boldsymbol{W}$ (with dimension [N x N]) is not sufficient for spectrogram inversion. Reconsidering the least squares solution derived in the previous chapter

$$\underline{x}_{LS} = (\boldsymbol{O}^T\boldsymbol{W}_{Block}\boldsymbol{W}_{Block}\boldsymbol{O})^{-1}\boldsymbol{O}^T(\boldsymbol{W}_{Block}\boldsymbol{F}_{Block}^{-1})\underline{X}.$$

reveals that the overlap matrix is preserved through the term $(\boldsymbol{O}^T\boldsymbol{W}_{Block}\boldsymbol{W}_{Block}\boldsymbol{O})^{-1}$ which corresponds to division of the time-domain signal $\underline{x}_m$ through the sum of all window functions squared (although for practical computational purposes, this sum can be limited to the dimension covered by $\underline{x}_m$ afterwards). Figure 14 shows the dimension obtained for an overlap of 1 sample and a signal length $N = 2$.

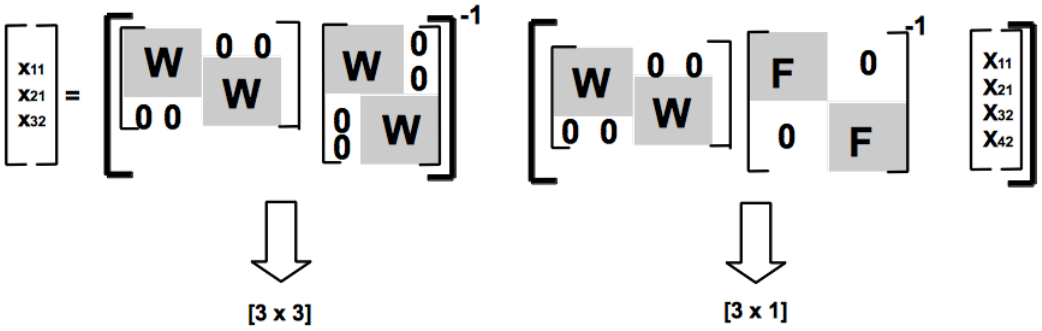


Figure 14: Simplified least Squares solution for N=2

Obviously, the result does not equal the output obtained by simply applying an IDFT. Therefore, when considering the STFT vector $\underline{X}_0$ as a stream of M overlapping (and therefore buffered) sections $\underline{X}_m$ with $M = \frac{N}{R} - 1$, the algorithm can be summarized quite similarly to the Griffin and Lim algorithm:

Algorithm 2 Basic RTISI as proposed by Zhu et al.

1: $\underline{X}_0$: $|\boldsymbol{H}\underline{x}|$; Phase: arbitrary
2: **for** frames: $m = 1$ to M **do**
3: **for** $i = 1$ to I **do**
4: $\underline{X}_{i,m} \leftarrow \boldsymbol{H}[(\boldsymbol{H}^H\boldsymbol{H})^{-1}\boldsymbol{H}_{\boldsymbol{1}}^H]\underline{X}_{i-1,m}$
5: $\underline{X}_{i,m} \leftarrow X_0 e^{j\angle X_{i,m}}$...element-wise
6: **end for**
7: $\underline{X}_m \leftarrow \underline{X}_{i,m}$
8: **end for**
9: $\underline{X}_0[m] \leftarrow \underline{X}_m$
10: $\underline{x}_{out} \leftarrow (\boldsymbol{H}^H\boldsymbol{H})^{-1}\boldsymbol{H}^H\underline{X}_0$

The starting point is a highly overlapping STFT of the signal $\underline{x}$, a known window function w, then the phase estimate of the current frame $\underline{x}_m$ can be obtained by performing a DFT on the sum of the windowed overlapping time-domain signals present at this frame (Step 1). Then again, Griffin and Lims' update rules are applied on the respective frame m (Step 2-5).

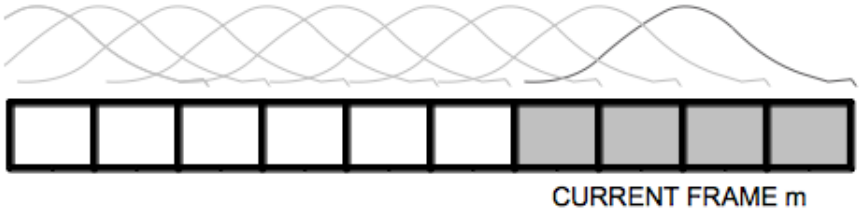


Figure 15: Basic Framework for RTISI with Overlap 4

Due to the lack of look-ahead, RTISI performed slightly worse than the original Griffin and Lim algorithm, since the signal could be optimized with regard to the already processed frames only without any knowledge about future frames. So RTISI was extended to a look-ahead stage RTISI-LA [ZBW06], where all overlapping frames with that currently under investigation are summed up in order to perform the phase updates. In the same work it was shown that assuming the number of look-ahead frames L is given by $R - 1$, with R the hopsize gives an optimal trade-off between computation time and reconstruction quality. RTISI-LA requires a slightly different framework, as the L subsequent frames to the one under processing are also updated according to Griffin and Lim, thus ensuring a phase estimate that corresponds to both, past and future values of the signal.

The algorithms structure remains the same if the respectives matrices depth M is assumed to range from $m - (\frac{N}{R} - 1)$ to $m + (\frac{N}{R} - 1)$.

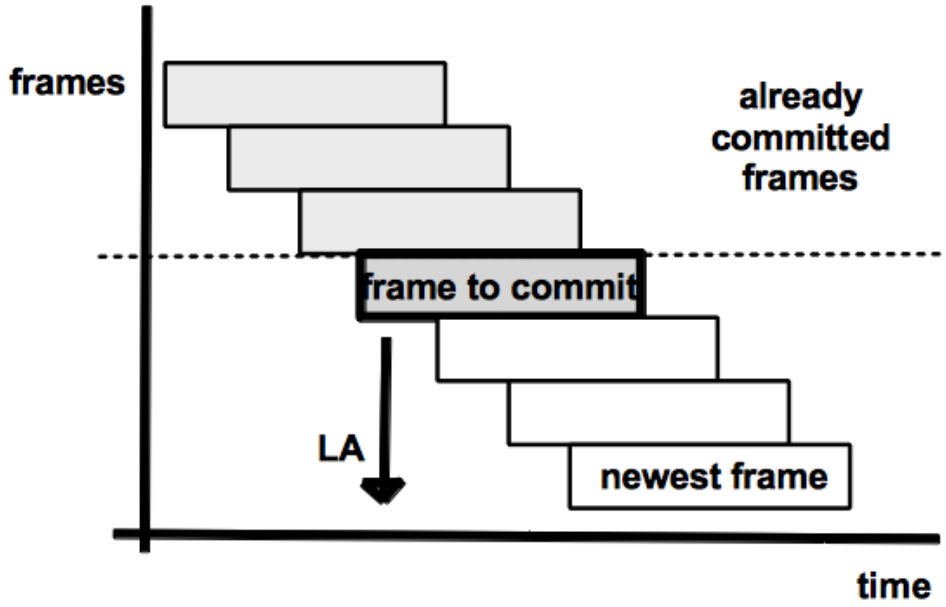


Figure 16: Framework for RTISI with Look-Ahead 3

4.2 RTISI Extended

Taking into account the respective STFT structure, further work has been done by Gnann and Spiertz [GS10], providing the RTISI with an unwrapped phase initialization and processing the respective frames according to their energy order rather than just sequentially. Also, they used different frame-lengths aiming to achieve a better temporal resolution for transients by assigning them a smaller frame [GS09]. Basically this latter approach could be extended to any integer multiple of the frame-length, thus allowing maximum freedom in the trade-off between time- and frequency-resolution by simply replacing the basic, longest frame with an arbitrary sequence of smaller ones where appropriate. Since RTISI is an iterative algorithm, using the phase information already available from the frames overlapping to the one under reconstruction for initalization seems to be a promising approach. So instead of setting the current frame to zero, initialization can be done by interpolating and unwrapping the phase of the buffer-sum.

Algorithm 3 RTISI Extended with Look-Ahead

X_0 : $|\boldsymbol{FWO}\underline{x}|$; Phase: arbitrary
for $m = 1$ to M **do**
  estimate frame-wise energy content E
  for $i = E_{max}$ to E_{min} **do**
    perform transient detection
    if transient **then**
      initialize phase as $\angle\underline{X}_m = \angle\boldsymbol{FO}x$
      split frames
      for $j = i - l_a$ to $i + l_a$ **do**
        $\underline{X}_{i,m} \leftarrow \boldsymbol{H}[(\boldsymbol{H}^H\boldsymbol{H})^{-1}\boldsymbol{H}^H]\underline{X}_{i-1,m}$
        $\underline{X}_{i,m} \leftarrow X_0 e^{j\angle X_{i,m}}$...element-wise
      end for
      $\underline{X}_m \leftarrow \underline{X}_{i,m}$
    else
      initialize phase as $\angle\underline{X}_m = \angle\boldsymbol{FO}\underline{x}$
      for $j = i - l_a$ to $i + l_a$ **do**
        $\underline{X}_{i,m} \leftarrow \boldsymbol{H}[(\boldsymbol{H}^H\boldsymbol{H})^{-1}\boldsymbol{H}^H]\underline{X}_{i-1,m}$
        $\underline{X}_{i,m} \leftarrow X_0 e^{j\angle X_{i,m}}$...element-wise
      end for
      $\underline{X}_m \leftarrow \underline{X}_{i,m}$
    end if
    $\underline{X}_0[m] \leftarrow \underline{X}_m$
  end for
end for
$\underline{x}_{out} \leftarrow (\boldsymbol{H}^H\boldsymbol{H})^{-1}\boldsymbol{H}^H\underline{X}_0$

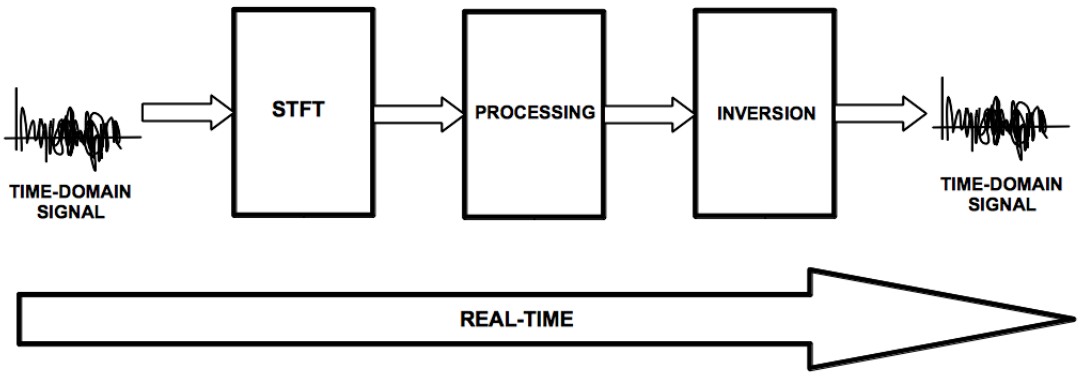
4.3 Fundamental Issues

Due to the sign indetermination of the phase, the algorithm might get stuck between $\angle(x)$ and $\angle(-x)$-the already mentioned stagnation. Occuring in all presented algorithms, this phenomenon generally provoques the largest reconstruction error and can only be encountered by knowing the sign in advance. Since the phase of overlapping lower frequencies changes less, this phenomenon also is frequency selective and gets more and more evident as frequency increases (as shown in [SD11]).

Generally, convergence remains an uncertain issue, both for the Griffin and Lim algorithm itself and for all extensions and modifications. The quadratic cost function usually causes convergence to happen in practice (at least in most cases), but it is neither be proven mathematically nor can an effective duration in steps be given for when convergence starts or is finished and at the moment nobody can say at which Signal-to-Error level the one or the other is at least likely to happen.

In addition, the circular nature of the DFT might lead to convergence happening on a frame's translated version introducing a time-shift on the original signal.
Another problem arises through the windowing necessary in STFT computation: usually many spectral components contribute to one frequency bin, thus forming a linear system that only finds a convenient solution when STFT-overlap and DFT-oversampling factors are high and the used window produces enough spectral leakage [Por79,SD11]. For practical conditions however, usually the contrary is desired in order to reduce computation time and data amount.

Implementation



5 Implementation

As outlined in the previous chapter, in order to implement RTISI, signal-buffering is an absolute necessity. Starting from a buffer similar to the one in figure (16), Spiertz and Gnann summarised the steps necessary in order to implement a RTISI with Look-Ahead [GS10]:

1. initialize last buffer row
2. calculate the sum of all buffer rows and limit it to the part covered by the last row
3. this sum has undergone a huge amount of analysis- and synthesis windowing, therefore compensate the window-sum error
4. calculate the phase spectrum of the sum by means of a DFT
5. combine the phase of the sum with the corresponding magnitude spectrum stored in the buffer
6. transform this combination into the time-domain
7. perform steps 2-7 on all frames required for look-ahead
8. perform steps 2-8 up to a certain number of iterations
9. commit the frame stored in the commit-frame row

While this list provides a good starting point for RTISI-implementation, during the course of this work it has become necessary to extend it by some considerations:

5.1 Basic Considerations

Step 1: Gnann and Spiertz [GS10] suggest that initialization depends on the application required and propose for general usage initialization with the unwrapped phase obtained after 3. However, despite its theoretical importance, initialization has not shown to affect the algorithms' output in any way, neither as far as the quality of the reconstruction nor as far as convergence time is concerned. On the other hand, when considering the trade-off between smooth phase aligning and relative-phase preservation shown in the vocoder-example in the Appendix, where an initialization with the unwrapped phase from the previously estimated blocks would correspond to the usual vocoder-approach without phase-locking, it might be, that this simply is the wrong approach in order to improve convergence.
Also, the overlap-add criterion, which usually guarantees STFT-consistence lives from its emergence. If enforcing the previous frame's phase contribution via according initialization of the current frame -without prior knowledge on the signal- really represents a suitable solution for the RTISI's convergence issues remains an open topic.

As far as 4 is concerned, for computational efficiency the suggested DFT will be replaced with the FFT in most cases. Since the FFT operates in its optimum on power of 2-length signals, and therefore most algorithms zero-pad signals up to the desired

length thus causing dimension inconsistencies and smearing for the latter RTISI operations, it should be considered to adapt the frame length accordingly. Moreover, this point leads to reconsiderations on the starting conditions: as already stated in chapter 2, establishing an overlap is necessary for assuming spectrogram consistency and thus being able to apply the Griffin and Lim approach in the first place. On the other hand, slicing an existing STFT in the spectral domain interrupts the overlapping-sequence and thus corrupts magnitude information. In the best case, an amplitude-modulated output signal is the result. So when the algorithm's goal is perfect reconstruction of the input signal, the frame-length must correspond to one FFT-length.
The same is true to a lesser degree for the relation of buffer-overlap and STFT-overlap. Changing the buffer-overlap with respect to that of the STFT corresponds to comparing two spectrograms with different parameters and some smearing will be inevitable. In summary, it is necessary to perform an STFT with known parameters on the time-domain signal and taking its absolute value before re-transformation and writing it into a buffer that has been constructed with the same characteristics as the spectrogram.

The number of iterations required is another important criterion for the algorithm's real-time capability: it ranges from 8 to 30 in literature [GS10, ZBW07, SD11]. In addition, since RTISI is not proven to converge and the duration until convergence starts varies and seems to be strongly dependent on the input, it can well be that for a certain signal and certain conditions an iteration number far greater than 100 is required. However, an iteration number of about 20 seems to suffice for most applications.

5.2 Experiments, Results and first Evaluation

In order to preserve a certain degree of comparability (especially to [GS10, GS09] , the testing scenario has been derived slightly modified from the Sound Quality Assessment Material (SQAM) from the EBU [Uni88].

- 25 random sweeps with noise
- sampling frequency 48 kHz
- Hamming window with window-length L=4R (overlap of 75%)
- 15 overall-iterations
- evaluation measure: $SER = 10 \cdot log \frac{\sum_{m=-\infty}^{\infty} \sum_{k=0}^{L-1} |X[mR,k]|^2}{\sum_{m=-\infty}^{\infty} \sum_{k=0}^{L-1} (|X[mR,k]| - |X'[mR,k]|)^2}$

As this SER-measurement operates on the magnitude spectrum, it depends on its own window-length. Unsurprisingly, the best results are achieved when the SER-window-length corresponds to the RTISIs spectrogram-window length as RTISI exactly minimizes the Mean-Squared Error between those borders given by its window length.

For 15 iterations, the standard RTISI achieved a comparable SER (about 20 dB) to [GS09, GS10, ZBW07, ZBW06]. However, the average computation time it took was about 66.348 seconds for an audio file of 1 second duration. As far as RTISI-LA is

FFT length	256	512	1024	2048	4096
Overlap 4	14.87	17.96	22.43	26.02	32.43
Variance [%]	0.62	0.42	0.42	0.69	0.10
Overlap 8	21.03	24.11	28.24	29.48	35.75
Variance [%]	0.67	0.31	0.40	0.36	0.88

Table 2: RTISI: Average SER [dB] for 25 random sweeps (rounded at 2 digits)

FFT length	256	512	1024	2048	4096
Overlap 4	17.93	21.52	23.05	21.88	19.99
Variance [%]	0.27	0.09	0.09	0.92	0.98
Overlap 8	36.96	41.93	43.13	42.33	40.46
Variance [%]	0.03	0.06	0.03	0.04	0.07

Table 3: RTISI-LA: Average SER [dB] for 25 random sweeps (rounded at 2 digits)

concerned in order to achieve the same SER, the number of iterations can be decreased to about 20 but computation time is still longer than for the standard RTISI (156.749 seconds). While the unwrapped phase initialization has not shown any improvement in reconstruction quality, it did not affect the computation time either. As far as the energy-order processing is concerned, computation time was increased at about 10 percent for the standard RTISI and about 30 for RTISI-LA.

For the testing scenario of random sweeps, RTISI and RTISI-LA perform as expected [ZBW07] with a reasonable variance for the mean SER averaged over the 25 output signals.

Generally, the reconstructed signals look very similar to the input signals -especially in the frequency domain. This is not surprising since the basic RTISI criterion states that the error between two spectra is subject to mimimization. Also perceptually, the algorithm delivers satisfying results. Unluckily this is not true for the computation time needed in order to deliver those. For audio signals, a delay smaller than 30 ms would be tolerable in order not to be perceptible by humans. Thus, RTISI fails the real-time requirement in the used programming environment (MATLAB).

5.3 The Real-Time Constraint

Considering the basic computational terms, real time ability in general is a weak expression as it always depends on the respective application and its framework. None of the papers dealing with RTISI (e.g. [ZBW07, GS09, GS10, SD11]) gives constraints required to achieve capacities in terms of computational load per second. Nevertheless, comparability of different algorithms with regard to real-time capacity is both desirable and a necessary goal. As already mentioned, the proposed iteration numbers range from 8 to 20. A closer investigation reveals that Zhu et al. [ZBW07], who suggest the lowest iter-

ation number, achieve an average SER of about 20 dB while Spiertz and Gnann obtain slightly better results (up to 28 dB) with a slightly larger number of iterations (about 12).

Certainly the repeated computation of FFTs and IFFTs represents a certain temporal challenge and moreover, the real advantage of using an FFT, the weak dependency of its computational demand on large signal lengths, cannot be exploited within signal (frame- lengths) of only about 2000 samples.

However, we also observed in some cases convergence not taking place immediately. Even though this must rather be seen as an informal statement, as no studies dealing explicitly with this topic were conducted, claiming that this is more an issue of the standard Griffin and Lim algorithm and that the extensive frame-overlapping done in the RTISI-context usually prevents the algorithm from diverging [ZBW07] seems justified.

5.4 RTISI Restructured

Pure Data is an object-oriented, graphical programming language in the tradition of *MAX/ISPW* intended for real-time audio signal processing and computer music [Puc96]. It provides two scheduling layers:
– Signal-Layer: for direct manipulation of signals, works in real-time synchronous to the computers' soundcard
– Control-Layer: designing the sound-manipulation framework, works asynchronous on-demand, suitable for more complex calculation processes

While signals need to be processed constantly in order to preserve soundcard synchronicity, for control messages, there is no need for constant calculation and they are processed as quickly as possible. The *iemmatrix* library delivers a suitable framework for doing calculations in the control domain, while keeping them very similar to MATLAB structurally. However, Pure Data processes one block of given size per DSP-cycle and therefore, the buffer-structure slightly changed as shown in figure 17.

In order to gain real-time ability while preserving a certain flexibility in order to being able to eventually include improvements and extensions to the code as well as to downsize it accordingly if required, also the preparation step needs to be reconsidered thoroughly.

The main difference is, as already mentioned, instead of working with frames as the smallest unit, this algorithm is working with blocks. Thus, a block is load into the buffer and shifted until it reaches the limit-and-sum stage. There, RTISI is performed.

Preparations

At first some parameters are determined:
– the desired FFT length N

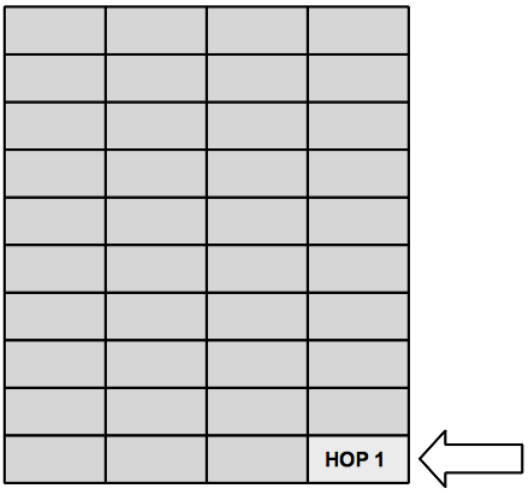


Figure 17: Buffer-Structure for overlap=4 and 16 iterations

– the desired overlap-factor, which corresponds to the STFT-overlap, since RTISI has been shown to deliver its best performance when those parameter match [ZBW07, SD11]
– the desired number of iterations n_{iter}

Then the blocklength in PureData corresponds to the desired hopsize R and subsequently N corresponds exactly to a frame-length of the other RTISIs. The basic idea is letting a block (subsequently referred to as HOP) travel through the buffer, while updating it at each stage.

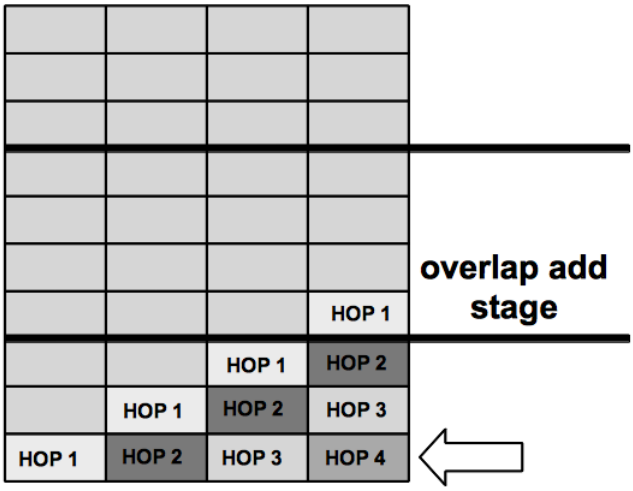


Figure 18: Overlap-add stage

This updating is performed via matrix multiplication, so at each step, all HOPs currently within the buffer are processed. Therefore, the size of the resulting buffer corresponds to $(N \times M)$ with N the FFT-length and M the number of FFTs performed at one iteration. Since for RTISI, more frames than those under reconstruction are required due to the overlap-add criterion not being fulfilled at the buffers borders, the number of iterated HOPs per iteration is smaller than M.

The preparation steps can be summarized as follows:

1. load HOPs sequentially into a buffer of dimension [1, N], until this buffer then corresponds to one FFT-frame
2. buffer those buffer-rows into a 2-dimensional buffer with the length N and a depth M for every incoming HOP
3. perform row-wise FFT over the whole buffer
4. compute spectrogram via taking its absolute value
5. update the buffer for each new incoming HOP

The result is an absolute-valued STFT (i.e. a spectrogram), where the STFT-length corresponds to $overlap \cdot R = N$, and the other dimension M can be chosen with respect to the desired number of RTISI-iterations and the tolerable time-delay. This finally is the input sent into RTISI, which when shifted corresponds to the buffer structure presented in chapter 4.

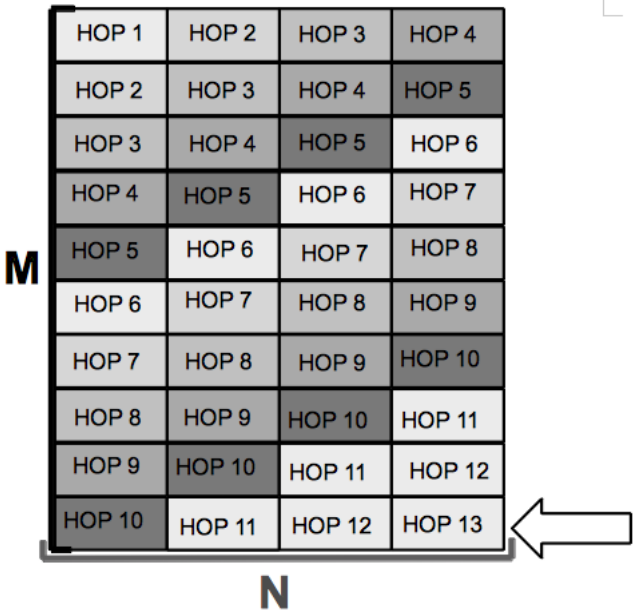


Figure 19: Full buffer

Referring to figure 19, not only the youngest HOPs need to get to the overlap-add stage in order to being processed correctly, but HOPs 1 - 3 can not be fully overlapped either, therefore HOP 4 is the oldest signal slice updated and can be written out.

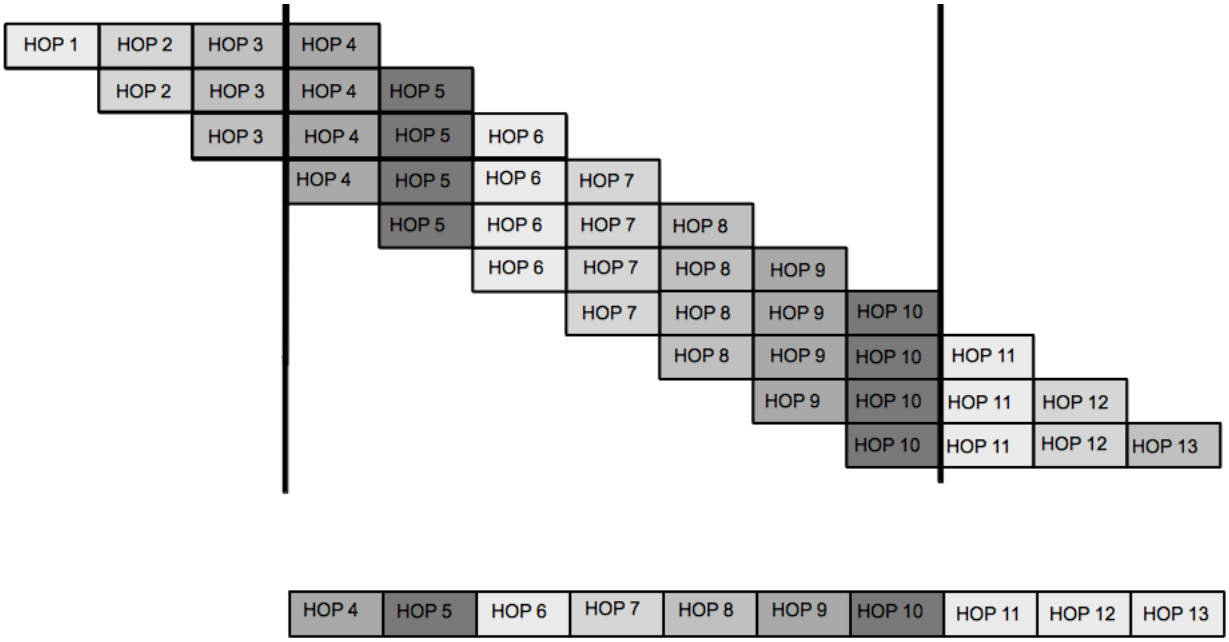


Figure 20: Full Buffer shifted

The RTISI algorithm

Basically, once a HOP enters the buffer, it is processed immediately since the whole spectrogram-matrix undergoes the modified ISTFT already outlined in chapter 3, but at the summing-up stage, as shown in figure 20, only the first HOP of the last row is taken into account. In addition, it has to be taken care of preserving the magnitude spectrum contained within the buffer somehow for combining it with the phase values obtained by RTISI later on.

The size of the then resulting matrix corresponds exactly to the size of the overlap-add stage in figure 19, summing up the signal from Block 4 to 10.

This sum is the buffered again, according to

finished	finished	finished	HOP 4
finished	finished	HOP 4	HOP 5
finished	HOP 4	HOP 5	HOP 6
HOP 4	HOP 5	HOP 6	HOP 7
HOP 5	HOP 6	HOP 7	HOP 8
HOP 6	HOP 7	HOP 8	HOP 9
HOP 7	HOP 8	HOP 9	HOP 10
HOP 8	HOP 9	HOP 10	void
HOP 9	HOP 10	void	void
HOP 10	void	void	void

Figure 21: The summed-up sequence buffered again

In a next step, this matrix is retransformed into the Fourier domain, its phase is calculated and combined with the input spectrogram matrix according to figure 19. For the newest frame (i.e. the last buffer row), phase can be initialized arbitrarily. Then, the next HOP is load into the buffer, all HOPs are shifted one position further and the procedure is repeated.

The first difference as compared to the state of the art structure consists in providing a reduced set of parameters without loss of generality. This enforces usability and ensures a relatively simple structure of the algorithm since in addition to the window-type, instead of previously seven, only three parameters need to be determined and two of them are already given by the STFT in most cases.

Without any additional measures to conquer spectral smearing, window-type and -length as well as the buffer's framelength should correspond to the number of FFT bins anyway, while RTISI has been shown to operate in its optimum when its overlap corresponds to the STFT overlap [ZBW07, SD11]. Linking the number of iterations and thus the time needed for computation to the buffer depth is a common approach in real-time computation that enforces the universal applicability of an algorithm since it facilitates up- and downsizing according to the respective's host processors requirements.

RTISI Parameters	
STFT Overlap Buffer Overlap	STFT Overlap
Buffer Blocklength FFT-length N Window Size	FFT length N
Buffer Depth Number of Iterations	Number of Iterations

Table 4: RTISI Parameters and their conjunction in the algorithm

5.5 Considerations on Realtime Capability

For calculations involving a huge amount of floating point operations, at the moment FLOPS (Floating Point Operations per Second) deliver the most accurate measure of computer performance.

The overall performance of a computer can be calculated as

$$FLOPS = cores \times clock \times \frac{FLOPs}{cycle} \tag{25}$$

and a typical processor nowadays can do about 4 FLOPs per cycle [Ver09]. Since the algorithm has been implemented in Pure Data which is conceptually not capable of multi-threading, only one core is used for processing. Assuming a 2.4 GHz clock, this yields a theoretical performance of about 9.6 billion FLOPS. However, as far as effective processing time is concerned this can rather be seen as a theoretical upper limit since the complexity of current computer systems prevents from direct comparison and equation 25 does not take into account practical issues related to the respective hardware implementation.

The Multiply-Accumulate (MAC-) procedure where at each instruction a floating point multiplication is computed and added to an accumulator, is very common in digital signal processing due to its elementary importance for filtering operations. Typically it is implemented as shown in figure 22 and is therefore computed as one single FLOP.

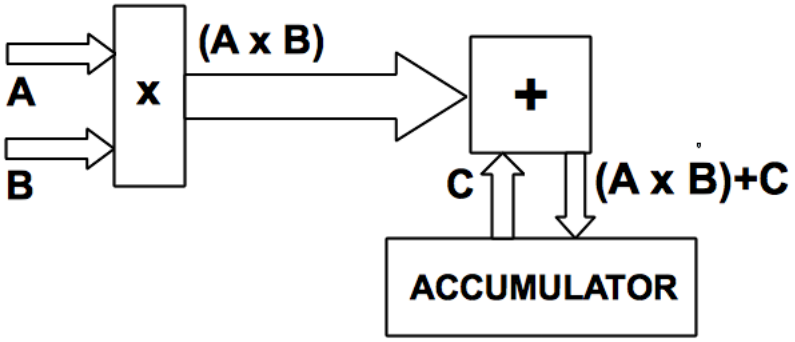


Figure 22: MAC operation within one instruction cycle

Assuming an FFT-length N, one HOP undergoes the following processing Computing one real FFT requires $NldNMACs$. At each position, the respective block undergoes a real FFT and IFFT, therefore

$$\#MACS = 2 \times NldN.$$

The number of HOP-positions depends entirely on the buffer-depth M, the FFT length N and the hopsize R. The number of additions per iteration amounts to the number of samples within the buffer $N \cdot M$ decreased by the finished and void blocks yielding (for even overlap factors)$2 \cdot N \cdot (\frac{R-1}{2}) = N \cdot M - N \cdot (R-1)$.
In summary, this gives the following number of calculations:

Operations		**MACs**
per iteration	rfft	$2 \cdot Nld(N)$
	rifft	$2 \cdot Nld(N)$
	Overlap-Add	$N \cdot M - N \cdot (R-1)$
# HOP-Positions		$\frac{N}{R} \cdot M$

Since an iteration number of 25 has shown to be sufficient for many applications, an estimate for the maximum HOPlength yields the result shown in figure 23:

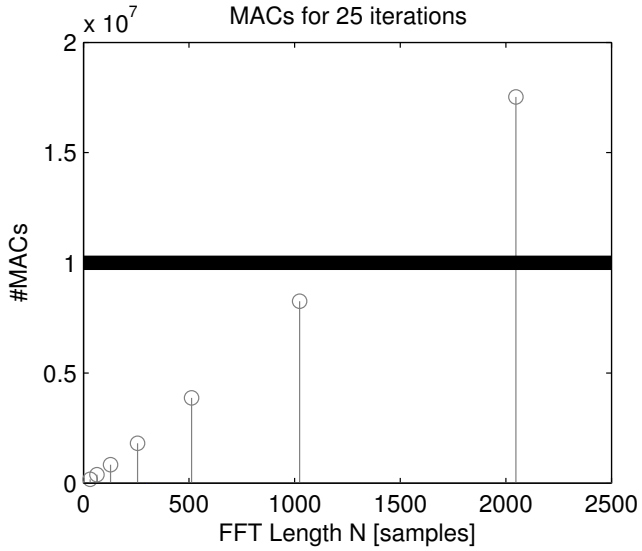


Figure 23: MACs required for respective FFT-lengths N

The black line designates the absolute upper limit of operations per second a state-of-the-art processor is able to perform. Therefore, for an overlap of 4, the blocklength should not exceed 256 samples. This yields a framelength of 1024 samples which seems to be at least a good compromise between time- and frequency resolution [SD11].

Therefore, in general, realtime computation is possible with a high-level state-of-the art computer system-especially when multithreading can be used, but it still is close to the limit of typical computational capacities.

6 Outlook and Conclusion

In this thesis, a framework for real-time iterative spectrogram inversion has been designed and implemented.

Chapter 2 provides an in-depth revision on the theory of spectral signal processing with a focus on the STFT and its computation. In order to facilitate further computation, the STFT was derived in vector form.

In Chapter 3 the conditions under which spectrogram inversion is theoretically possible are clarified. Derivation of a least squares approach leads to the Griffin and Lim algorithm which is the core of most state-of-the-art applications.

Subsequently in chapter 4 its extensions to real-time capability and several improvements concerning the adjustment of the respectives algorithms parameters with regard to the signal under reconstruction are discussed.

Chapter 5 features the implementation of the Real-Time Iterative Spectrogram Inversion algorithm (RTISI), an evaluation and gives a note on its realtime capability. In addition, all the knowledge derived from the previous chapters is combined within the implementation of a simplified and more flexible, block-based and real-time capable framework. Also, the provided structure favors down- and upsizing according to the respective platforms' processing capabilities and can thus be implemented on various devices.

After all, the algorithm still does not take into account any knowledge that might be obtainable from the spectrogram. There are various low-level features that could be applied on the spectrogram at negligible computational cost in order to pre-inform the RTISI on its structure. For example, the last mentioned structure could be extended to a dual resolution method using Miller Puckette's pd~-object [Puc09]. A transient detection could be performed, deriving the optimal length for the respective block. Pd~ could enable running two (or more) whole RTISI-buffers running simultaneously, one for short and one for long blocklengths. A final overlap-add stage could then resynthesize the signal according to its degree of harmonicity.

Momentarily, in order to ensure the real time requirement, enabling the algorithm to multi-threading seems an absolute necessity since for the next couple of years processing units are more likely to grow larger in number than in singular computational capabilities.

Further investigations could include the consistency criterion as an interpolation for steady spectrogram components while transients could be estimated via the standard RTISI thus saving further computation time.

References

[AKP07] L. Alsteris, K. Kuldip, and K. Paliwal, "Iterative reconstruction of speech from short-time fourier transform phase and magnitude spectra," *Science Direct, Computer Speech and Language, vol. 21*, pp. 174–186, 2007.

[CSD10] E. Cano, G. Schuller, and C. Dittmar, "Exploring phase information in source separation applications," *Proceedings of the* 13^{th} *Int. Conference on Digital Audio Effects (DAFx-10)*, pp. 1–7, 2010.

[DH04] N. Dourdoumas and M. Horn, *Regelungstechnik*. Pearson Verlag, 2004, ($2^{nd} Edition$).

[Dud39] H. Dudley, "The vocoder," *Bell Labs Rec., vol. 18*, pp. 122–126, 1939.

[Ell02] D. Ellis, *A phase vocoder in Matlab*. http://www.ee.columbia.edu/dpwe/resources/matlab/pvoc/, 2002, online book.

[FW86] J. Fienup and C. Wackerman, "Phase-retrieval stagnation problems and solutions," *J.Opt.Soc.Am.A,vol. 3*, pp. 1897–1907, 1986.

[GL84] D. Griffin and S. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1984.

[Gro14a] M. Group, "Alu performance - sisoftware sandra 2010 pro (alu)," accessed February, 24^{th} 2014, online ressource.

[Gro14b] ——, "Cpu charts," accessed February, 24^{th} 2014, online ressource.

[Gro14c] ——, "The intel core i7 - processor review," accessed February, 24^{th} 2014, online ressource.

[GS09] V. Gnann and M. Spiertz, "Inversion of short-time fourier transform magnitude spectrograms with adaptive window lengths," *Proceedings of the* 12^{th} *Int. Conference on Digital Audio Effects (DAFx-09)*, 2009.

[GS10] ——, "Improving rtisi phase estimation with energy order and phase unwrapping," *Proceedings of the* 13^{th} *Int. Conference on Digital Audio Effects (DAFx-10)*, 2010.

[HLO80] M. Hayes, J. Lim, and A. Oppenheim, "Signal reconstruction from phase or magnitude," *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 672–680, 1980.

[KC92] R. Kent and R. C, "The acoustic analysis of speech," *San Diego: Singular Publishing Group, Inc*, 1992.

[LKOS10] J. LeRoux, H. Kameoka, N. Ono, and S. Sagayama, "Fast signal reconstruction from magnitude stft spectrogram based on spectrogram consistency," *Proceedings of the* 13^{th} *Int. Conference on Digital Audio Effects (DAFx-10)*, pp. 1–7, 2010.

[NQL83] S. Nawab, T. Quatieri, and J. Lim, "Signal reconstruction from short-time fourier transform magnitude," *IEEE Transactions on Acoustics, Speech and Signal Processing,vol. 31*, pp. 986–998, 1983.

[Nut81] A. Nuttall, "Some windows with very good sidelobe behavior," *IEEE Transactions on Acoustics, Speech and Signal Processing, vol.ÊASSP-29*, pp. 84–91, 1981.

[OS99] A. Oppenheim and R. Schaefer, *Discrete-Time Signal Processing*. Prentice-Hall Signal Processing Series, 1999, (2^{nd} Edition).

[Pal03] K. Paliwal, "Usefulness of phase in speech processing," *Science Direct, Computer Speech and Language, vol. 21*, pp. 174–186, 2003.

[Por76] M. Portnoff, "Implementation of the digital phase vocoder using the fast Fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing, vol.ÊASSP-24*, pp. 243–248, 1976.

[Por79] ——, "Magnitude-phase relationships for short-time fourier transforms based on gaussian analysis windows," *IEEE Conference on Acoustics, Speech and Signal Processing, ICASSP 79, vol. 4*, pp. 186–189, 1979.

[Puc95] M. Puckette, "Phase-locked vocoder," *Proceedings of the IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 222–225, 1995.

[Puc96] ——, "Pure data," *Proceedings of the Int. Computer Music Conference*, pp. 224–227, 1996.

[Puc09] ——, "Multiprocessing for pd," *Proceedings of the 3^{rd} pure data Int. convention*, 2009.

[QH95] T. Quatieri and T. Hanna, "Time-scale modification with inconsistent constraints," *Proceedings of the IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 263–266, 1995.

[Roe03] A. Roebel, "A new approach to transient processing in the phase vocoder," *Proceedings of the 14^{th} Int. Conference on Digital Audio Effects (DAFx-03)*, pp. 1–6, 2003.

[Sal07] J. Salsman, *Matlab programming/phase vocoder and encoder*. http://en.wikibooks.org/wiki/MATLAB$_{P}rogramming/Phase_{V}ocoder_{a}nd_{E}ncoder, 2007$.

[SD11] N. Sturmel and L. Daudet, "Signal reconstruction from stft magnitude: A state of the art," *Proceedings of the 14^{th} Int. Conference on Digital Audio Effects (DAFx-11)*, pp. 375–386, 2011.

[Smi10] J. Smith, *Physical Audio Signal Processing*. https://ccrma.stanford.edu/ jos/pasp/, 2010, online book.

[Smi14] ——, *Spectral Audio Signal Processing*. `http:`http://ccrma.stanford.edu/ jos/sasp///-`ccrma.stanford.edu/~jos/sasp/`, accessed January, 14^{th} 2014, online book.

[Sue11] S. Suess, "Auffinden von Ursignalen aus Aufnahmen umgebender kugelfoermiger Mikrofonanordnungen," 2011, Project Report, University of Music and performing Arts Graz.

[Uni88] E. B. Union, "Sound Quality Assessment Material," *Tech 3253*, 1988.

[Ver09] D. Versick, *Verfahren und Werkzeuge zur Leistungsmessung -analyse und -bewertung der Ein-/Ausgabeeinheiten von Rechensystemen*, 2009, dissertation, Universitaet Rostock.

[Yan08] B. Yang, "A study of inverse short-term Fourier transform," *IEEE Int. Conference on Acoustics, Speech and Signal Processing*, pp. 3541–3544, 2008.

[YSK84] B. Yegnanarayana, D. Saikia, and T. Krishnan, "Significance of group delay functions in signal reconstruction from magnitude or phase," *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 610–622, 1984.

[ZBW06] X. Zhu, G. Beauregard, and L. Wyse, "Real-time iterative spectrum inversion with look-ahead," *IEEE Transactions on Multimedia and Expo*, pp. 229–232, 2006.

[ZBW07] ——, "Real-time signal estimation from modified short-time fourier transform magnitude spectra," *IEEE Transactions on Audio, Speech and Language Processing, vol.15 no.5*, pp. 1645–1653, 2007.

A Characteristics of Audio Signals

Generally spoken, there are two different kinds of audio signals: speech and music. Since both are acoustic events and therefore belong to the realm of real signals and are equally perceived by the human ear, it is not easy to find some characteristic differences. Even worse. There are lots of systematic similarities, as all languages consist of phonemes and all musical systems consist of notes. The specific sets thereof may vary, but from psychoacoustics it is known that human perception works best in the spectrum from near zero to about 3.5 kHz. This might be due to the reason that speech covers the same range and in any case, is used by songwriters, composers, instrument manufacturers and sound engineers in order to promote their music. For both audio waveforms is true that most of the signal power is concentrated at lower frequencies.

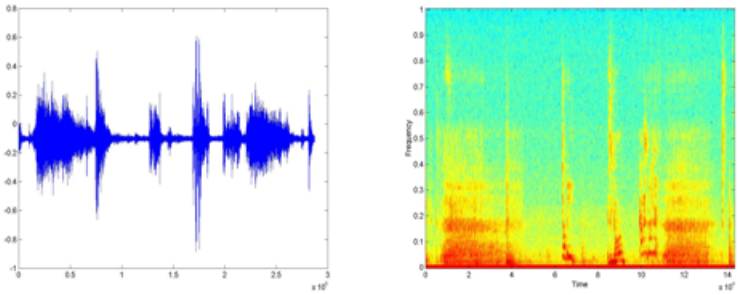


Figure 24: Speech Signal

However, after closer investigation, some differences of the signal's form and distribution can be derived.

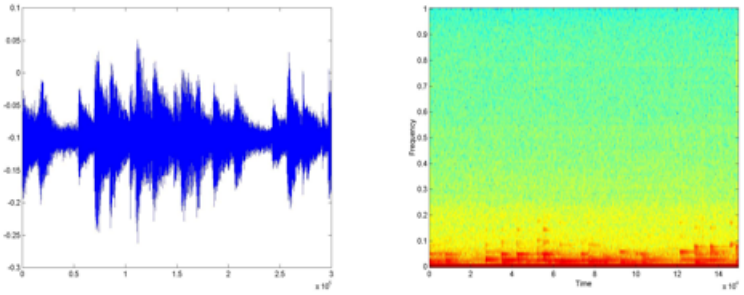


Figure 25: Music Signal

From a theoretical point of view, there are even more differences as a typical music spectrum has about twice the bandwidth of a speech spectrum. Usually, there don't

occur any sounds between A_1 (at about 27.5 Hz) and c^8 (at 4186 Hz) in music although musical instruments theoretically may cover more than the audible band.

Music tends to be composed of a multiplicity of tones, each with a unique distribution of harmonics. This pattern should remain consistent regardless of the type of music or instruments while in speech, much more emphasis is placed on voice tonality. Also, speech exhibits an alternating sequence of noise-like segments while music alternates in more tonal shape. Put differently, a speech signal is distributed through its spectrum more randomly than music. This also explains the functionality of speech simulation as an Auto Regressive process.

Speech has concentrated most of its power in frequencies lower than 4 kHz and is limited to 8 kHz, whereas music can extend through the upper limits of the ear's response at 20 kHz. But, as already mentioned, most of the signal power in music is concentrated at lower frequencies as well.

Both signal types really differ as far as the power spectral distribution is concerned. Usually the power of speech concentrates at low frequencies, then collapses very fast through the higher values and contains no DC. For music, there is no such specific shape. If a specific person talks alone, human tend to perceive the fundamental frequency almost accurately. This is not necessarily the case for a specific music instrument. A instrument in general has many fundamental frequencies while the speech of a specific person has only one. The same goes for the dominant frequency which also can only be determined in its average when many musical instruments play simultaneously since the amplitude reaches its maximum in a wide spectral range. For speech, this task is much easier.

Excitation patterns for speech usually only cover a span of three octaves present, while the fundamental music tones can span a range up to six octaves. Also the duration of vowels in speech is limited by the talking person's breath and is therefore very regular. Not being constrained by the process of articulation, music exhibits a wider variation in tone lengths.
Some additional generalizations can be made on the pattern speech exhibits: high-energy conditions of voicing follow low-energy conditions. The envelope of music is unlikely to exhibit such a shape -neither spectral nor temporal- and will be more continuous through time. Also, in most cases, the zero-crossing rate in music will be greater than in speech.

B Fast Fourier Transform

While the standard DFT delivers complexity $O(N^2)$ as the computation of N inner products of length N is required, for an input signal of length N, where N is a power of 2, the Cooley-Tukey algorithm delivers complexity $O(NlgN)$. This can be achieved by breaking up the original N-point signal into two sample-sequences of length $\frac{N}{2}$. This way, the length N-DFT is replaced by a several length-2 DFTs which can be computed without any multiplication. This can be seen when rewriting the DFT in matrix-notation $X(\omega) = Wx$ with x the input signal.

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ : \\ X[N-1] \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & ... & 1 & \\ 1 & \phi & \phi^2 & \phi^3 & ... & \phi^{N-1} \\ 1 & \phi^2 & \phi^4 & \phi^6 & ... & \phi^{2(N-1)} \\ 1 & \phi^3 & \phi^6 & \phi^9 & ... & \phi^{3(N-1)} \\ : & : & : & : & & \phi^{:} \\ 1 & \phi^{N-1} & \phi^{2(N-1)} & \phi^{3(N-1)} & ... & \phi^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ : \\ x[N-1] \end{bmatrix} \tag{26}$$

with $\phi = e^{-\frac{j2\pi}{N}}$

For the case of the 2-point DFT this gives:

$$\begin{bmatrix} X[0] \\ X[1] \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} = \begin{bmatrix} x[0] + x[1] \\ x[0] - x[1] \end{bmatrix} \tag{27}$$

since according to Euler's theorem:

$$e^{j\theta} = cos(\theta) + jsin(\theta) \tag{28}$$

$$e^{j2\pi k} = cos(2\pi k) + jsin(2\pi k) = 1 \forall k \in N$$

Those DFT-fractions can be recombined later via multiplication. Most FFT-algorithms operate in their optimum when the input signal's length is n^2, usually the input is zero-padded accordingly.

In order to reduce an even signal of length N to its 2-point DFTs, the DFT-sum can be split into its odd and even terms:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x[n] e^{-jw_k n} \\ &= \sum_{n_{even}=0}^{N-2} x[n] e^{-jw_k n} + \sum_{n_{odd}=0}^{N-1} x[n] e^{-jw_k n} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[2n] e^{-j2\pi \frac{k}{\frac{N}{2}} n} + e^{-j2\pi \frac{k}{N}} \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] e^{-j2\pi \frac{k}{\frac{N}{2}} n} \end{aligned}$$

This splitting-approach is called *decimation in time*. When applied in the frequency-domain by splitting the number of frequency bins (for the IDFT), it is called *decimation in frequency*. For the case where $N = 2^k$, where $k > 1$ and an integer, this decomposition can be done $K - 1$ times, yielding several length-2 DFTs for which no multiplication is required. In fact, the only multiplies needed are (approximately) N ones for re-combining the $\lg N$ time-decimation stages. Thus, the radix-2 Cooley-Tukey algorithm yields the complexity $O(N \lg N)$.

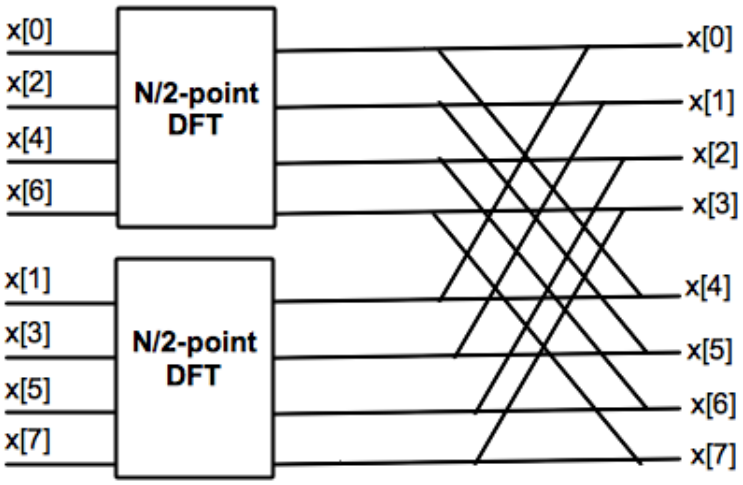


Figure 26: Data-Flow Diagram for the FFT

The overall FFT-algorithm summarized:

Algorithm 4 Fast Fourier Transform

1: $X_{0,\ldots,N-1} \leftarrow DIT_{fft2}(x, N, s)$
2: $X_{0,\ldots,\frac{N}{2}-1} \leftarrow DIT_{fft2}(x, \frac{N}{2}, 2s)$
3: $X_{\frac{N}{2},\ldots,N-1} \leftarrow DIT_{fft2}(x+s, \frac{N}{2}, 2s)$
4: **for** $k = 0$ to $\frac{N}{2} - 1$ **do**
5: $\quad t \leftarrow X_k$
6: $\quad X_k \leftarrow t + e^{-j2\pi \frac{k}{N}} X_{k+\frac{N}{2}}$
7: $\quad X_{k+\frac{N}{2}} \leftarrow t - e^{-j2\pi \frac{k}{N}} X_{k+\frac{N}{2}}$
8: **end for**

Generally, there are also other algorithms for FFT-computation at hardly the same cost (such as Bluestein's, Rader's, Bruun's or the Prime-Factor FFT), but not all of them are universally applicable and the Cooley-Tukey algorithm is the most commonly used method.

Additional computation time can be saved when acomodating the fact that real signals

satisfy the symmetry property

$$X_{N-k} = X_k^* \tag{29}$$

where X^* is the complex conjugate of X. Therefore, there is a lot of redundancy that can be removed in order to make the FFT-computation more efficient and computation effectively only needs to be performed on about half of the samples.

C TSM: Operating directly on the STFT

A typical example for the problem of phase aligning is pitch-shifting (or, equivalently, time-stretching). The frequency of a sound piece is changed either in order to correct the pitch or to change the pieces tempo. This was also the application the first universal phase estimation approach by Griffin and Lim [GL84] was dedicated to. As already derived in the introduction, phase depends on the frequency and thus, has to be corrected accordingly. The common standard approaches to do so will be discussed in this chapter.

C.1 Time Scale Modification

TSM aims to either change a sound files tempo without affecting its frequency content (i.e. the perceived pitch) or to change the frequency content (e.g. changing the pitch) without affecting the file's temporal structure. The first is especially useful for speech processing purposes since perceptually, the formant spacing is a measure of the talking human's vocal tract and therefore should not be affected by time-stretching or -squeezing as the whole impression and characteristic of the speaker would be changed (*Munchkin Effect* [Smi14]). But also music sounds natural only within certain limits of temporal deviation [Smi10]. Changing the frequency content however, is usually a desired feature in recording studios providing often the last possible method to correct a wrong note.

The basic implementation of TSM happens usually within the sines-noise-transient (S+N+T) framework. When a transient is detected, it is translated to the new signal unchanged only the harmonic (and noisy) parts of the signal undergo the desired modification.

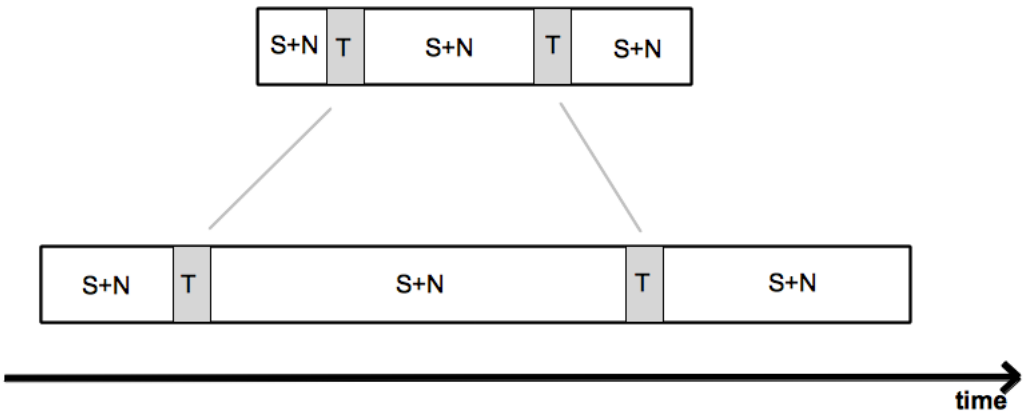


Figure 27: Time Scaling using S+N+T

The time-scale modification itself can be efficiently done in the spectral domain based on the STFT and its properties: At first the STFT is performed as outlined in chapter 2. Then, for resynthesis, the hopsize R is changed according to $\frac{R}{\alpha}$, where α denotes

the desired scaling factor. Under the assumption of perfect transient detection for time-squeezing purposes, no major error is to be expected, as the FFT of a continuous harmonic is just shortened. When the time-scale needs to be stretched, however, spectral interpolation becomes necessary, both for magnitude and phase. For the simplest case of a rectangular window with no overlap, the first STFT frame will be resynthesized normally, but the starting point for the second frame will be situated somewhere in-between the first two frames and some magnitude information needs to be invented. Linear magnitude interpolation gives perceptively good sounding results and can be computed quite easily:

$$X_{new}(\omega) = [(1 - m_{i-1} + m_i)|X_{i-1}(\omega)| + (m_{i-1} - m_i)|X_i(\omega)|]e^{j\theta(\omega)} \qquad (30)$$

with X_i and X_{i-1} the STFT frames the new frames X_{new} starting point is situated in-between and where m is the "FFT-pointer" which is shifted each step by $\frac{R}{\alpha}$.

The question that arises now is how to choose a suitable phase $e^{j\theta(\omega)}$? And how does it evolve over time?

Interpolating phase is in fact not easy since on the one hand, what is known about the relative phase at a certain bin k should be preserved in order to prevent attack-smearing. On the other hand however, frame transitions should be smooth and therefore some adjustment is inevitable. And as a third condition, the overlap-add property must not be violated in order to ensure perfect reconstruction.

In the context of the S+N+T-model, it might be applicable to keep relative phase values within the transient-section while taking care of a smooth phase continuation for the harmonic parts of the signal. For a perceptual satisfying result it is neither possible nor necessary to have continuos phase propagation for all harmonics within a signal.

Typically in literature, three approaches can be found for accomplishing the task of phase aligning:

– aligning phase continuously: Phase-Vocoder
– preserving relative phase information: Phase-locked Vocoder
– operating in the time-domain: Overlap-Add TSM

C.2 Phase Vocoder

When proposed in 1928 [Dud39], the channel vocoder was initially intended to encode speech electronically. Based on the assumption of computing the envelope from a smooth sinusoidal signals, it consisted of filterbank driven by a relaxation oscillator and resistor noise respectively for simulating voiced and unvoiced speech. It was reinvented as the phase-vocoder in the 60ies of the previous century [Puc09]. Computer technology enabled its implementation as a sliding STFT and both, amplitude and phase values could be computed, saved and modified. Soon thereafter, the phase vocoder became the tool of choice for computer musicians performing additive synthesis and can be said to have

pioneered the following subband-coders. Many variations and improvements have been proposed and apart from the time scale modification referred to in this work, it has been used in speech- and data compression and for many musical related applications such as reverb suppression and frequency shifting.

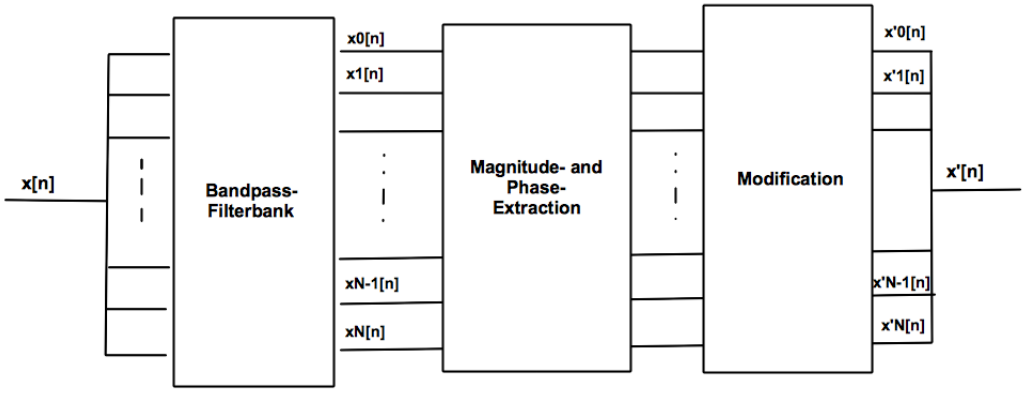


Figure 28: Schematic of the Vocoder

The phase vocoder [Dud39, Ell02, Por76] does not keep any relative phase information when continuously unwrapping the phase. As a result, the time-domain signal obtained is severely corrupted as can be seen in the next figure.

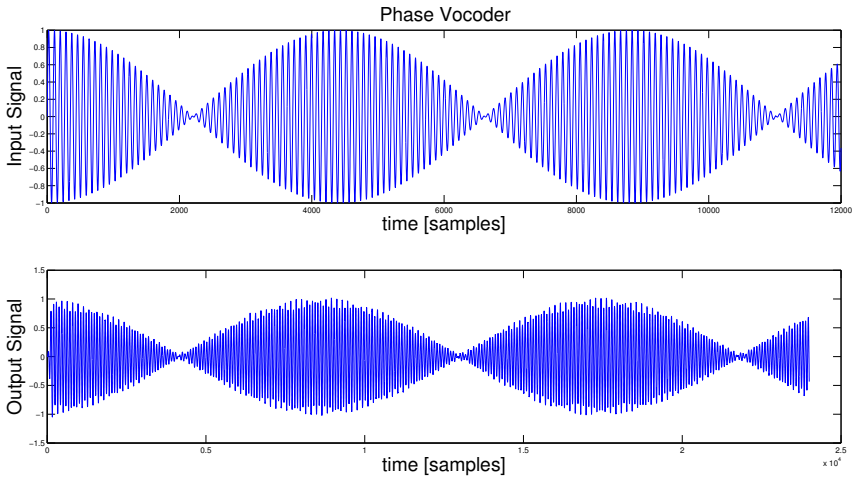


Figure 29: Time-Stretching using the Phase Vocoder - Time Domain

Further investigation in the frequency domain reveals that also the magnitude coherence suffers from this approach. The incoherent amplitude values at the frames' transition

points lead to distortions in the frequency domain at multiples of the frame rate (figure 30).

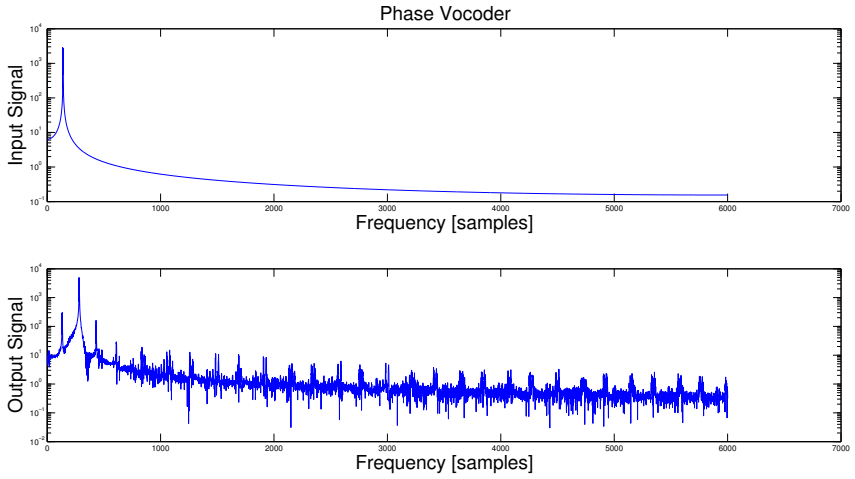


Figure 30: Time-Stretching using the Phase Vocoder - Frequency Domain

C.3 Phase-locked Vocoder

The improved vocoder naturally delivers a better result [Puc95, Sal07, Roe03]. Since relative phase values are kept, the frequency bins at frame transitions are phase-modulated, but the temporal envelope is more or less preserved (figure 31). The spectral distortion is negligible as compared to that of the standard phase-vocoder, but, as expected, the signal spreads over the whole spectrum due to the modulation introduced (figure 32). Primarily harmonic signals are affected by this kind of corruption.

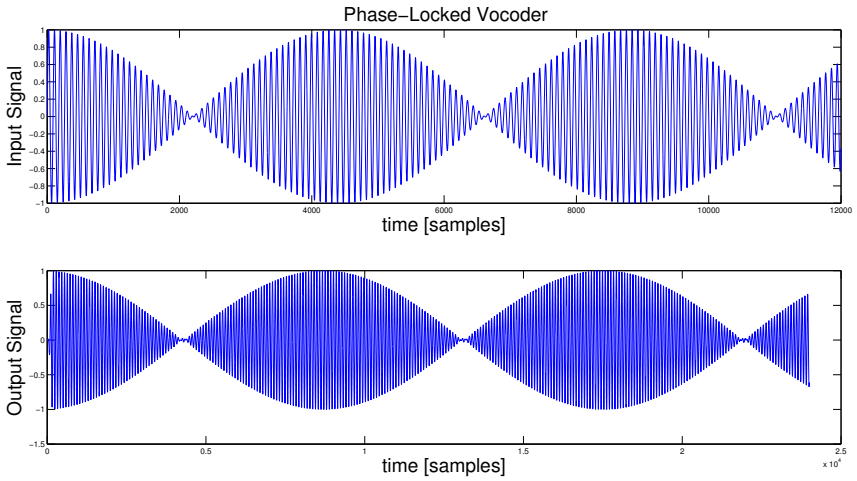


Figure 31: Time-Stretching using the Phase Locked Vocoder - Time Domain

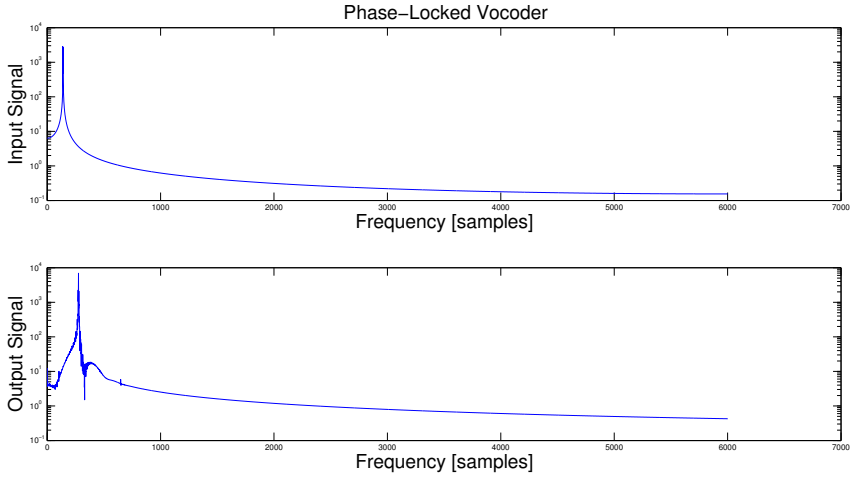


Figure 32: Time-Stretching using the Phase Locked Vocoder - Frequency Domain

C.4 Overlapp-Add Method

Since phase aligning in the frequency domain does not deliver completely satisfying results, for completeness the third method presented operates in the time-domain only. Either, the respective S+N piece is cut, or it is looped by the desired factor. At the transition points from one loop to another, the signal's envelope is smoothed and correlation is maximized in order to prevent sharp steps. This approaches solution is a compromise between instantaneous phase preservation and phase unwrapping, the signal's time-domain shape is not preserved as well as in the locked-vocoder case (figure

33) but is neither as severely corrupted as the standard vocoder would do.

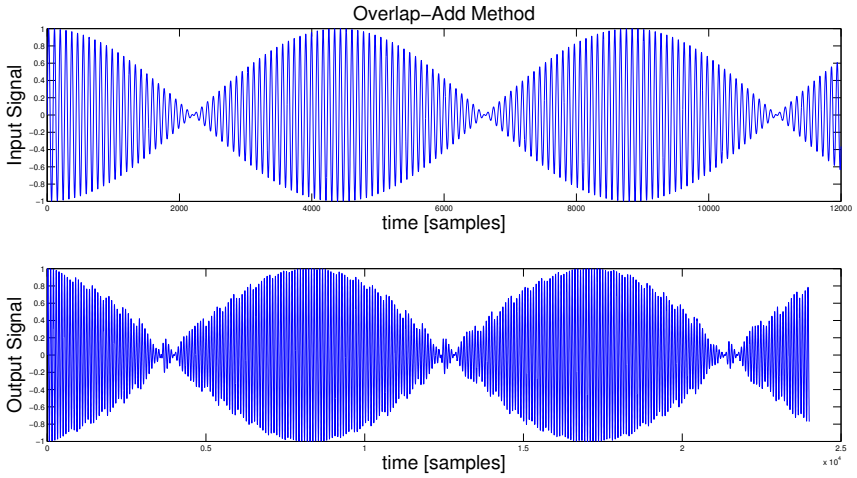


Figure 33: Time-Stretching using the Overlap-Add Method - Time Domain

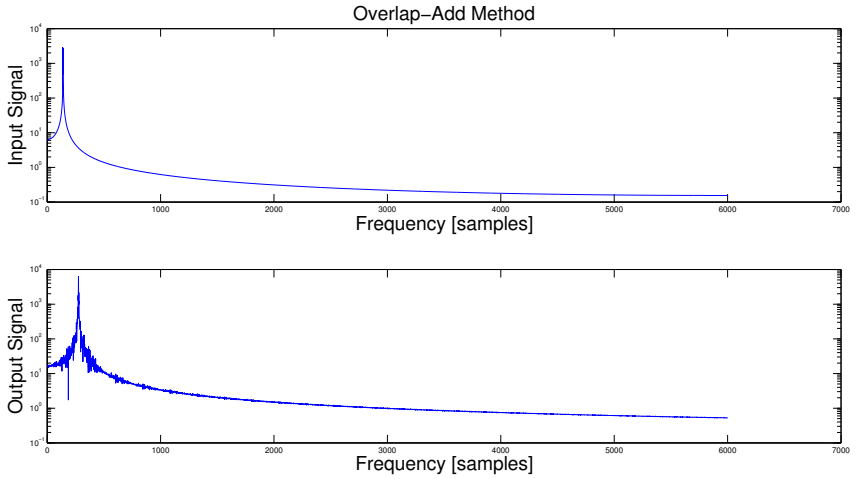


Figure 34: Time-Stretching using the Overlap-Add Method - Frequency Domain

Since the signal under investigation did not contain any transients, the magnitude delivered by the algorithm is quite smooth (figure 34). In general, the algorithm should be adjusted in order to not affect transients, otherwise the consequences both in the time and in the frequency domain might become unpredictable.

For speech signals however, the latter (possibly modified or extended [QH95]) is often the method of choice, since formant shapes get not affected by this approach. For music however, artifacts caused by signal cutting and looping are usually perceptible.

Apart from the fact, that signal reconstruction with those three approaches is far from perfect, none of them can be trusted to deliver reliable results in any constellation and situation. The methods introduced in the next chapter approach phase estimation from a more elaborated point of view.