A Non-iterative Method for (Re)Construction of Phase from STFT Magnitude

Zdeněk Průša, Peter Balazs, *Senior Member, IEEE,* and Peter L. Søndergaard

***Abstract*—A non-iterative method for the construction of the Short-Time Fourier Transform (STFT) phase from the magnitude is presented. The method is based on the direct relationship between the partial derivatives of the phase and the logarithm of the magnitude of continuous STFT with respect to the Gaussian window. Although the theory holds in the continuous setting only, the experiments show that the algorithm performs well even in the discretized setting (Discrete Gabor transform) with low redundancy using the sampled Gaussian window, the truncated Gaussian window and even other compactly supported windows like the Hann window.**

Due to the non-iterative nature, the algorithm is very fast and it is suitable for long audio signals. Moreover, solutions of iterative phase reconstruction algorithms can be improved considerably by initializing them with the phase estimate provided by the present algorithm.

We present an extensive comparison with the state-of-the-art algorithms in a reproducible manner.

***Index Terms*—STFT, Gabor transform, Phase reconstruction, Gradient theorem, Numerical integration**

I. Introduction

The phase retrieval problem has been actively investigated for decades. It was first formulated for the Fourier transform and later for generic linear systems.

In this paper, we consider a particular case of the phase retrieval problem; the reconstruction from the magnitude of the Gabor transform coefficients obtained by sampling the STFT magnitude at discrete time and frequency points [1]. The need for an effective way of the phase (re)construction arises in audio processing applications such as source separation and denoising [2], [3], time-stretching/pitch shifting [4], channel mixing [5], and missing data inpainting [6].

The problem has already been addressed by many authors. Among the iterative algorithms, the most widespread and influential is the Griffin-Lim algorithm (GLA) [7] which inspired several extensions [8], [9], [10]. See [11] for a detailed overview of the algorithms based on the idea of Griffin and Lim. A different approach was taken in [12], where the authors proposed to express the problem as an unconstrained optimization problem and to solve it using the limited memory Broyden-Flatcher-Goldfarb-Shanno algorithm. It is again an iterative algorithm and the computational cost of a single iteration is comparable to that of GLA.

Other approaches are based on reformulating the task as a convex problem [13], [14], [15], [16]. The dimension of the problem however squares, which makes it unsuitable for long audio signals which typically consist of tens of thousands of samples per second.

The approach from the recently published work [17] builds upon the assumption that the signal is sparse in the original domain, which is not realistic in the context of the audio processing applications mentioned above.

An interesting approach was presented in [18]. It is based on solving non-linear system of equations for each time frame. The authors proposed to use iterative solver and initialize it with samples obtained from previous frames. The algorithm is, however, designed to work exclusively with a rectangular window.

The common problem of the iterative state-of-the-art algorithms is that they require many relatively expensive iterations in order to produce acceptable results. Recently, a non-iterative algorithm was proposed in [19]. It is based on the notion of *phase consistency* used in the phase vocoder [4]. Although the algorithm is simple, fast and it is directly suitable for the real-time setting, it relies on the fact that the signal consists of slowly varying sinusoidal components and fails for transients and broadband components in general. A similar algorithm was introduced in [20], which, in addition, tries to treat impulse-like components separately.

In this paper, we propose another non-iterative algorithm (Phase Gradient Heap Integration – PGHI). The theory behind PGHI has been known at least since 1979. Indeed, it is based on the relationship between gradients of the Gaussian window-based STFT phase and log-magnitude published already in [21] and on the *gradient theorem*. More precisely, the phase gradient can be expressed using the STFT magnitude and the gradient theorem gives a prescription how to integrate the phase gradient field to recover the phase up to a global phase shift (or up to sign ambiguity in case of real signals). To our knowledge, no such algorithm has ever been published yet. Curiously enough, in [22] it was even explicitly discouraged to use such algebraic results for practical purposes.

The aforementioned algorithms [19] and [20] are in fact very close to the PGHI algorithm since they basically perform a crude integration of the estimate of *instantaneous frequency* and of the *local group delay* in case of [20], which are components of the STFT phase gradient.

In the spirit of reproducible research, the implementation of the algorithms, audio examples, color version of the figures as

Manuscript received April 19, 2005; revised August 26, 2015.

Z. Průša* and P. Balazs are with the Acoustics Research Institute, Austrian Academy of Sciences, Wohllebengasse 12–14, 1040 Vienna, Austria, email: zdenek.prusa@oeaw.ac.at (corresponding address), peter.balazs@oeaw.ac.at

P. L. Søndergaard is with Oticon A/S, Kongebakken 9, 2765 Smørum, Denmark, email: peter@sonderport.dk

This work was supported by the Austrian Science Fund (FWF) START-project FLAME ("Frames and Linear Operators for Acoustical Modeling and Parameter Estimation"; Y 551-N13).

well as scripts reproducing experiments from this manuscript are freely available at http://ltfat.github.io/notes/040. The code depends on our Matlab/GNU Octave[23] packages LTFAT [24], [25] (version 2.1.2 or above) and PHASERET (version 0.1.0 or above). Both toolboxes can be obtained freely from http://ltfat.github.io and http://ltfat.github.io/phaseret, respectively.

The paper is organized as follows. Section II summarizes the necessary theory of the STFT and the Gabor analysis, Section III presents the theory behind the proposed algorithm, Section IV contains a detailed description of the numerical algorithm. Finally, in Section V we present an extensive evaluation of the proposed algorithm and comparison with the iterative and non-iterative state-of-the-art algorithms using the Gaussian window, the truncated Gaussian window, the Hann and the Hamming windows.

II. Gabor analysis

The *short-time Fourier transform* of a function $f \in L^2(\mathbb{R})$ with respect to a window $g \in L^2(\mathbb{R})$ can be defined as[1]

$$(\mathcal{V}_g f)(\omega, t) = \int_{\mathbb{R}} f(\tau)\overline{g(\tau - t)}\mathrm{e}^{-\mathrm{i}2\pi\omega(\tau-t)}\,\mathrm{d}\tau, \quad \omega, t \in \mathbb{R} \quad (1)$$

$$=: M_g^f(\omega, t) \cdot \mathrm{e}^{\mathrm{i}\Phi_g^f(\omega,t)}. \quad (2)$$

Using the modulation $(\mathcal{E}_\omega f)(\tau) := \mathrm{e}^{\mathrm{i}2\pi\omega\tau} \cdot f(\tau)$ and translation $(\mathcal{T}_t f)(\tau) := f(\tau - t)$ we get the alternative $\left(\mathcal{V}_g f\right)(\omega, t) = \langle f, \mathcal{T}_t \mathcal{E}_\omega g\rangle$.

The (complex) logarithm of the STFT can be written as

$$\log(\mathcal{V}_g f)(\omega, t) = \log M_g^f(\omega, t) + \mathrm{i}\Phi_g^f(\omega, t). \quad (3)$$

The *Gaussian function* is a particularly suitable window function as it possesses optimal time-frequency properties and it allows an algebraic treatment of the equations. It is defined by the following formula

$$\varphi_\lambda(t) = \left(\frac{\lambda}{2}\right)^{-\frac{1}{4}} \mathrm{e}^{-\pi\frac{t^2}{\lambda}} = \left(D_{\sqrt{\lambda}}\varphi_1\right)(t), \quad (4)$$

where $\lambda \in \mathbb{R}^+$ denotes the "width" or the time-frequency ratio of the Gaussian window and D_α is a dilation operator such that $(D_\alpha f)(t) = \frac{1}{\sqrt{|\alpha|}} f(t/\alpha)$, $\alpha \neq 0$. We will use the shortened notation $\varphi(t) = \varphi_1(t)$ in the following text.

A. Discrete Gabor Transform

We define the Discrete Gabor Transform (DGT) of a signal $f \in \mathbb{C}^L$ with respect to a window $g \in \mathbb{C}^L$ as [26], [27]

$$c(m, n) = \sum_{l=0}^{L-1} f(l)\overline{g(l - na)}\mathrm{e}^{-\mathrm{i}2\pi m(l-na)/M} \quad (5)$$

$$=: s(m, n) \cdot \mathrm{e}^{\mathrm{i}\phi(m,n)} \quad (6)$$

for $m = 0, \ldots, M-1$ and $n = 0, \ldots, N-1$, where $M = L/b$ is the number of frequency channels, $N = L/a$ number of time shifts, a is the length of the time shift or a hop size in samples in time and b is a hop size in samples in frequency. The bar denotes complex conjugation and $(l - na)$ is assumed to be evaluated modulo L. The redundancy of the DGT is defined as $MN/L = M/a$. In the matrix notation, we can write $c = F_g^* f$, where F_g^* is a $MN \times L$ matrix.

The DGT can be seen as sampling of STFT (both of the arguments ω and t and the involved functions f and g themselves) of one period of L-periodic continuous signal f such that

$$c(m, n) = \left(\mathcal{V}_g f\right)(bm, an) + \mathcal{A}(m, n), \quad (7)$$

for $m = 0, \ldots, M-1$, $n = 0, \ldots, N-1$ where $\mathcal{A}(m, n)$ models both the aliasing and numerical errors introduced by the sampling.

For real signals $f \in \mathbb{R}^L$, the range of m can be shrunken to the first $\lfloor M/2 \rfloor + 1$ values as the remaining coefficients are complex conjugated. Moreover, coefficients $c(0, \bullet)$ are always real and so are $c(M/2, \bullet)$ if M is even.

Signal f can be recovered (up to a numerical precision error) using the following formula

$$f(l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} c(m, n)\widetilde{g}(l - na)\mathrm{e}^{\mathrm{i}2\pi m(l-na)/M} \quad (8)$$

for $l = 0, \ldots, L-1$, where $\widetilde{g}$ is the canonical dual window.

Again, in the matrix notation, we can write $f = F_{\widetilde{g}}c$, where $F_{\widetilde{g}}$ is a $L \times MN$ matrix. The canonical dual window can be obtained as

$$\widetilde{g} = \left(F_g F_g^*\right)^{-1} g. \quad (9)$$

See e.g. [27] for conditions under which the product $F_g F_g^*$ is (easily) invertible and [28] for an overview of efficient algorithms for computing (5), (8) and (9).

The discretized and periodized Gaussian window is given by

$$\varphi_\lambda^{\mathrm{D}}(l) = \left(\frac{\lambda L}{2}\right)^{-\frac{1}{4}} \sum_{k\in\mathbb{Z}} \mathrm{e}^{-\pi\frac{(l+kL)^2}{\lambda L}}, \quad (10)$$

for $l = 0, \ldots, L-1$. We assume that L and λ are chosen such that the time aliasing is numerically negligible and therefore it is sufficient to sum over $k \in \{-1, 0\}$ in practice.

The width of the Gaussian window at its relative height $h \in [0, 1]$ is given by

$$w_h = \sqrt{-\frac{4\log(h)}{\pi}\lambda L}. \quad (11)$$

The width is given in samples and it can be a non-integer number.

Note that all windows used in this manuscript are odd symmetric, such that they have just one center sample, and they are non-causal such that they introduce no delay. Finally, the discrete Fourier transform of such windows is real.

III. Theory Behind The Algorithm

The algorithm is based on the direct relationship between the partial derivatives of the phase and the log-magnitude of the STFT with respect to the Gaussian window. In this section, we derive such relations. We include a complete derivation

[1]In the literature, two other STFT phase conventions can be found. The present one is most common in the engineering community.

since the relations for STFT defined as (1) has not appeared in the literature, as far as we know.

It is known that the Bargmann transform of $f \in L^2(\mathbb{R})$

$$(\mathcal{B}f)(z) = 2^{\frac{1}{4}} \int_{\mathbb{R}} f(\tau) \mathrm{e}^{2\pi\tau z - \pi\tau^2 - \frac{\pi}{2}z^2} \mathrm{d}\tau, \quad z \in \mathbb{C} \tag{12}$$

is an entire function [29] for all $z \in \mathbb{C}$ and that it relates to the STFT defined in (1) such that [1]

$$(\mathcal{B}f)(z) = \mathrm{e}^{\pi i t\omega + \pi\frac{|z|^2}{2}} (\mathcal{V}_\varphi f)(-\omega, t), \tag{13}$$

assuming $z = t + \mathrm{i}\omega$. The logarithm of the Bargmann transform is an entire function as well (apart from zeros). The real and imaginary parts of $\log(\mathcal{B}f)(z)$ can be written as

$$\log(\mathcal{B}f)(t + \mathrm{i}\omega) = u(\omega, t) + \mathrm{i}v(\omega, t) \tag{14}$$

$$u(\omega, t) = \pi(t^2 + \omega^2)/2 + \log M_\varphi^f(-\omega, t) \tag{15}$$

$$v(\omega, t) = \pi t\omega + \Phi_\varphi^f(-\omega, t) \tag{16}$$

and using the Cauchy-Riemann equations

$$\tfrac{\partial u}{\partial t}(\omega, t) = \tfrac{\partial v}{\partial \omega}(\omega, t) \tag{17}$$

$$\tfrac{\partial u}{\partial \omega}(\omega, t) = -\tfrac{\partial v}{\partial t}(\omega, t) \tag{18}$$

we can write (substituting $\omega' = -\omega$) that

$$\tfrac{\partial}{\partial \omega'} \Phi_\varphi^f(\omega', t) = -\tfrac{\partial}{\partial t} \log M_\varphi^f(\omega', t) \tag{19}$$

$$\tfrac{\partial}{\partial t} \Phi_\varphi^f(\omega', t) = \tfrac{\partial}{\partial \omega'} \log M_\varphi^f(\omega', t) + 2\pi\omega'. \tag{20}$$

A little more general relationships can be obtained for windows defined as $g = \mathcal{O}\varphi_1$ ($\mathcal{O}$ being a fixed bounded operator) and Proposition 1.

Proposition 1. *Let $\mathcal{O}, \mathcal{P}$ be bounded operators such that for all (ω, t) there exist differentiable, one-to-one functions $\eta(t)$ and $\xi(\omega)$, such that $\mathcal{T}_t\mathcal{E}_\omega\mathcal{O} = \mathcal{P}\mathcal{T}_{\eta(t)}\mathcal{E}_{\xi(\omega)}$ and let $g = \mathcal{O}\varphi_1$. Then*

$$\tfrac{\partial}{\partial \omega} \Phi_g^f(\omega, t) = -\tfrac{\partial}{\partial t} \log M_g^f(\omega, t) \cdot \frac{\xi'(\omega)}{\eta'(t)} \tag{21}$$

$$\tfrac{\partial}{\partial t} \Phi_g^f(\omega, t) = \tfrac{\partial}{\partial \omega} \log M_g^f(\omega, t) \cdot \frac{\eta'(t)}{\xi'(\omega)} + 2\pi\xi(\omega)\eta'(t). \tag{22}$$

Proof. Consider

$$\begin{aligned}(\mathcal{V}_g f)(\omega, t) &= \langle f, \mathcal{T}_t\mathcal{E}_\omega g\rangle = \langle f, \mathcal{T}_t\mathcal{E}_\omega\mathcal{O}\varphi_1\rangle \\ &= \left\langle \mathcal{P}^* f, \mathcal{T}_{\eta(t)}\mathcal{E}_{\xi(\omega)}\varphi_1 \right\rangle \\ &= \left(\mathcal{V}_{\varphi_1}(\mathcal{P}^* f)\right)(\xi(\omega), \eta(t))\end{aligned}$$

Therefore

$$\begin{aligned}\tfrac{\partial}{\partial t} \Phi_g^f(\omega, t) &= \tfrac{\partial}{\partial t} \Phi_{\varphi_1}^{\mathcal{P}^* f}(\xi(\omega), \eta(t)) \\ &= \left[\tfrac{\partial}{\partial \eta} \Phi_{\varphi_1}^{\mathcal{P}^* f}(\xi(\omega), \eta(t))\right] \cdot \eta'(t)\end{aligned}$$

and

$$\tfrac{\partial}{\partial \omega} \Phi_g^f(\omega, t) = \left[\tfrac{\partial}{\partial \xi} \Phi_{\varphi_1}^{\mathcal{P}^* f}(\xi(\omega), \eta(t))\right] \cdot \xi'(\omega).$$

Furthermore

$$\tfrac{\partial}{\partial t} \log M_g^f(\omega, t) = \left[\tfrac{\partial}{\partial \eta} \log M_{\varphi_1}^{\mathcal{P}^* f}(\xi(\omega), \eta(t))\right] \cdot \eta'(t)$$

and

$$\tfrac{\partial}{\partial \omega} \log M_g^f(\omega, t) = \left[\tfrac{\partial}{\partial \xi} \log M_{\varphi_1}^{\mathcal{P}^* f}(\xi(\omega), \eta(t))\right] \cdot \xi'(\omega).$$

Combining this with (20) we get

$$\begin{aligned}\tfrac{\partial}{\partial t} \Phi_g^f(\omega, t) &= \left[\tfrac{\partial}{\partial \eta} \Phi_{\varphi_1}^{\mathcal{P}^* f}(\xi(\omega), \eta(t))\right] \cdot \eta'(t) \\ &= \left[\tfrac{\partial}{\partial \xi} \log M_{\varphi_1}^{\mathcal{P}^* f}(\xi(\omega), \eta(t)) + 2\pi\xi(\omega)\right] \cdot \eta'(t) \\ &= \tfrac{\partial}{\partial \omega} \log M_g^f(\omega, t) \cdot \frac{\eta'(t)}{\xi'(\omega)} + 2\pi\xi(\omega)\eta'(t).\end{aligned}$$

The other equality can be shown using the same arguments and (19). □

Choosing $\mathcal{O} = D_{\sqrt{\lambda}}$, $\xi(\omega) = \sqrt{\lambda}\omega$ and $\eta(t) = t/\sqrt{\lambda}$ leads to equations for dilated Gaussian window φ_λ

$$\tfrac{\partial}{\partial \omega} \Phi_{\varphi_\lambda}^f(\omega, t) = -\lambda \tfrac{\partial}{\partial t} \log M_{\varphi_\lambda}^f(\omega, t) \tag{23}$$

$$\tfrac{\partial}{\partial t} \Phi_{\varphi_\lambda}^f(\omega, t) = \frac{1}{\lambda} \tfrac{\partial}{\partial \omega} \log M_{\varphi_\lambda}^f(\omega, t) + 2\pi\omega. \tag{24}$$

The relations were already published in [21], [30], [31] in slightly different forms obtained using different techniques than we use here. The equations differ because the authors of the above mentioned papers use different STFT phase conventions.

It should be noted that the relations for general windows were already studied in [32], they however involve partial derivatives of the logarithm of the modified Bargmann transform and thus it seems they cannot be exploited directly. Moreover, the experiments presented in Section V show that the performance degradation is not too significant when using windows resembling the Gaussian window.

The STFT phase gradient of a signal f with respect to dilated Gaussian φ_λ will be further denoted as

$$\nabla\Phi_{\varphi_\lambda}^f(\omega, t) = \left[\tfrac{\partial}{\partial \omega} \Phi_{\varphi_\lambda}^f(\omega, t), \tfrac{\partial}{\partial t} \Phi_{\varphi_\lambda}^f(\omega, t)\right]. \tag{25}$$

A. *Gradient Integration and the Phase Shift Phenomenon*

Knowing the phase gradient, one can exploit the gradient theorem (see e.g. [33]) to recover the original phase $\Phi_{\varphi_\lambda}^f(\omega, t)$ such that

$$\Phi_{\varphi_\lambda}^f(\omega, t) - \Phi_{\varphi_\lambda}^f(\omega_0, t_0) = \int_0^1 \nabla\Phi_{\varphi_\lambda}^f(r(\tau)) \cdot \frac{\mathrm{d}r}{\mathrm{d}\tau}(\tau)\, \mathrm{d}\tau, \tag{26}$$

where $r(\tau) = [r_\omega(\tau), r_t(\tau)]$ is any curve starting at (ω_0, t_0) and ending at (ω, t) provided the phase at the initial point (ω_0, t_0) is known. When the phase is unknown completely, we consider $\Phi_{\varphi_\lambda}^f(\omega_0, t_0) = 0$ and therefore the phase one obtains by (26) is

$$\widetilde{\Phi}_{\varphi_\lambda}^f(\omega, t) = \Phi_{\varphi_\lambda}^f(\omega, t) + \Phi_0, \tag{27}$$

where $\Phi_0 = \Phi_{\varphi_\lambda}^f(\omega_0, t_0)$ is a constant global phase shift.

The global phase shift of the STFT carries over to the global phase shift of the reconstructed signal trough the linearity of the reconstruction. One must however treat real input signals with care as the phase shift breaks the complex conjugate relation of the positive and negative frequency coefficients. Therefore if one simply takes only the real part of the reconstructed signal the phase shift causes its amplitude attenuation or even causes the signal to vanish in the extreme case. To explain this phenomenon, consider the following example where we compare the effect of the phase shift on analytic and on real signals. We denote the constant phase shift as ψ_0 and define an analytic signal as $x_{\text{an}}(t) = A(t)\mathrm{e}^{\mathrm{i}\psi(t)}$. The real part including the global phase shift ($\mathrm{e}^{\mathrm{i}\psi_0}$) is given as $\mathcal{R}(x_{\text{an}}(t)\mathrm{e}^{\mathrm{i}\psi_0}) = A(t)\cos(\psi(t) + \psi_0)$ which is what one would expect. Similarly, we define a real signal as $x(t) = \frac{A(t)}{2}\left(\mathrm{e}^{\mathrm{i}\psi(t)} + e^{-\mathrm{i}\psi(t)}\right)$ and the real part of such signal with the global phase shift ψ_0 amounts to $\mathcal{R}(x(t)e^{\mathrm{i}\psi_0}) = A(t)\cos(\psi_0)\cos(\psi(t))$ which causes the signal to vanish when $\psi_0 = \pi/2 + k\pi$, $k \in \mathbb{Z}$.

In theory, the global phase shift of the STFT of a real signal can be compensated for, leaving only a global signal sign ambiguity. For real signals, it is clear that the following holds for $\omega \neq 0$

$$\widetilde{\Phi}^f_{\varphi_\lambda}(\omega, t) + \widetilde{\Phi}^f_{\varphi_\lambda}(-\omega, t) = 2\Phi_0. \tag{28}$$

Due to the phase wrapping, after the compensation, the phase shift is still ambiguous such that $\Phi_0 = k\pi$ for $k \in \mathbb{Z}$, which causes the aforementioned signal sign ambiguity.

IV. The Algorithm

In the discrete time setting (recall Section II-A; in particular (5) and (6)) the STFT phase gradient approximation $\widehat{\nabla\Phi_{\varphi_\lambda}}(bm, an) := \nabla\phi(m, n)$ is obtained by numerical differentiation of $s_{\log}(m, n) := \log\big(s(m, n)\big)$ as

$$\nabla\phi(m, n) = \big[\phi_\omega(m, n), \phi_t(m, n)\big] := \tag{29}$$

$$\left[-\frac{\lambda}{a}(s_{\log}D_t^T)(m, n), \frac{1}{\lambda b}(D_\omega s_{\log})(m, n) + 2\pi m/M\right] \tag{30}$$

where D_t^T, D_ω denote matrices performing the numerical differentiation of $s_{\log}$ along rows (in time) and columns (in frequency) respectively. The matrices are assumed to be scaled such that the sampling step of the differentiation scheme they represent is equal to 1. The central finite difference scheme is the most suitable because it ensures the gradient components to be sampled at the same grid.

The steps of the numerical integration will be done in either horizontal or vertical directions such that exclusivelly one of the components in $\frac{\mathrm{d}r}{\mathrm{d}\tau}$ from (26) is zero. Due to this property, the gradient can be pre-scaled using lengths of the steps (hop sizes a and b) such that

$$\nabla\phi^{\text{SC}}(m, n) := \big[b\phi_\omega(m, n), a\phi_t(m, n)\big] = \tag{31}$$

$$\left[-\frac{\lambda L}{aM}(s_{\log}D_t^T)(m, n), \frac{aM}{\lambda L}(D_\omega s_{\log})(m, n) + 2\pi am/M\right]. \tag{32}$$

Note that the dependency on L can be avoided when (11) is used to express λL.

The numerical gradient line integration is performed adaptively using the simple trapezoidal rule. The algorithm makes use of a heap data structure (from the heapsort algorithm [34]). In case of the present algorithm it is used for holding pairs (m, n) and it has the property of having (m, n) of the maximum $|c(m, n)|$ always at the top. It is further equipped with efficient operations for insertion and deletion. The effect of the parameter tol is twofold. First, a random phase (uniformly distributed random values from the range $[0, 2\pi]$) is assigned to coefficients small in magnitude for which the phase gradient is unreliable and second, the integration is done only locally on “islands” with the max coefficient within the island serving as the zero phase reference. The randomization of the phase of the coefficients below the tolerance is chosen over the zero phase because in practice it helps to avoid the impulsive disturbances introduced by the small phase-aligned coefficients. The algorithm is summarized in Alg. 1.

After $\widehat{\phi}(m, n)$ has been estimated by Alg. 1, it is combined with the target magnitude of the coefficients such that

$$\widehat{c}(m, n) = s(m, n)\mathrm{e}^{\mathrm{i}\widehat{\phi}(m, n)} \tag{33}$$

and the signal is recovered by simply plugging these coefficients into (8).

A. Practical Considerations

In this section, we analyze the effect of the discretization on the performance of the algorithm.

The obvious sources of error are the numerical differentiation and integration schemes. However, the aliasing introduced by subsampling in time and frequency domains is more serious. In the discrete time setting, since the signal is considered to be band-limited and periodic, the truly aliasing-free case occurs when $a = 1, b = 1$ ($M = L, N = L$) regardless of the time or the frequency effective supports of the window. DGT with such setting is however highly redundant and only signals up to several thousands samples in length can be handled effectively.

In the subsampled case, the amount of aliasing and therefore the performance of the algorithm depends on the effective support of the window. Increasing a introduces aliasing in frequency and increasing b introduces aliasing in time.

This property is illustrated by Figure 1, which shows that the algorithm performs very well in the aliasing-free case ($a = 1, b = 1$) and the performance becomes worse when longer hop sizes in time are introduced while keeping the effective width of the window constant. The length of the signal is 5888 samples and the time-frequency ratio of the Gaussian window is $\lambda = 1$. The hop size in frequency is $b = 1$ (i.e. $M = 5888$). Only the values for the 60 dB range of the highest coefficients are shown.

Even though the Gaussian window is in theory infinitely supported in both time and frequency, it decays exponentially and therefore aliasing might not significantly degrade the performance of the algorithm when choosing the hop sizes and the effective support carefully. Obviously the finer the hop sizes the higher the computational cost. Settings used in Section V seem to be a good compromise.

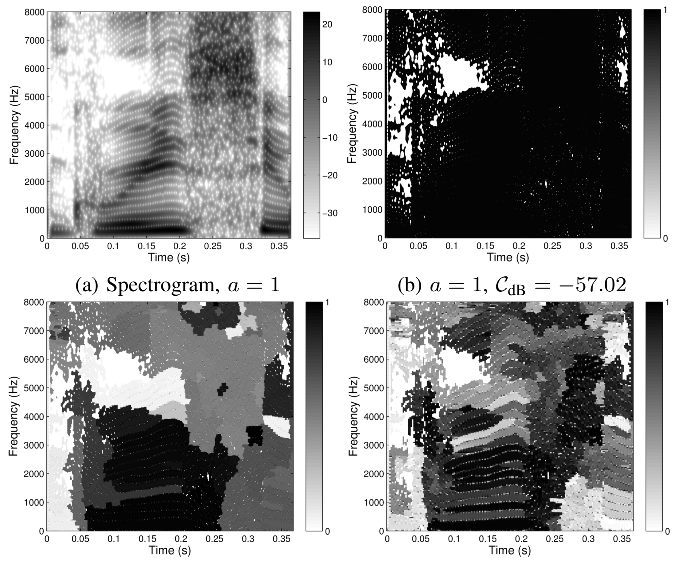
Since it is clear that the phase shift achieved by the algorithm is not constant, the conjugate symmetry of the DGT of real signals cannot be easily recovered. Therefore, when dealing with real signals, we reconstruct the phase only for the positive frequency coefficients and enforce the conjugate symmetry to the negative frequency coefficients.

Algorithm 1: Phase gradient heap integration – PGHI

```
Input: DGT phase gradient
       $\nabla\phi^{\mathrm{SC}}(m,n) = \left(\phi_\omega^{\mathrm{SC}}(m,n), \phi_t^{\mathrm{SC}}(m,n)\right)$ obtained
       from (32), magnitude of DGT coefficients
       $\left|c(m,n)\right|$, relative tolerance $tol$.
Output: Estimate of the DGT phase $\widehat{\phi}(m,n)$.
1  Set $\mathcal{I} = \left\{(m,n) : \left|c(m,n)\right| > tol \cdot \max\left(\left|c(m,n)\right|\right)\right\}$;
2  Assign random values to $\widehat{\phi}(m,n)_{(m,n)\notin\mathcal{I}}$;
3  Construct a self-sorting heap for $(m,n)$ pairs;
4  while $\mathcal{I}$ is not $\emptyset$ do
5      if heap is empty then
6          Insert $(m,n)_{\max} = \arg\max\left(\left|c(m,n)_{(m,n)\in\mathcal{I}}\right|\right)$
           into the heap;
7          $\widehat{\phi}(m,n)_{\max} \leftarrow 0$;
8          Remove $(m,n)_{\max}$ from $\mathcal{I}$;
9      end
10     while heap is not empty do
11         $(m,n) \leftarrow$ remove the top of the heap;
12         if $(m+1,n) \in \mathcal{I}$ then
13             $\widehat{\phi}(m+1,n) \leftarrow$
                 $\widehat{\phi}(m,n) + \frac{1}{2}\left(\phi_\omega^{\mathrm{SC}}(m,n) + \phi_\omega^{\mathrm{SC}}(m+1,n)\right)$;
14             Insert $(m+1,n)$ into the heap;
15             Remove $(m+1,n)$ from $\mathcal{I}$;
16         end
17         if $(m-1,n) \in \mathcal{I}$ then
18             $\widehat{\phi}(m-1,n) \leftarrow$
                 $\widehat{\phi}(m,n) - \frac{1}{2}\left(\phi_\omega^{\mathrm{SC}}(m,n) + \phi_\omega^{\mathrm{SC}}(m-1,n)\right)$;
19             Insert $(m-1,n)$ into the heap;
20             Remove $(m-1,n)$ from $\mathcal{I}$;
21         end
22         if $(m,n+1) \in \mathcal{I}$ then
23             $\widehat{\phi}(m,n+1) \leftarrow$
                 $\widehat{\phi}(m,n) + \frac{1}{2}\left(\phi_t^{\mathrm{SC}}(m,n) + \phi_t^{\mathrm{SC}}(m,n+1)\right)$;
24             Insert $(m,n+1)$ into the heap;
25             Remove $(m,n+1)$ from $\mathcal{I}$;
26         end
27         if $(m,n-1) \in \mathcal{I}$ then
28             $\widehat{\phi}(m,n-1) \leftarrow$
                 $\widehat{\phi}(m,n) - \frac{1}{2}\left(\phi_t^{\mathrm{SC}}(m,n) + \phi_t^{\mathrm{SC}}(m,n-1)\right)$;
29             Insert $(m,n-1)$ into the heap;
30             Remove $(m,n-1)$ from $\mathcal{I}$;
31         end
32     end
33 end
```

B. Exploiting Partially Known Phase

In some scenarios, true phase of some of the coefficients or regions of coefficients is available. In order to exploit such information, the proposed algorithm has to be adjusted slightly. First, we introduce a mask to select the reliable coefficients and second, we select the border coefficients i.e. coefficients with at least one neighbor in the time-frequency plane with unknown phase. Then we simply initialize the algorithm with the border coefficients stored in the heap.



(a) Spectrogram, $a = 1$ (b) $a = 1$, $\mathcal{C}_{\mathrm{dB}} = -57.02$

(c) $a = 16$, $\mathcal{C}_{\mathrm{dB}} = -28.17$ (d) $a = 32$, $\mathcal{C}_{\mathrm{dB}} = -24.06$

Fig. 1: Spectrogram of a spoken word *greasy* (a). The absolute phase differences of the STFT of the original and reconstructed signal in rad$/\pi$ (modulo 1) for varying time hop size a (b) (c) (d). The error $\mathcal{C}_{\mathrm{dB}}$ is introduced in Section V.

Formally, Algorithm 1 will be changed such that steps summarized in Algorithm 2 are inserted after line 3.

Algorithm 2: Initialization for partially known phase

```
Additional input: Set of indices of coefficients $\mathcal{M}$ with
                  known phase $\phi(m,n)_{(m,n)\in\mathcal{M}}$.
1  $\widehat{\phi}(m,n) \leftarrow \phi(m,n)$ for $(m,n) \in \mathcal{M}$;
2  for $(m,n) \in \mathcal{M} \cap \mathcal{I}$ do
3      if $(m+1,n) \notin \mathcal{M}$ or $(m-1,n) \notin \mathcal{M}$ or
       $(m,n+1) \notin \mathcal{M}$ or $(m,n-1) \notin \mathcal{M}$ then
4          Add $(m,n)$ to the heap;
5      end
6  end
```

Note that the phase of the border coefficients can be used directly (i.e. no unwrapping is necessary). Depending on the situation, the phase might be propagated from more than one border coefficient, however the phases coming from distinct sources are never combined.

C. Connections to Phase Vocoder

In this section we discuss some connections between the proposed algorithm and the phase vocoder [4] and consequently algorithms presented in [19] and [20].

The basic phase vocoder employs non-equal analysis and synthesis time hop sizes in order to change the signal duration. The pitch change can be achieved by playing the signal at sampling rate adjusted by the ratio of the analysis and synthesis hop sizes. In the synthesis, the phase must be kept *consistent* in order not to introduce artifacts. In the phase reconstruction task, the original phase is not available, but the basic phase behavior can be yet exploited. For example, it is known that for a sinusoidal component with a constant frequency the phase grows linearly in time for all frequency channels the component influences in the spectrogram. For these coefficients, the instantaneous frequency (STFT phase derivative with respect to time (24)) is constant and the local group delay (STFT phase derivative with respect to frequency (23)) is zero.

In the aforementioned papers [19] and [20], the instantaneous frequency is estimated in each spectrogram column (time frame) from the magnitude by peak picking and interpolation. The instantaneous frequency determines phase increments for each frequency channel m such that

$$\phi(m,n) = \phi(m,n-1) + 2\pi a m_0/M, \tag{34}$$

where m_0 is the estimated, possibly non-integer instantaneous frequency belonging to the interval $\left[0, \lfloor M/2 \rfloor\right]$. This is exactly what the proposed algorithm does in case of constant sinusoidal components, except the instantaneous frequency is determined from the DGT log-magnitude. Integration in Alg. 1 performs nothing else than a cumulative sum of the instantaneous frequency in the time direction.

The algorithm from [20] goes further and employs an impulse model. The situation is reciprocal to sinusoidal components such that the phase changes linearly in frequency for all coefficients belonging to an impulse component but the rate is only constant for fixed n and it is inversely proportional to the local group delay $n_0 - n$ such as

$$\phi(m,n) = \phi(m-1,n) + 2\pi a(n-n_0)/M, \tag{35}$$

where an_0 is the time instant of the impulse occurrence. Again, this is what the proposed algorithm does for coefficients corresponding to impulses.

The advantage of the proposed algorithm over the other two is that the phase gradient is computed from the DGT log-magnitude such that it is available at every time-frequency position without even analysing the spectrogram content. This allows an arbitrary integration path which combines both the instantaneous frequency and the local group delay according to the magnitude ridge orientation. In the other approaches, the phase time derivative can be only estimated in a vicinity of sinusoidal components and, vice versa, the frequency derivative only in a vicinity of impulse-like events. Obviously, such approaches will not cope well with deviations from the model assumptions although careful implementation can handle multiple sinusoidal components with slowly varying instantaneous frequencies and impulses with frequency-varying onsets. The difficulty of algorithm [20] lies in detecting the onsets in the spectrogram and separating the coefficients belonging to the impulse-like component from the coefficients belonging to sinusoidal components.

Figure 2 shows phase deviations achieved by algorithms from [19], [20] and by the proposed algorithm. The phase difference at the transient coefficients is somewhat smoother for Alg. [20] when compared to [19] because of the involved impulse model. The proposed algorithm produces almost constant phase difference due to the adaptive integration direction. The setup used in the example is the following: the length of the signal is $L = 8192$ samples, time hop size $a = 16$, number of channels $M = 2048$, time-frequency ratio of the Gaussian window is $\lambda = aM/L$. Only the values for the 50 dB range of the highest coefficients are shown.

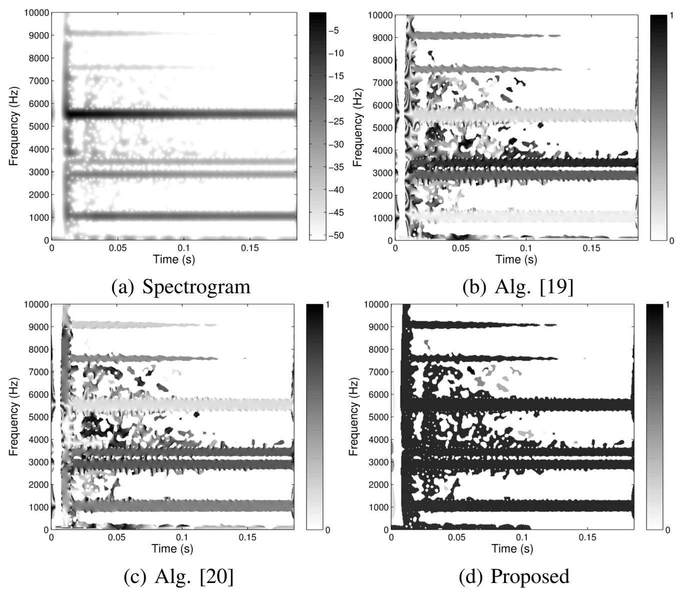


Fig. 2: Spectrogram of an excerpt form the *glockenspiel* signal and the absolute phase differences in rad/π (modulo 1) for three different algorithms.

V. Experiments

In the experiments, we use the following equation to measure the error

$$E(x,y) = \frac{\|x-y\|_{\text{fro}}}{\|x\|_{\text{fro}}},\ E_{\text{dB}}(x,y) = 20\log_{10} E(x,y), \tag{36}$$

where x, y are either vectors or matrices. In [11] the *spectral convergence* is defined as

$$\mathcal{C} = E\left(s, |P\widehat{c}|\right),\ \mathcal{C}_{\text{dB}}(x,y) = 20\log_{10}\mathcal{C}, \tag{37}$$

where $P = F_g^* F_{\widetilde{g}}$. In [9] the authors proposed a slightly different measure $E(\widehat{c}, P\widehat{c})^2$ called *normalised inconsistency measure* defining normalised energy lost by the reconstruction/projection. Such measures clearly do not accurately reflect the actual signal reconstruction error $E(f,\widehat{f})$, but they are independent of the phase shift. Some other papers evaluate the algorithms using the signal to noise ratio, which they define as $SNR(x,y) = 1/E(x,y)$ and $SNR_{\text{dB}}(x,y) = -E_{\text{dB}}(x,y)$ respectively.

Unfortunately, as the phase difference plots in Fig. 1 and Fig. 2 show, the phase difference is usually far from being constant when subsampling is involved (this holds for any algorithm, even the iterative ones). Therefore, the time-frames and even each frequency bin within the frame might have a different phase shift, causing the error $E(f,\widehat{f})$ to be very high, even when the other error measures are low and the actual perceived quality is good.

The testing was performed on the speech corpus database MOCHA-TIMIT [35] consisting of recordings of 1 male and 1 female English speakers each of which performing 460 sentences. The total duration of the recordings is 61 minutes and 5 seconds. The sampling rate of all recordings is 16 kHz. The Gabor system parameters used with this database (Table I and Fig. 3) were: number of channels $M = 1024$, hop size $a = 128$, time-frequency ratio of the Gaussian window $\lambda = aM/L$, time support of the truncated Gaussian window and the other compactly supported windows was M samples.

Next, we used the EBU SQAM database of 70 test sound samples [36] recorded at 44.1 kHz. Only the first 10 seconds of the first channel was used from the stereophonic recordings to reduce the execution time to a reasonable value. The Gabor system parameters used with this database (Table II and Fig. 4) were the following: number of channels $M = 2048$, hop size $a = 256$, time-frequency ratio of the Gaussian window $\lambda = aM/L$, time support of the truncated Gaussian window and of the other compactly supported windows was M samples.

A. Performance Of The Proposed Algorithm

In this section, we evaluate the performance of the proposed algorithm and compare it to results obtained by the Single Pass Spectrogram Inversion algorithm [19] (SPSI). Unfortunately, we were not able to get good results with the algorithm from [20] consistently due to the imperfect onset detection and due to the limitation of the impulse model.

The implementation of SPSI has been taken from http://anclab.org/software/phaserecon/ and it was modified to fit our framework. The most prominent change has been the removal of the alternating π and 0 phase modulation in the frequency direction which is not present when computing the transform according to (5).

The results for the proposed algorithm were computed via a two step procedure. In the first step Alg. 1 with $tol = 10^{-1}$ was used, and in the second step, the algorithm was run again with $tol = 10^{-10}$ initialized with the result from the previous step, according to Alg. 2. This approach avoids error spreading during the numerical integration and improves the result considerably when compared to a single run with either of the thresholds.

Tables I and II show the average $\mathcal{C}$ (converted to a value in dB) over whole databases for the SPSI and the proposed algorithm. The proposed algorithm clearly outperforms the SPSI algorithm by a large margin. The performance of the proposed algorithm further depends on the choice of the window. While the Gaussian window truncation introduces only a negligible performance degradation, the choice of Hann or Hamming windows increase the error by about 2 dB. For a detailed comparison, please find the scores and sound examples for the individual files from the EBU SQAM database using the Gaussian window at the accompanying web page http://ltfat.github.io/notes/040.

We can only provide a rough timing for the algorithms as the actual execution time is highly signal dependent and our implementations might be suboptimal. In the setup used in the tests, the runtime of the proposed algorithm is about 6–8 times longer than of the implementation of the SPSI algorithm. In particular, the current implementation of the proposed algorithm is very slow for noise signals.

TABLE I: Average $\mathcal{C}$ in dB for the MOCHA-TIMIT database

	Gauss	Trunc. Gauss	Hann	Hamming
SPSI	-16.75	-16.75	-14.53	-14.02
PGHI (proposed)	$\mathbf{-27.36}$	$\mathbf{-27.37}$	$\mathbf{-25.09}$	$\mathbf{-24.87}$

TABLE II: Average $\mathcal{C}$ in dB for the EBU SQAM database

	Gauss	Trunc. Gauss	Hann	Hamming
SPSI	-18.01	-18.19	-17.09	-16.79
PGHI (proposed)	$\mathbf{-24.79}$	$\mathbf{-24.71}$	$\mathbf{-23.14}$	$\mathbf{-22.66}$

B. Comparison With The State-of-the-art

We further compare the present algorithm with the following iterative algorithms:

- The Griffin-Lim algorithm [7] (GLA) as the baseline.
- A combination of Le Roux's modifications of GLA [9] and the fast version of GLA [10] with constant $\alpha = 0.99$ (FleGLA). More precisely from [9] we use the modification called *on-the-fly truncated modified update* which was reported to perform the best. The on-the-fly phase updates is performed in the natural order of frames starting with the zero frequency bin within each frame. The projection kernel was always truncated to size $2M/a - 1$ in both directions.
  This combination outperformes both algorithms [9] and [10] when used individually.
- The gradient descend-like algorithm from [12] (lBFGS) with the refined objective function ($p = 2/3$). Unfortunately, the lBFGS implementation we use (downloaded from [37]) fails in some cases.
- The real-time iterative spectrogram inversion algorithm with look-ahead (RTISI-LA). The algorithm was published in [8], but we implemented a refined version using the truncated projection kernel from [9] as proposed in [38]. The number of the look-ahead frames was always $M/a - 1$ and an asymmetric analysis window was used for the latest look-ahead frame. Due to the nature of the algorithm, its performance can only be evaluated at M/a multiples of per-frame iterations.

Since all the iterative algorithms optimize a non-convex objective function, the result depends strongly on the initial phase estimate. In addition to the zero phase initialization, we also evaluate performance of the algorithms initialized with the phase computed using the proposed algorithm. We will denote such initialization as *warm-start* (ws). Unfortunately, our tests showed that the RTISI-LA algorithm does not benefit from the warm-starting as it performs its own initial phase guess from the partially reconstructed signal.

Figures 3 and 4 show average $\mathcal{C}$ in dB over the MOCHA-TIMIT and EBU SQAM databases respectively depending on the number of iterations without (solid lines) or with (dashed lines) the warm start. The horizontal dashed line is the average

$\mathcal{C}$ in dB achieved by the proposed algorithm (values from Tables I and II). In addition, the scores and sound examples for individual files from the EBU SQAM database using the Gaussian window can be found at the accompanying web page.

Graphs for the truncated Gaussian window are not shown as they exhibit no visual difference from the graphs for the full-length Gaussian window. Further, the lBFGS algorithm has been excluded from the comparison using the EBU-SQAM database (Fig. 4); it failed to finish for a considerable number of the excerpts.

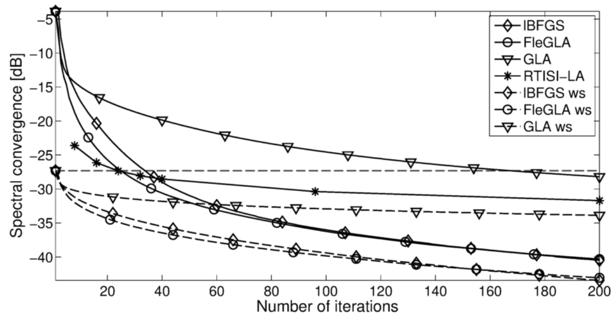
The graphs show that the proposed algorithm provides a suitable initial phase for the iterative algorithms i.e. the best overall results are obtained when combined with the FleGLA and lBFGS algorithms. When considering the iterative algorithms without warm-starting, the crossing points indicating number of iterations necessary to achieve the performance of the proposed algorithm can be clearly identified (if present). For the MOCHA TIMIT database (Fig. 3), the crossing point of the best algorithms is at about 20 iterations and the behavior is consistent for all the windows. The results for the EBU SQAM database (Fig. 4) are more erratic. First of all, the GLA algorithm never reaches the performance of the proposed algorithm even in 200 iterations. The crossing point of the FleGLA algorithm varies from 45 to 170 iterations depending on the window used. On the other hand, the RTISI-LA algorithm gets close to the line only for 8 per-frame iterations using the Gaussian window (Fig. 4a) and it performs better than the proposed algorithms in all the other cases. The RTISI-LA algorithm however "fails" for sound excerpts like castanets (crossing point at 80–120 iterations), drums, cymbals and glockenspiel (crossing points 20–40 iterations).

In the tests, the execution time of the proposed algorithm was comparable to the execution time of 2–4 iterations of the GLA algorithm with the Gaussian window and to the execution time of 4–10 iterations for the compactly supported windows.

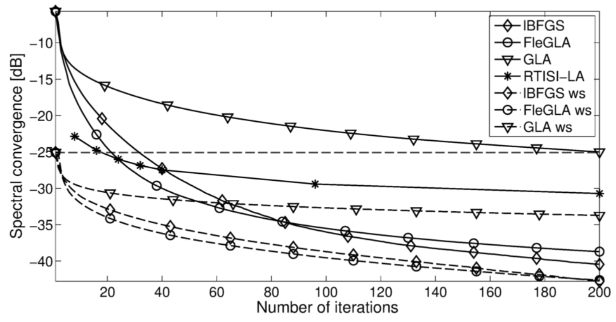
C. Modified Spectrograms

The main application area of the phase reconstruction algorithms is the reconstruction from the modified spectrograms. In general, the modified spectrogram is no longer a valid spectrogram (there is no signal having such spectrogram) and therefore the task is to construct rather than reconstruct a suitable phase. Unfortunately, it is not clear for which spectrogram modifications the equations (23) and (24) still hold nor how does it affect the performance if they do not. Moreover, an objective comparion of the algorithms becomes difficult as the error measures become irrelevant.

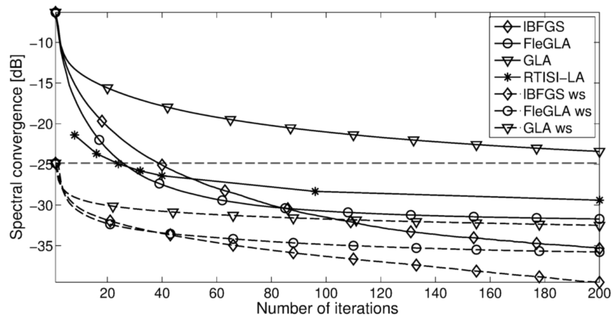
Nevertheless, in order to get the idea of the performance of the proposed algorithm acting on modified spectrograms, we implemented phase vocoder-like pitch shifting (up and down by 6 semitones) via changing the hop size ([4], [39]) using all the algorithms to rebuild the phase. The synthesis hop size $a = 256$ was fixed and the analysis hop size was changed accordingly to achieve the desired effect. Sound examples for the EBU SQAM database along with Matlab/GNU Octave script generating them can be found at the accompanying web page. According to our informal listening tests, there is a



(a) Gaussian window



(b) Hann window



(c) Hamming window

Fig. 3: Comparison with the iterative algorithms, MOCHA-TIMIT database.

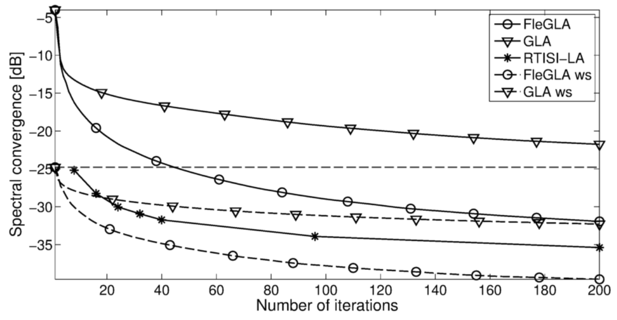
little perceivable difference between the algorithms with the exception of SPSI. As expected, the SPSI algorithm introduces disturbing "echo-like" effects to sounds that do not conform with the model assumption.

VI. Conclusion

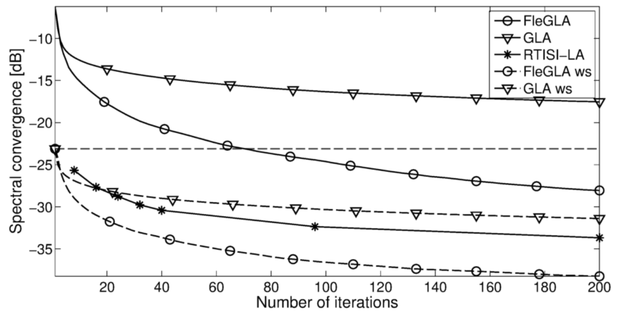
A novel, non-iterative algorithm for the reconstruction of the phase from the STFT magnitude has been proposed. The algorithm is computationally efficient and its performance is competitive with the state-of-the-art algorithms. It can also provide a suitable initial phase for the iterative algorithms.

As a future work it would be interesting to investigate whether (simple) equations similar to (23) and (24) could be found for non-Gaussian windows. Moreover, the effect of the aliasing and spectrogram modifications on the phase-magnitude relationship should be systematically explored.

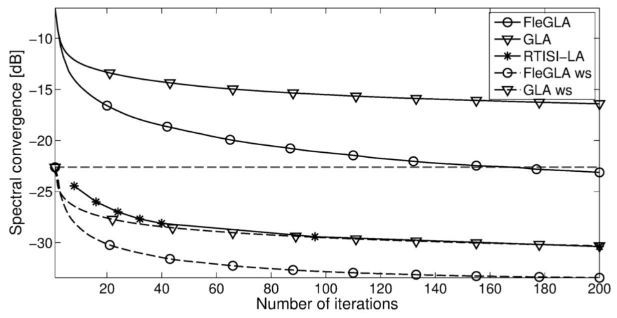
From the practical point of view, a drawback of the proposed algorithm is the inability to run in real-time setting i.e. to process streams of audio data in a frame by frame manner. Clearly,



(a) Gaussian window



(b) Hann window



(c) Hamming window

Fig. 4: Comparison with the iterative algorithms, EBU SQAM database.

the way how the phase is spread among the coefficients would have to be adjusted. This was done in [40] where we present a version of the algorithm introducing one or even zero frame delay.

Acknowledgements

The authors thank Pavel Rajmic for his valuable comments.

References

[1] K. Gröchenig, *Foundations of time-frequency analysis*, ser. Applied and numerical harmonic analysis. Boston, Basel, Berlin: Birkhäuser, 2001.

[2] D. Gunawan and D. Sen, “Iterative phase estimation for the synthesis of separated sources from single-channel mixtures,” *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 421–424, May 2010.

[3] N. Sturmel and L. Daudet, “Iterative phase reconstruction of Wiener filtered signals,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, March 2012, pp. 101–104.

[4] J. Laroche and M. Dolson, “Improved phase vocoder time-scale modification of audio,” *Speech and Audio Processing, IEEE Transactions on*, vol. 7, no. 3, pp. 323–332, May 1999.

[5] V. Gnann and M. Spiertz, “Comb-filter free audio mixing using STFT magnitude spectra and phase estimation,” in *Proc. 11th Int. Conf. on Digital Audio Effects (DAFx-08)*, Sep. 2008.

[6] P. Smaragdis, B. Raj, and M. Shashanka, “Missing data imputation for time-frequency representations of audio signals,” *Journal of Signal Processing Systems*, vol. 65, no. 3, pp. 361–370, 2011.

[7] D. Griffin and J. Lim, “Signal estimation from modified short-time Fourier transform,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 32, no. 2, pp. 236–243, Apr 1984.

[8] X. Zhu, G. T. Beauregard, and L. Wyse, “Real-time signal estimation from modified short-time Fourier transform magnitude spectra,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 5, pp. 1645–1653, July 2007.

[9] J. Le Roux, H. Kameoka, N. Ono, and S. Sagayama, “Fast signal reconstruction from magnitude STFT spectrogram based on spectrogram consistency,” in *Proc. 13th Int. Conf. on Digital Audio Effects (DAFx-10)*, Sep. 2010, pp. 397–403.

[10] N. Perraudin, P. Balazs, and P. Søndergaard, “A fast Griffin-Lim algorithm,” in *Applications of Signal Processing to Audio and Acoustics (WASPAA), IEEE Workshop on*, Oct 2013, pp. 1–4.

[11] N. Sturmel and L. Daudet, “Signal reconstruction from STFT magnitude: A state of the art,” *Proc. 14th Int. Conf. Digital Audio Effects (DAFx-11)*, pp. 375–386, 2011.

[12] R. Decorsiere, P. Søndergaard, E. MacDonald, and T. Dau, “Inversion of auditory spectrograms, traditional spectrograms, and other envelope representations,” *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 23, no. 1, pp. 46–56, Jan 2015.

[13] I. Waldspurger, A. d’Aspremont, and S. Mallat, “Phase recovery, maxcut and complex semidefinite programming,” *Math. Program.*, vol. 149, no. 1-2, pp. 47–81, 2015.

[14] E. J. Candes, Y. C. Eldar, T. Strohmer, and V. Voroninski, “Phase retrieval via matrix completion,” *SIAM Review*, vol. 57, no. 2, pp. 225–251, 2015.

[15] D. L. Sun and J. O. Smith, III, “Estimating a signal from a magnitude spectrogram via convex optimization,” in *Audio Engineering Society Convention 133*, Oct 2012.

[16] R. Balan, “On signal reconstruction from its spectrogram,” in *Information Sciences and Systems (CISS), 44th Annual Conference on*, March 2010, pp. 1–4.

[17] Y. Eldar, P. Sidorenko, D. Mixon, S. Barel, and O. Cohen, “Sparse phase retrieval from short-time Fourier measurements,” *IEEE Signal Processing Letters*, vol. 22, no. 5, pp. 638–642, May 2015.

[18] J. V. Bouvrie and T. Ezzat, “An incremental algorithm for signal reconstruction from short-time Fourier transform magnitude.” in *Spoken Language Processing (INTERSPEECH), 9th International Conference on*. ISCA, 2006.

[19] G. T. Beauregard, M. Harish, and L. Wyse, “Single pass spectrogram inversion,” in *Digital Signal Processing (DSP), IEEE International Conference on*, July 2015, pp. 427–431.

[20] P. Margon, R. Badeau, and B. David, “Phase reconstruction of spectrograms with linear unwrapping: application to audio signal restoration,” in *Proc. 23rd European Signal Processing Conference (EUSIPCO 2015)*, Aug 2015.

[21] M. R. Portnoff, “Magnitude-phase relationships for short-time Fourier transforms based on Gaussian analysis windows,” in *Acoustics, Speech, and Signal Processing (ICASSP), 1979 IEEE International Conference on*, vol. 4, Apr 1979, pp. 186–189.

[22] B. Nouvel, “A study of a local-features-aware model for the problem of phase reconstruction from the magnitude spectrogram,” in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, March 2010, pp. 4026–4029.

[23] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, *GNU Octave version 4.0.0 manual: A high-level interactive language for numerical computations*, 2015. [Online]. Available: http://www.gnu.org/software/octave/doc/interpreter

[24] P. L. Søndergaard, B. Torrésani, and P. Balazs, “The Linear Time Frequency Analysis Toolbox,” *International Journal of Wavelets, Multiresolution Analysis and Information Processing*, vol. 10, no. 4, 2012.

[25] Z. Průša, P. L. Søndergaard, N. Holighaus, C. Wiesmeyr, and P. Balazs, “The Large Time-Frequency Analysis Toolbox 2.0,” in *Sound, Music, and Motion*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 419–442.

[26] P. L. Søndergaard, “Gabor frames by Sampling and Periodization,” *Adv. Comput. Math.*, vol. 27, no. 4, pp. 355 –373, 2007.

[27] ——, “Finite discrete Gabor analysis,” Ph.D. dissertation, Technical University of Denmark, available from: http://ltfat.github.io/notes/ltfatnote003.pdf.

[28] ——, “Efficient algorithms for the discrete Gabor transform with a long FIR window,” *J. Fourier Anal. Appl.*, vol. 18, no. 3, pp. 456–470, 2012.

[29] J. B. Conway, *Functions of One Complex Variable I*, 2nd ed., ser. Graduate texts in Mathematics. Springer-Verlag New York, 1978, vol. 11.

[30] T. J. Gardner and M. O. Magnasco, "Sparse time-frequency representations," *Proceedings of the National Academy of Sciences*, vol. 103, no. 16, pp. 6094–6099, 2006.

[31] F. Auger, E. Chassande-Mottin, and P. Flandrin, "On phase-magnitude relationships in the short-time Fourier transform," *IEEE Signal Processing Letters*, vol. 19, no. 5, pp. 267–270, May 2012.

[32] E. Chassande-Mottin, I. Daubechies, F. Auger, and P. Flandrin, "Differential reassignment," *IEEE Signal Processing Letters*, vol. 4, no. 10, pp. 293–294, 1997.

[33] G. N. Felder and K. M. Felder, *Mathematical Methods in Engineering and Physics*. John Wiley & Sons, Inc., 2015.

[34] J. W. J. Williams, "Algorithm 232: Heapsort," *Communications of the ACM*, vol. 7, no. 6, p. 347348, 1964.

[35] A. Wrench, "MOCHA-TIMIT: Multichannel articulatory database," 1999. [Online]. Available: http://www.cstr.ed.ac.uk/research/projects/artic/mocha.html

[36] "Tech 3253: Sound Quality Assessment Material recordings for subjective tests," The European Broadcasting Union, Geneva, Tech. Rep., Sept. 2008. [Online]. Available: https://tech.ebu.ch/docs/tech/tech3253.pdf

[37] M. Schmidt, "minFunc: unconstrained differentiable multivariate optimization in Matlab," 2005, http://www.cs.ubc.ca/%7Eschmidtm/Software/minFunc.html.

[38] J. Le Roux, H. Kameoka, N. Ono, and S. Sagayama, "Phase initialization schemes for faster spectrogram-consistency-based signal reconstruction," in *Proceedings of the Acoustical Society of Japan Autumn Meeting*, no. 3-10-3, Mar. 2010.

[39] U. Zoelzer, Ed., *DAFX: Digital Audio Effects*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[40] Z. Průša and P. L. Søndergaard, "Real-Time Spectrogram Inversion Using Phase Gradient Heap Integration," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Sep 2016, to appear.