A Non-iterative Method for Reconstruction of Phase from STFT Magnitude

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

Abstract—A non-iterative method for the reconstruction 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 the un-sampled STFT with respect to the Hann 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 [1] and later for generic linear systems [2]. 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 [3]. The need for an effective way to reconstruct the phase arises in audio processing applications such as source separation and denoising [4], [5], time-stretching/pitch shifting [6], channel mixing [7], and missing data imputation [8].

The problem has already been addressed by many authors. Among the iterative algorithms, the most widespread and influential is the algorithm introduced by Griffin and Lim [9] (GLA) which inspired several extensions [10], [11] (FleGLA) and [12], [13] (TF-RTSI-LA). For a detailed overview of the algorithms based on GLA we refer the reader to the work by Sturmel and Daudet [14]. A different approach was taken by Decorsiere et al. [15] (BBFGS). They expressed the problem as an unconstrained optimization problem and 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 optimization problem [16], [17], [18], [19]. 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. Eldar et al. [20] assume the signal to be sparse in the original domain, which is not realistic in the context of the audio processing applications mentioned above. An approach presented by Bouvrie and Ezzat [21] is based on solving a non-linear system of equations for each time frame. The authors proposed to use an iterative solver and initialize it with samples obtained from previous frames. The algorithm is, however, designed to work exclusively with a rectangular window, which is known to have bad frequency selectivity.

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. A non-iterative algorithm proposed by Beauregard et al. [22] (SPSI) is based on the notion of phase consistency used in the phase vocoder [6]. 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. Magron et al. [23] introduced a similar algorithm based on phase unwrapping (PU). It acts entirely like SPSI for harmonic components but it tries to treat the impulse-like components separately.

In this paper, we propose a non-iterative algorithm called Phase Gradient Heap Integration (PGHI). The theory behind PGHI has been known at least since 1979 when Portnoff [24] presented a simple relationship between the partial derivatives of the phase and the log of the magnitude of a STFT computed using a Gaussian window. Given the phase gradient expressed using the magnitude gradient and given the phase at one point, one can invoke the gradient theorem to integrate and obtain the phase elsewhere. To our knowledge, no such algorithm has been published yet. In our previous work [25], we have presented a special case of PGHI adapted to the real-time setting. The present paper focuses on providing a complete mathematical treatment and on a thorough comparison with other algorithms in the offline setting. The aforementioned algorithms SPSI and PU are in fact close to the PGHI algorithm since they both basically perform a simple integration of the estimate of instantaneous frequency and in case of PU also of the local group delay, which are components of the STFT phase gradient. Their approach however cannot estimate the
gradient at every time-frequency position and the estimation requires analysing the spectrogram content.

In the spirit of reproducible research, the implementation of the algorithms, audio examples, color version of the figures as 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 [26] packages LTFAT [27], [28] and PHASERET available at http://ltfat.github.io and http://ltfat.github.io/phasernet, 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 algorithms, and http://ltfat.github.io/phaseret, respectively. [27], [28] and PHASERET available at http://ltfat.github.io and http://ltfat.github.io/notes/040. The code as well as scripts reproducing experiments from this manuscript are freely available at http://ltfat.github.io/notes/040. The code requires analysing the spectrogram content.

B. Discrete Gabor Transform – DGT

The discrete Gabor transform coefficients \( c \in \mathbb{C}^{M \times N} \) of a signal \( f \in \mathbb{R}^L \) with respect to a window \( g \in \mathbb{R}^L \) can be obtained as [29]

\[
    c(m, n) = \sum_{l=0}^{L-1} f(l + na)g(l)e^{-i2\pi ml/M} \quad (4)
\]

for \( m = 0, \ldots, M - 1 \) and \( n = 0, \ldots, N - 1, M = L/b \) 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 the time direction and \( b \) is a hop size in samples in the frequency direction and \((l+na)\) is assumed to be evaluated modulo \( L \). Separating amplitude and phase also in the discrete case we get

\[
    c(m, n) = s(m, n) \cdot e^{i\phi(m, n)}, \quad (5)
\]

\( s \) denoting magnitude of the coefficients and \( \phi \) denoting their phase. In the matrix notation, we can write \( c_{\text{vec}} = F_g^* f \), where \( c_{\text{vec}} \in \mathbb{C}^{MN} \) denotes the vectorized \( c \) such that \( c_{\text{vec}}(m + nM) = c(m, n) \) and \( F_g^* \) is a conjugate transpose of \( L \times MN \) matrix \( F_g \) (note that this matrix has a very particular block-structure [30]). The DGT can be seen as sampling of the STFT (both of the arguments \( \omega \) and \( t \) and the involved functions \( f \) and \( g \) themselves) of one period of \( L \)-periodic continuous signal \( f \) such that

\[
    c(m, n) = (V_g f) (bm, an) + A(m, n), \quad (6)
\]

for \( m = 0, \ldots, M - 1, n = 0, \ldots, N - 1 \) where \( A(m, n) \) models both the aliasing and numerical errors introduced by the sampling. The range of \( m \) can be shrunken to the first \( \lfloor M/2 \rfloor + 1 \) values as the remaining coefficients are only different by complex conjugation. Moreover, the zero-frequency coefficients (\( m = 0 \)) are always real and so are the Nyquist-frequency coefficients (\( m = M/2 \)) if \( M \) is even.

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

\[
    f(l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} c(m, n) g(l - na)e^{i2\pi m(l-na)/M} \quad (7)
\]

for \( l = 0, \ldots, L - 1 \). In the matrix notation, we can write \( f = F_{g^*} c_{\text{vec}} \). Here \( \tilde{g} \) is the canonical dual window, defined by

\[
    \tilde{g} = \left( F_g F_g^* \right)^{-1} g. \quad (8)
\]

See e.g. [31] for conditions under which the product \( F_g F_g^* \) is (easily) invertible and [32], [33] for efficient algorithms for computing (4), (7) and (8). In particular the block structure can be used for a pre-conditioning approach [30].

One period of the discretized and periodized Gaussian window is given by

\[
    \varphi_\lambda(l) = \sum_{k \in \mathbb{Z}} e^{-\pi \lambda^2 \frac{(l + k)^2}{M}}, \quad l = 0, \ldots, L - 1. \quad (9)
\]

We assume that \( L \) and \( \lambda \) are chosen such that the overlap of the window “tails” after periodization 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 \( h \in [0, 1] \) can be written as (expressed from (9) using just \( k = 0 \))

\[
    w_h = \sqrt{-\frac{4 \log(h)}{\pi}} \lambda L. \tag{10}
\]

The width is given in samples and it can be a non-integer number. This equation becomes relevant when working with truncated Gaussian window and when determining \( \lambda \) for non-Gaussian windows. For other window type used, we took \( \lambda \) of the closest Gaussian window in the least mean square error sense. The window was obtained via a simple heuristic search. Note that all windows used in this manuscript are odd symmetric, such that they have a unique center sample, and they are non-causal such that they introduce no delay. Finally, the discrete Fourier transform of such windows is real.

### III. STFT Phase Reconstruction

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. More precisely, the time derivative of the log-magnitude defines the frequency derivative of the phase and, vice versa, the frequency derivative of the log-magnitude defines the time derivative of the phase. In this section, we derive such relations and show that, in theory, it is possible to reconstruct the phase from its gradient up to a constant global phase shift. We include a complete derivation since the relations for the STFT as defined in (1) have not appeared in the literature, as far as we know. Our approach is based on the properties of the Bargmann transform [3], [34] which is closely related to the STFT with respect to the Gaussian window with \( \lambda = 1 \).

#### A. Phase-Magnitude Relationship

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

\[
(Bf)(z) = \int \! f(\tau) e^{2\pi i z \tau - \pi \tau^2} d\tau, \quad z \in \mathbb{C}
\]

is an entire function [35] and that it relates to the STFT defined in (1) such that

\[
(Bf)(z) = e^{z t \omega + \frac{|z|^2}{4}} (V_{\omega} f)(-\omega, t), \tag{12}
\]

assuming \( f \) is real valued and \( z = t + i \omega \). Furthermore, the logarithm of the Bargmann transform is an entire function (apart from zeros) and the real and imaginary parts of \( \log(Bf)(z) \) can be written as

\[
\log(Bf)(t + i \omega) = u(\omega, t) + iv(\omega, t) \tag{13}
\]

\[
u(\omega, t) = \pi (t^2 + \omega^2)/2 + \log M_{\omega}^f(-\omega, t) \tag{14}
\]

\[
v(\omega, t) = \pi t \omega + \Phi_{\omega}^f(-\omega, t) \tag{15}
\]

and using the Cauchy-Riemann equations

\[
\frac{\partial u}{\partial t}(\omega, t) = \frac{\partial \Phi_{\omega}^f}{\partial \omega}(-\omega, t), \quad \frac{\partial u}{\partial \omega}(\omega, t) = -\frac{\partial \Phi_{\omega}^f}{\partial t}(-\omega, t) \tag{16}
\]

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

\[
\frac{\partial \Phi_{\omega}^f}{\partial t}(\omega', t) = -\frac{\partial}{\partial t} \log M_{\omega}^f(\omega', t) \tag{17}
\]

\[
\frac{\partial \Phi_{\omega}^f}{\partial \omega}(\omega', t) = \frac{\partial}{\partial \omega} \log M_{\omega}^f(\omega', t) + 2\pi \omega'. \tag{18}
\]

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

**Proposition 1.** Let \( O, P \) be bounded operators such that for all \((\omega, t)\) there exist differentiable, strictly monotonous functions \( \eta(t) \) and \( \xi(\omega) \), such that \( T_{t}\mathcal{E}_{\omega} O = P T_{t}(\eta(t)) \mathcal{E}_{\xi(\omega)} \) and let \( g = O_{\varphi_1} \). Then

\[
\frac{\partial}{\partial t} \Phi_{\omega}^f(\omega, t) = -\frac{\partial}{\partial t} \log M_{\omega}^f(\omega, t) \cdot \xi'(\omega)(t) \tag{19}
\]

\[
\frac{\partial}{\partial \omega} \Phi_{\omega}^f(\omega, t) = \frac{\partial}{\partial \omega} \log M_{\omega}^f(\omega, t) \cdot \eta'(t) + 2\pi \xi(\omega) \eta'(t). \tag{20}
\]

**Proof.** Consider

\[
(V_{\eta} f)(\omega, t) = \langle f, T_{t\mathcal{E}_{\omega} O_{\varphi_1}} \rangle = \left\langle P^f, T_{t}(\eta(t)) \mathcal{E}_{\xi(\omega)} \varphi_1 \rightangle = \left(V_{\varphi_1} (P^f)\right)(\xi(\omega), \eta(t))
\]

and therefore

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

Furthermore

\[
\frac{\partial}{\partial \omega} \log M_{\omega}^f(\omega, t) = \left[ \frac{\partial}{\partial \omega} \log M_{\omega}^{P^f} (\xi(\omega), \eta(t)) \right] \cdot \xi'(\omega).
\]

Combining this with (18) we obtain (20)

\[
\frac{\partial}{\partial \omega} \Phi_{\omega}^f(\omega, t) = \left[ \frac{\partial}{\partial \omega} \log M_{\omega}^{P^f} (\xi(\omega), \eta(t)) \right] \cdot \eta'(t) = \left[ \frac{\partial}{\partial \omega} \log M_{\omega}^f(\omega, t) \cdot \eta'(t) + 2\pi \xi(\omega) \eta'(t). \right.
\]

The other equality can be shown using the same arguments and (17).

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

\[
\frac{\partial}{\partial t} \Phi_{\omega}^f(\omega, t) = -\lambda \frac{\partial}{\partial t} \log M_{\omega}^f(\omega, t) \tag{21}
\]

\[
\frac{\partial}{\partial \omega} \Phi_{\omega}^f(\omega, t) = \frac{1}{\lambda} \frac{\partial}{\partial \omega} \log M_{\omega}^f(\omega, t) + 2\pi \omega. \tag{22}
\]

The relations were already published in [24], [36], [37], [38] 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. Chassande-Mottin et al. [36] showed that similar equations exist even for general windows. They however involve additional non-analytic terms and thus it seems they cannot be exploited directly. Moreover, the experiments presented in Section V showed that the performance degradation is not too significant when using windows resembling the Gaussian window like the Hann, the Hamming or the Blackman window.

The STFT phase gradient of a signal \( f \) with respect to dilated Gaussian \( \varphi_\lambda \) will be further denoted as

\[
\nabla \Phi_{\varphi_\lambda}^f(\omega, t) = \left[ \frac{\partial}{\partial t} \Phi_{\varphi_\lambda}^f(\omega, t), \frac{\partial}{\partial \omega} \Phi_{\varphi_\lambda}^f(\omega, t) \right]. \tag{23}
\]
Note that the derivative of the phase has a peculiar pole pattern around zeros [39].

B. Gradient Integration and the Phase Shift Phenomenon

Knowing the phase gradient, one can exploit the gradient theorem (see e.g. [40]) to reconstruct the original (unwrapped) phase \( \Phi^f_{\psi_0}(\omega, t) \) such that

\[
\Phi^f_{\psi_0}(\omega, t) - \Phi^f_{\psi_0}(\omega_0, t_0) = \int_{t_0}^{t} \nabla \Phi^f_{\psi_0}(r(\tau)) \cdot \frac{dr}{d\tau}(\tau) \, d\tau,
\]

where \( r(\tau) = [r_\omega(\tau), r_t(\tau)] \) is any curve starting at \((\omega_0, t_0)\) and ending at \((\omega, t)\) provided the phase at the initial point \((\omega_0, t_0)\) is known. When the phase is unknown completely, we consider \( \Phi^f_{\psi_0}(\omega_0, t_0) = 0 \) which causes a global phase shift. The phase shift of the STFT carries over to the global phase shift of the reconstructed signal through 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. This relationship has to be either recovered or enforced because if one simply takes only the real part of the reconstructed signal the phase shift can cause amplitude attenuation or even causes the signal to vanish in extreme cases. 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 \( \psi_0 \) and define an analytic signal as

\[ x_\text{an}(t) = A(t)e^{i \psi(t)}. \]

The real part of the global phase shift \( e^{i \psi(t)} \) is given as \( R(x_\text{an}(t)e^{i \psi(t)}) = A(t) \cos(\psi(t) + \psi_0) \) which is what one would expect. Similarly, we define a real signal as \( x(t) = \frac{A(t)}{2} \left( e^{i \psi(t)} + e^{-i \psi(t)} \right) \) and the real part of such signal with the global phase shift \( \psi_0 \) amounts to \( R(x(t)e^{i \psi_0}) = A(t) \cos(\psi_0) \cos(\psi(t)) \) 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 sign ambiguity. For real signals, it is clear that the following holds for \( \omega \neq 0 \)

\[
\Phi^f_{\psi_0}(\omega, t) + \Phi^f_{\psi_0}(-\omega, t) = 2\psi_0.
\]

After the compensation, due to the phase wrapping, the phase shift is still ambiguous up to an integer multiple of \( \pi \), which causes the aforementioned signal sign ambiguity.

IV. THE ALGORITHM

In the discrete time setting (recall Section II-B; in particular (4) and (5)) the STFT phase gradient approximation \( \nabla \Phi^f_{\psi_0}(bm, mn) = \nabla \Phi(m, n) \) is obtained by numerical differentiation of \( s_{\text{log}}(m, n) = \log(s(m, n)) \) as

\[
\nabla \Phi(m, n) = \left[ \phi_w(m, n), \phi_t(m, n) \right] = \left[ \frac{\lambda}{a} (s_{\text{log}}D_t)(m, n), \frac{1}{\lambda b} (D_w s_{\text{log}})(m, n) + 2\pi m/M \right]
\]

where \( D_t, D_w \) denote matrices performing the numerical differentiation of \( s_{\text{log}} \) 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 (mid-point) finite difference scheme (see e.g. [41]) 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 exclusively one of the components in \( \frac{\partial \Phi}{\partial \tau} \) from (24) is zero. Due to this property, the gradient can be pre-scaled using hop sizes \( a \) and \( b \) such that

\[
\nabla \Phi^\text{SE}(m, n) = \left[ b\phi_w(m, n), a\phi_t(m, n) \right] = \left[ \frac{-\lambda L}{aM} (s_{\text{log}}D_t)(m, n), \frac{aM}{\lambda L} (D_w s_{\text{log}})(m, n) + 2\pi m/M \right].
\]

Note that the dependency on \( L \) can be avoided when (10) is used to express \( \lambda L \). This is useful e.g. when the signal length is not known in advance.

The numerical integration of the phase gradient is performed over the prominent contours of the spectrogram first in order to reduce accumulation of the error. The magnitude of the coefficients is used as a guide such that integration paths are chosen adaptively following the spectrogram ridges first. Such behavior is achieved by employing a heap data structure (from the heapsort algorithm [42], which 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. Even after employing the heap, nothing stops the integration paths to go trough areas with coefficients small in magnitude where the phase gradient estimate is unreliable [39]. Therefore, in order to avoid further accumulation of the error, we introduce the relative magnitude tolerance \( tol \). It causes the algorithm to perform the integration only locally on “islands” of coefficients above \( tol \) with the max coefficient within the island serving as the zero phase reference. The coefficients below \( tol \) are assigned a random phase (uniformly distributed random values from the range \([0, 2\pi]\)). 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 and a graphical step-by-step example can be found at the accompanying webpage.

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

\[
\hat{c}(m, n) = s(m, n)e^{i \hat{\phi}(m, n)}
\]

and the signal \( \hat{f} \) is recovered by simply plugging these coefficients into (7).

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
Algorithm 1: Phase gradient heap integration – PGHI

Input: DGT phase gradient
\( \nabla \phi^\text{SC}(m,n) = (\phi^\text{SC}_0(m,n), \phi^\text{SC}_1(m,n)) \) obtained from (29), magnitude of DGT coefficients |c(m,n)|, relative tolerance tol.

Output: Estimate of the DGT phase \( \hat{\phi}(m,n) \).

1 Set \( I = \{(m,n) : \|c(m,n)\| > tol \cdot \max |c(m,n)| \} \);
2 Assign random values to \( \hat{\phi}(m,n) \) where \( (m,n) \notin I \);
3 Construct a self-sorting heap for \( (m,n) \) pairs;
4 while \( I \) is not empty do
5 
6 if heap is empty then
7 
8 Assign \( (m,n)\)max = arg max\( (m,n)\) \in I \((|c(m,n)|)\) into the heap;
9 \( \hat{\phi}(m,n)\)max \( \leftarrow \) 0;
10 Remove \( (m,n)\)max from \( I \);
11 end
12 while heap is not empty do
13 
14 if \( (m+1,n) \in I \) then
15 
16 \( \hat{\phi}(m+1,n) \) \( \leftarrow \)
17 \( \hat{\phi}(m,n) + \frac{1}{2} (\phi^\text{SC}_0(m,n) + \phi^\text{SC}_1(m+1,n)) \);
18 Insert \( (m+1,n) \) into the heap;
19 Remove \( (m+1,n) \) from \( I \);
20 end
21 if \( (m-1,n) \in I \) then
22 
23 \( \hat{\phi}(m-1,n) \) \( \leftarrow \)
24 \( \hat{\phi}(m,n) - \frac{1}{2} (\phi^\text{SC}_0(m,n) + \phi^\text{SC}_1(m-1,n)) \);
25 Insert \( (m-1,n) \) into the heap;
26 Remove \( (m-1,n) \) from \( I \);
27 end
28 if \( (m,n+1) \in I \) then
29 
30 \( \hat{\phi}(m,n+1) \) \( \leftarrow \)
31 \( \hat{\phi}(m,n) + \frac{1}{2} (\phi^\text{SC}_0(m,n) + \phi^\text{SC}_1(m,n+1)) \);
32 Insert \( (m,n+1) \) into the heap;
33 Remove \( (m,n+1) \) from \( I \);
34 end
35 if \( (m,n-1) \in I \) then
36 
37 \( \hat{\phi}(m,n-1) \) \( \leftarrow \)
38 \( \hat{\phi}(m,n) - \frac{1}{2} (\phi^\text{SC}_0(m,n) + \phi^\text{SC}_1(m,n-1)) \);
39 Insert \( (m,n-1) \) into the heap;
40 Remove \( (m,n-1) \) from \( I \);
41 end
42 end
43 end

Algorithm 2: Initialization for partially known phase

Input: Set of indices of coefficients \( M \) with known phase \( \phi(m,n) \).

1 \( \hat{\phi}(m,n) \) \( \leftarrow \) \( \phi(m,n) \) for \( (m,n) \in M \);
2 for \( (m,n) \in M \cap I \) do
3 
4 if \( (m+1,n) \notin M \) or \( (m-1,n) \notin M \) or \( (m,n+1) \notin M \) or \( (m,n-1) \notin M \) then
5 Add \( (m,n) \) to the heap;
6 end
7 end

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. The effect of the length of the time hop size \( a \) on the performance of the algorithm is illustrated by the phase error plots depicted in Figure 1. In the aliasing free case (Fig. 1b), the algorithm even achieves a constant phase shift for all coefficients with relative magnitude above –60 dB (cf. phase shift phenomenon in Section III-B). This behavior however quickly deteriorates when longer hop size is introduced (Fig. 1c and 1d). 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. \( M = 5888 \)).

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. The authors recommend to use redundancy \( M/a = 8 \) and \( \lambda = aM/L \) or simply 87.5% window overlap with compactly supported windows. Such setting is also used in Section V. An interested reader can find a demo script comparing several window overlaps at the accompanying web page.

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, we reconstruct the phase only for the positive frequency coefficients and enforce the conjugate symmetry to the negative frequency coefficients.

B. Exploiting Partially Known Phase

In some scenarios, the true phase of some of the 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. Formally, Alg. 1 will be changed such that steps summarized in Alg. 2 are inserted after line 3. 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.
where \( m_0 \) is the estimated, possibly non-integer instantaneous frequency belonging to the interval \([0, [M/2]]\). 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 PU goes further and also 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

\[
\phi(m, n) = \phi(m - 1, n) + 2\pi a(n - n_0)/M,
\]

where \( an_0 \) is the time index 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 the PU algorithm 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 SPSI and PU and by the proposed algorithm PGHI. The phase difference at the transient coefficients is somewhat smoother for PU when compared to SPSI because of the involved impulse model. PGHI 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 \( \lambda = aM/L \).

V. EXPERIMENTS

In the experiments, we use the normalized mean-squared error to measure the performance

\[
E(x, y) = \frac{\|x - y\|}{\|x\|}, \quad E_{\text{db}}(x, y) = 20 \log_{10} E(x, y),
\]

where \( \| \cdot \| \) denotes the standard energy norm. The spectral convergence [14] is defined as

\[
C = E(s_{\text{vec}}, |\mathbf{P}c_{\text{vec}}|), \quad C_{\text{db}} = 20 \log_{10} C,
\]
where \( P = F^*_g F_g \) i.e. synthesis followed by analysis. Other authors proposed a slightly different measure \( \|C_{vec} \|_2^2 \), termed normalised inconsistency measure \( [10] \), which represents the normalised energy lost by the reconstruction/projection. Such measures clearly do not accurately reflect the actual signal reconstruction error \( R = E(f, \hat{f}) \), but they are independent of the phase shift. Some other authors evaluate the algorithms using the signal to noise ratio, which they define as \( SNR(x, y) = 1/E(x, y) \) and \( SNR_{db}(x, y) = -E_{db}(x, y) \) respectively. Unfortunately, as 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 (i.e. individual short-time spectra) and even each frequency bin within the frame might have a different phase shift, causing the error \( R \) to be very high, even when the other error measures are low and the actual perceived quality is good. An interested reader can find sound examples demonstrating this phenomenon at the accompanying webpage.

The testing was performed on the speech corpus database MOCHA-TIMIT \([43]\) consisting of recordings of 1 male and 1 female speakers (460 recordings for each, 61 minutes in total). The sampling rate of all recordings is 16 kHz. The Gabor system parameters used with this database (Fig. 3 and 5) were: number of channels \( M = 1024 \), hop size \( a = 128 \), time-frequency ratio of the Gaussian window \( \lambda = \alpha M/L \), 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 \([44]\) 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 (Fig. 4 and 6) were the following: number of channels \( M = 2048 \), hop size \( a = 256 \), time-frequency ratio of the Gaussian window \( \lambda = \alpha M/L \), time support of the truncated Gaussian window and of the other compactly supported windows was \( M \) samples.

In the following Section V-A, we evaluate the performance of the PGHI algorithm alone. Later in Section V-B, we evaluate the performance of several iterative algorithms initialized by the outcome of PGHI. In both cases, we will compare the results with the SPSI \([22]\) algorithm. Unfortunately, we were not able to get good results with the PU \([23]\) algorithm consistently due to the imperfect onset detection and due to the limitation of the impulse model and so did not include it here. The implementation of SPSI has been taken from http://anlab.org/software/phaserecon/ and it was modified to fit our framework. The most prominent change has been the removal of the alternating \( \pi \) and \( 0 \) phase modulation in the frequency direction which is not present when computing the transform according to (4).

The results for the PGHI 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 \( tol = 10^{-10} \) including steps from Alg. 2 while using the result from the first step as known phase. 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.

A. Comparison With Non-iterative Method

Figures 3 and 4 show box plots of \( C_{db} \) over entire databases for the SPSI and the proposed algorithm PGHI. The proposed algorithm very 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.

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. On a standard PC, the execution time of PGHI was generally less than 1 second for the 10 second excerpts from the SQAM database. The SPSI algorithm was roughly 6–8 times faster. Our current implementation of PGHI is however very slow for noise signals.

B. Comparison with Iterative Methods

It is known that the iterative phase reconstruction algorithms optimize a non-convex objective function and, therefore, the result depends strongly on the initial phase estimate. In this
section, we compare the effect of PGHI and SPSI initializations on the performance of the following iterative algorithms:

- A combination of Le Roux’s modification of GLA [10] using the on-the-fly truncated modified update and of the fast version of GLA [11] with constant $\alpha = 0.99$ (FleGLA). The projection kernel was always truncated to size $2M/a - 1$ in both directions. This combination outperforms both algorithms [10] and [11] when used individually.
- The gradient descend-like algorithm by Decorsiere et al. [15] (IBFGS). Unfortunately, the IBFGS implementation we use (downloaded from [45]) fails in some cases.

In the comparisons, we also include the following algorithm, which, unfortunately, does not benefit from phase initialization as it performs its own initial phase guess from the partially reconstructed signal:

- Time-Frequency domain Real-Time Iterative Spectrogram Inversion with Look-Ahead [13] (TF-RTISI-LA). 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.

Figures 5 and 6 show average $C$ in dB over the MOCHA-TIMIT and EBU SQAM databases respectively depending on the number of iterations with SPSI initialization (solid lines) and with PGHI initialization (dashed lines). 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 IBFGS algorithm has been excluded from the comparison using the EBU-SQAM database (Fig. 6); it failed to finish for a considerable number of the excerpts. The graphs show that PGHI provides a better initial phase estimate than SPSI for all algorithms considered and the best overall results are obtained when PGHI is combined with the FleGLA and IBFGS algorithms. The performance gap is however less prominent for non-Gaussian windows. That is to be expected as the PGHI algorithm performs suboptimally. For the non-Gaussian windows the FleGLA and the IBFGS algorithms give almost equivalent results after 200 iteration for both types of initializations.

In the tests, the execution time of the PGHI algorithm was comparable to the execution time of 2–4 iterations of GLA 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 modified spectrograms. The spectrograms are modified in the complex-valued STFT
domain. This could be done by multiplication which leads to so-called Gabor filters [46], [47], [48] or by moving/copying of contents. In general, such a modified spectrogram is no longer a valid (consistent [10]) spectrogram, i.e. there is no signal having such a spectrogram. Therefore the task is to construct rather than reconstruct a suitable phase. Unfortunately, it is neither clear for which spectrogram modifications the equations (21) and (22) still hold nor how it does affect the performance if they do not. Moreover, an objective comparison of the algorithms becomes difficult as the error measures chosen above 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 [6], [49] 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.

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 iterative algorithms.

As future work, it would be interesting to investigate whether (simple) equations similar to (21) and (22) 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. For that we will extend Proposition 1 to a more general setting. Ideally, we hope that a similar result could be possible for $\alpha$-modulation frames [50], [51] and warped time-frequency frames [52], [53].

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, the way how the phase is spread among the coefficients would have to be adjusted. This was done in [25] where we present a version of the algorithm introducing one or even zero frame delay.

Further, please note that equations (21) and (22) hold “in the other direction” as well; meaning they can be used to estimate the magnitude given the phase. This property might be useful in many applications since the phase-aware signal processing is a promising field of research [54], [55].

Acknowledgements

The authors thank Pavel Rajmic and the anonymous reviewers for their valuable comments.

References


