HAL
open science

Sparsity and cosparsity for audio declipping: a flexible non-convex approach

Srđan Kitić, Nancy Bertin, Rémi Gribonval

To cite this version:

Srđan Kitić, Nancy Bertin, Rémi Gribonval. Sparsity and cosparsity for audio declipping: a flexible non-convex approach. LVA/ICA 2015 - The 12th International Conference on Latent Variable Analysis and Signal Separation, Aug 2015, Liberec, Czech Republic. pp.8. hal-01159700v2

HAL Id: hal-01159700

https://inria.hal.science/hal-01159700v2

Submitted on 8 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sparsity and cosparsity for audio declipping: a flexible non-convex approach

Srđan Kitić, Nancy Bertin, and Rémi Gribonval *

Inria/IRISA, Rennes, PANAMA team
srdjan.kitic@inria.fr, nancy.bertin@irisa.fr, remi.gribonval@inria.fr
team.inria.fr/panama

Abstract. This work investigates the empirical performance of the sparse synthesis versus sparse analysis regularization for the ill-posed inverse problem of audio declipping. We develop a versatile non-convex heuristics which can be readily used with both data models. Based on this algorithm, we report that, in most cases, the two models perform almost similarly in terms of signal enhancement. However, the analysis version is shown to be amenable for real time audio processing, when certain analysis operators are considered. Both versions outperform state-of-the-art methods in the field, especially for the severely saturated signals.

Keywords: Clipping, audio, sparse, cosparse, non-convex, real-time

1 Introduction

Clipping, or magnitude saturation, is a well-known problem in signal processing, from audio [1,13] to image processing [2,18] and digital communications [17]. The focus of this work is audio declipping, to restore clipped audio signals. Audio signals become saturated usually during acquisition, reproduction or A/D conversion. The perceptual manifestation of clipped audio depends on the level of clipping degradation and the audio content. In case of mild to moderate clipping, the listener may notice occasional "clicks and pops" during playback. When clipping becomes severe, the audio content is usually perceived as if it was contaminated with a high level of additive noise, which may be explained by the introduction of a large number of harmonics caused by the discontinuities in the degraded signal. In addition to audible artifacts, some recent studies have shown that clipping has a negative impact on Automatic Speech Recognition (ASR) performance [22,11].

In the following text, a sampled audio signal is represented by the vector $\boldsymbol{x} \in \mathbb{R}^{\mathrm{n}}$ and its clipped version is denoted by $\boldsymbol{y} \in \mathbb{R}^{\mathrm{n}}$. The latter can be easily deduced from $\boldsymbol{x}$ through the following nonlinear observation model, called *hard clipping*:

$$\boldsymbol{y}_{\mathrm{i}} = \begin{cases} \boldsymbol{x}_{\mathrm{i}} & \text{for } |\boldsymbol{x}_{\mathrm{i}}| \leq \tau, \\ \mathrm{sign}(\boldsymbol{x}_{\mathrm{i}})\tau & \text{otherwise.} \end{cases} \tag{1}$$

* This work was supported in part by the European Research Council, PLEASE project (ERC-StG-2011-277906).

While idealized, this clipping model is a convenient approximation allowing to clearly distinguish the clipped parts of a signal by identifying the samples having the highest absolute magnitude. Indices corresponding to "reliable" samples of $\boldsymbol{y}$ (not affected by clipping) are indexed by Ω_r, while Ω_c^+ and Ω_c^- index the clipped samples with positive and negative magnitude, respectively.

Our goal is to estimate the original signal $\boldsymbol{x}$ from its clipped version $\boldsymbol{y}$, *i.e.* to "declip" the signal $\boldsymbol{y}$. Ideally, the estimated signal $\hat{\boldsymbol{x}}$ should satisfy natural magnitude constraints in order to be consistent with the clipped observations. Thus, we seek an estimate $\hat{\boldsymbol{x}}$ which fulfills the following criteria:

$$\boldsymbol{M}_{\mathrm{r}}\,\hat{\boldsymbol{x}} = \boldsymbol{M}_{\mathrm{r}}\,\boldsymbol{y} \qquad \boldsymbol{M}_{\mathrm{c}}^{+}\,\hat{\boldsymbol{x}} \geq \boldsymbol{M}_{\mathrm{c}}^{+}\,\boldsymbol{y} \qquad \boldsymbol{M}_{\mathrm{c}}^{-}\,\hat{\boldsymbol{x}} \leq \boldsymbol{M}_{\mathrm{c}}^{-}\,\boldsymbol{y}, \tag{2}$$

where the matrices $\boldsymbol{M}_{\mathrm{r}}$, $\boldsymbol{M}_{\mathrm{c}}^{-}$ and $\boldsymbol{M}_{\mathrm{c}}^{+}$ are *restriction operators*. These are simply row-reduced identity matrices used to extract the vector elements indexed by the sets Ω_r, Ω_c^+ and Ω_c^-, respectively. We write the constraints (2) as $\hat{\boldsymbol{x}} \in \Gamma(\boldsymbol{y})$.

Obviously, consistency alone is not sufficient to ensure uniqueness of $\hat{\boldsymbol{x}}$, thus one needs to further regularize the inverse problem. The declipping inverse problem is amenable to several regularization approaches proposed in the literature, such as based on linear prediction [12], minimization of the energy of high order derivatives [11], psychoacoustics [6], sparsity [1,15,21,6,23] and cosparsity [14] (where we introduced a simplified version of the analysis-based algorithm presented in this paper). The last two *priors*, briefly explained in the next section, enable some state-of-the-art methods in clipping restoration.

In this paper we empirically compare the performance of the two priors, by means of a declipping algorithm which is easily adaptable to both cases. Our findings are that the sparsity-based version of the algorithm marginally outperforms the cosparsity-based one, but this fact may be attributed to the choice of the stopping criterion. On the other hand, for a class of analysis operators, the cosparsity-based algorithm has very low complexity per iteration, which makes it suitable for real-time audio processing.

2 The sparse synthesis and sparse analysis data models

It is well-known that the energy of audio signals is often concentrated either in a small number of frequency components, or in short temporal bursts [20], *i.e.* they are (approximately) time-frequency sparse. The traditional sparse synthesis viewpoint [8,9] on this property is that audio signals are well approximated by linearly combining few columns of a *dictionary* matrix $\boldsymbol{D} \in \mathbb{C}^{\mathrm{n}\times\mathrm{d}}$, $\mathrm{d} \geq \mathrm{n}$ such as a Gabor dictionary, *i.e.* $\boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{z}$, where $\boldsymbol{z} \in \mathbb{C}^{\mathrm{d}}$ is sparse. A less explored alternative is the cosparse analysis perspective [19] asserting that $\boldsymbol{A}\boldsymbol{x}$ is approximately sparse, with $\boldsymbol{A} \in \mathbb{C}^{\mathrm{p}\times\mathrm{n}}$, $\mathrm{p} \geq \mathrm{n}$ and *analysis operator*. The two data models are different [7,19], unless $\mathrm{p} = \mathrm{n}$ and $\boldsymbol{A} = \boldsymbol{D}^{-1}$. Finding the sparsest (in the sense of synthesis or analysis) vector $\boldsymbol{x}$ satisfying constraints such as (2) is in general intractable, but convex or greedy heuristics provide efficient algorithms with certain performance guarantees [8,9,19].

3 Algorithms

Some empirical evidence [6,23] suggests that standard ℓ_1 convex relaxation does not perform well for sparse synthesis regularization of the declipping inverse problem. Therefore, we developed an algorithmic framework based on non-convex heuristics, that can be straightforwardly parametrized for use in both the synthesis and the analysis setting. To allow for possible real-time implementation, the algorithms operate on individual blocks (chunks) of audio data, which is subsequently resynthesized by means of the overlap-add scheme.

The heuristics should approximate the solution of the following synthesis- and analysis-regularized inverse problems[1]:

$$\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}}\, \|\boldsymbol{z}\|_0 + \mathbf{1}_{\Gamma(\boldsymbol{y})}(\boldsymbol{x}) + \mathbf{1}_{\ell_2\leq\varepsilon}(\boldsymbol{x}-\boldsymbol{D}\boldsymbol{z}) \tag{3}$$

$$\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}}\, \|\boldsymbol{z}\|_0 + \mathbf{1}_{\Gamma(\boldsymbol{y})}(\boldsymbol{x}) + \mathbf{1}_{\ell_2\leq\varepsilon}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{z}). \tag{4}$$

The indicator function $\mathbf{1}_{\Gamma(\boldsymbol{y})}$ of the constraint set $\Gamma(\boldsymbol{y})$ forces the estimate $\boldsymbol{x}$ to satisfy (2). The additional penalty $\mathbf{1}_{\ell_2\leq\varepsilon}$ is a *coupling* functional. Its role is to enable the end-user to explicitly bound the distance between the estimate and its sparse approximation. These are difficult optimization problems: besides inherited NP-hardness, the two problems are also non-convex and non-smooth.

We can represent (3) and (4) in an equivalent form, using the indicator function on the cardinality of $\boldsymbol{z}$ and an integer-valued unknown k:

$$\underset{\boldsymbol{x},\boldsymbol{z},\mathrm{k}}{\text{minimize}}\, \mathbf{1}_{\ell_0\leq\mathrm{k}}(\boldsymbol{z}) + \mathbf{1}_{\Gamma(\boldsymbol{y})}(\boldsymbol{x}) + F_c(\boldsymbol{x},\boldsymbol{z}) \tag{5}$$

where $F_c(\boldsymbol{x},\boldsymbol{z})$ is the appropriate coupling functional. For a fixed k, problem (5) can be seen as a variant of the *regressor selection* problem, which is (locally) solvable by the Alternating Direction Method of Multipliers (ADMM) [5,3]:

Synthesis version	Analysis version
$\bar{\boldsymbol{z}}^{(\mathrm{i}+1)} = \mathcal{H}_\mathrm{k}(\hat{\boldsymbol{z}}^{(\mathrm{i})} + \boldsymbol{u}^{(\mathrm{i})})$	$\bar{\boldsymbol{z}}^{(\mathrm{i}+1)} = \mathcal{H}_\mathrm{k}(\boldsymbol{A}\hat{\boldsymbol{x}}^{(\mathrm{i})} + \boldsymbol{u}^{(\mathrm{i})})$
$\hat{\boldsymbol{z}}^{(\mathrm{i}+1)} = \arg\min_{\boldsymbol{z}} \|\boldsymbol{z} - \bar{\boldsymbol{z}}^{(\mathrm{i}+1)} + \boldsymbol{u}^{(\mathrm{i})}\|_2^2$	$\hat{\boldsymbol{x}}^{(\mathrm{i}+1)} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \bar{\boldsymbol{z}}^{(\mathrm{i}+1)} + \boldsymbol{u}^{(\mathrm{i})}\|_2^2$
subject to $\boldsymbol{D}\boldsymbol{z} \in \Gamma(\boldsymbol{y})$	subject to $\boldsymbol{x} \in \Gamma(\boldsymbol{y})$
$\boldsymbol{u}^{(\mathrm{i}+1)} = \boldsymbol{u}^{(\mathrm{i})} + \hat{\boldsymbol{z}}^{(\mathrm{i}+1)} - \bar{\boldsymbol{z}}^{(\mathrm{i}+1)}$	$\boldsymbol{u}^{(\mathrm{i}+1)} = \boldsymbol{u}^{(\mathrm{i})} + \boldsymbol{A}\hat{\boldsymbol{x}}^{(\mathrm{i}+1)} - \bar{\boldsymbol{z}}^{(\mathrm{i}+1)}.$

(6)

The operator $\mathcal{H}_\mathrm{k}(\boldsymbol{v})$ performs hard thresholding, *i.e.* sets all but k highest in magnitude components of $\boldsymbol{v}$ to zero. Unlike the standard regressor selection algorithm, for which the ADMM multiplier [5] needs to be appropriately chosen to avoid divergence, the above formulation is independent of its value.

In practice, it is difficult to guess the optimal value of k beforehand. An adaptive estimation strategy is to periodically increase k (starting from some

[1] Observe that if $\boldsymbol{D}$ and $\boldsymbol{A}$ are unitary matrices, the two problems become identical.

small value), perform several runs of (6) for a given k and repeat the procedure until the constraint embodied by F_c is satisfied. This corresponds to *sparsity relaxation*: as k gets larger, the estimated $\boldsymbol{z}$ becomes less sparse.

The proposed algorithm, dubbed *SParse Audio DEclipper (SPADE)*, comes in two flavors. The pseudocodes for the synthesis version ("*S-SPADE*") and for the analysis version ("*A-SPADE*") are given in Algorithm 1 and Algorithm 2.

Algorithm 1 S-SPADE

Require: $\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{M}_{\mathrm{r}}, \boldsymbol{M}_{\mathrm{c}}^{+}, \boldsymbol{M}_{\mathrm{c}}^{-}, \mathrm{s}, \mathrm{r}, \varepsilon$

1: $\hat{\boldsymbol{z}}^{(0)} = \boldsymbol{D}^{\mathrm{H}}\boldsymbol{y}, \boldsymbol{u}^{(0)} = \boldsymbol{0}, \mathrm{i} = 1, \mathrm{k} = \mathrm{s}$
2: $\bar{\boldsymbol{z}}^{(\mathrm{i})} = \mathcal{H}_{\mathrm{k}}\left(\hat{\boldsymbol{z}}^{(\mathrm{i}-1)} + \boldsymbol{u}^{(\mathrm{i}-1)}\right)$
3: $\hat{\boldsymbol{z}}^{(\mathrm{i})} = \arg\min_{\boldsymbol{z}} \|\boldsymbol{z} - \bar{\boldsymbol{z}}^{(\mathrm{i})} + \boldsymbol{u}^{(\mathrm{i}-1)}\|_2^2$ s.t. $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{z} \in \Gamma$
4: **if** $\|\hat{\boldsymbol{z}}^{(\mathrm{i})} - \bar{\boldsymbol{z}}^{(\mathrm{i})}\|_2 \leq \varepsilon$ **then**
5: terminate
6: **else**
7: $\boldsymbol{u}^{(\mathrm{i})} = \boldsymbol{u}^{(\mathrm{i}-1)} + \hat{\boldsymbol{z}}^{(\mathrm{i})} - \bar{\boldsymbol{z}}^{(\mathrm{i})}$
8: $\mathrm{i} \leftarrow \mathrm{i} + 1$
9: **if** $i \bmod \mathrm{r} = 0$ **then**
10: $\mathrm{k} \leftarrow \mathrm{k} + \mathrm{s}$
11: **end if**
12: go to 2
13: **end if**
14: **return** $\hat{\boldsymbol{x}} = \boldsymbol{D}\hat{\boldsymbol{z}}^{(\mathrm{i})}$

Algorithm 2 A-SPADE

Require: $\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{M}_{\mathrm{r}}, \boldsymbol{M}_{\mathrm{c}}^{+}, \boldsymbol{M}_{\mathrm{c}}^{-}, \mathrm{s}, \mathrm{r}, \varepsilon$

1: $\hat{\boldsymbol{x}}^{(0)} = \boldsymbol{y}, \boldsymbol{u}^{(0)} = \boldsymbol{0}, \mathrm{i} = 1, \mathrm{k} = \mathrm{s}$
2: $\bar{\boldsymbol{z}}^{(\mathrm{i})} = \mathcal{H}_{\mathrm{k}}\left(\boldsymbol{A}\hat{\boldsymbol{x}}^{(\mathrm{i}-1)} + \boldsymbol{u}^{(\mathrm{i}-1)}\right)$
3: $\hat{\boldsymbol{x}}^{(\mathrm{i})} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \bar{\boldsymbol{z}}^{(\mathrm{i})} + \boldsymbol{u}^{(\mathrm{i}-1)}\|_2^2$ s.t. $\boldsymbol{x} \in \Gamma$
4: **if** $\|\boldsymbol{A}\hat{\boldsymbol{x}}^{(\mathrm{i})} - \bar{\boldsymbol{z}}^{(\mathrm{i})}\|_2 \leq \varepsilon$ **then**
5: terminate
6: **else**
7: $\boldsymbol{u}^{(\mathrm{i})} = \boldsymbol{u}^{(\mathrm{i}-1)} + \boldsymbol{A}\hat{\boldsymbol{x}}^{(\mathrm{i})} - \bar{\boldsymbol{z}}^{(\mathrm{i})}$
8: $\mathrm{i} \leftarrow \mathrm{i} + 1$
9: **if** $i \bmod \mathrm{r} = 0$ **then**
10: $\mathrm{k} \leftarrow \mathrm{k} + \mathrm{s}$
11: **end if**
12: go to 2
13: **end if**
14: **return** $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}^{(\mathrm{i})}$

The relaxation rate and the relaxation stepsize are controlled by the integer-valued parameters $\mathrm{r} > 0$ and $\mathrm{s} > 0$, while the parameter $\varepsilon > 0$ is the stopping threshold.

Lemma 1 *The SPADE algorithms terminate in no more than* $\mathrm{i} = \lceil \mathrm{dr}/\mathrm{s} + 1 \rceil$ *iterations.*

Proof. Once $\mathrm{k} \geq \mathrm{d}$, the hard thresholding operation $\mathcal{H}_{\mathrm{k}}$ becomes an identity mapping. Then, the minimizer of the constrained least squares step 3 is $\hat{\boldsymbol{z}}^{(\mathrm{i}-1)}$ (respectively, $\hat{\boldsymbol{x}}^{(\mathrm{i}-1)}$) and the distance measure in the step 4 is equal to $\|\boldsymbol{u}^{(\mathrm{i}-1)}\|_2$. But, in the subsequent iteration, $\boldsymbol{u}^{(\mathrm{i}-1)} = \boldsymbol{0}$ and the algorithm terminates.

This bound is quite pessimistic: in practice, we observed that the algorithm terminates much sooner, which suggest that there might be a sharper upper bound on the iteration count.

4 Computational aspects

The general form of the *SPADE* algorithms does not impose restrictions on the choice of the dictionary nor the analysis operator. From a practical perspective, however, it is important that the complexity per iteration is kept low. The

dominant cost of *SPADE* is in the evaluation of the linearly constrained least squares minimizer step, whose computational complexity can be generally high. Fortunately, for some choices of $\boldsymbol{D}$ and $\boldsymbol{A}$ this cost is dramatically reduced.

Namely, if the matrix $\boldsymbol{A}^{\mathsf{H}}$ forms a *tight frame* ($\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \zeta\boldsymbol{I}$), it is easy to show that the step 3 of A-SPADE reduces to[2]:

$$\boldsymbol{x}^{(\mathrm{i})} = \mathcal{P}_{\Xi}\left(\frac{1}{\zeta}\boldsymbol{A}^{\mathsf{H}}(\bar{\boldsymbol{z}}^{(\mathrm{i})} - \boldsymbol{u}^{(\mathrm{i}-1)})\right), \text{ where:}$$
$$\Xi = \{\boldsymbol{x} \mid \begin{bmatrix} -\boldsymbol{M}_{\mathrm{c}}^{+} \\ \boldsymbol{M}_{\mathrm{c}}^{-} \end{bmatrix} \boldsymbol{x} \leq \begin{bmatrix} -\boldsymbol{M}_{\mathrm{c}}^{+} \\ \boldsymbol{M}_{\mathrm{c}}^{-} \end{bmatrix} \boldsymbol{y} \text{ and } \boldsymbol{M}_{\mathrm{r}}\,\boldsymbol{x} = \boldsymbol{M}_{\mathrm{r}}\,\boldsymbol{y}\}.$$

The projection $\mathcal{P}_{\Xi}(\cdot)$ is straightforward and corresponds to component-wise mappings, thus the per iteration cost of the algorithm is reduced to the cost of evaluating matrix-vector products.

On the other hand, for S-SPADE this simplification is not possible and the constrained minimization in step 3 needs to be computed iteratively. However, by exploiting the tight frame property of $\boldsymbol{D} = \boldsymbol{A}^{\mathsf{H}}$ and the Woodbury matrix identity, one can build an efficient algorithm that solves this optimization problem with low complexity.

Finally, the computational cost can be further reduced if the matrix-vector products with $\boldsymbol{D}$ and $\boldsymbol{A}$ can be computed with less than quadratic cost. Some transforms that support both tight frame property and fast product computation are also favorable in our audio (co)sparse context. Such well-known transforms are Discrete Fourier Transform, (Modified) Discrete Cosine Transform, (Modified) Discrete Sine Transform and Discrete Wavelet Transform, for instance.

5 Experiments

The experiments are aimed to highlight differences in signal enhancement performance between S-SPADE and A-SPADE, and implicitly, the sparse and cosparse data models. It is noteworthy that in the formally equivalent setting ($\boldsymbol{A} = \boldsymbol{D}^{-1}$), the two algorithms become identical. As a sanity-check, we include this setting in the experiments. The relaxation parameters are set to $\mathrm{r} = 1$ and $\mathrm{s} = 1$, and the stopping threshold is $\varepsilon = 0.1$.

In addition to *SPADE* algorithms, we also include Consistent IHT [15] and social sparsity declipping algorithm [21] as representatives of state-of-the-art. The latter two algorithms use the sparse synthesis data model for regularizing the declipping inverse problem. Consistent IHT is a low-complexity algorithm based on famous Iterative Hard Thresholding for Compressed Sensing [4], while the social sparsity declipper is based on a structured sparsity prior [16].

As mentioned before, this work is not aimed towards investigating the appropriateness of various time-frequency transforms in the context of audio recovery, which is why we choose traditional Short Time Fourier Transform (STFT) for all experiments. We use sliding square-rooted Hamming window of size 1024

[2] Recall that the matrices $\boldsymbol{M}_{\mathrm{r}}$, $\boldsymbol{M}_{\mathrm{c}}^{+}$ and $\boldsymbol{M}_{\mathrm{c}}^{-}$ are tight frames by design.

samples with 75% overlap. The redundancy level of the involved frames (corresponding to *per-chunk* inverse DFT for the dictionary and forward DFT for the analysis operator) is 1 (no redundancy), 2 and 4. The social sparsity declipper, based on Gabor dictionary, requires batch processing of the whole signal. We adjusted the temporal shift, the window and the number of frequency bins in accordance with previously mentioned STFT settings [3].

For a measure of performance, we use a simple difference between signal-to-distortion ratios of clipped ($\mathrm{SDR}_{\boldsymbol{y}}$) and processed ($\mathrm{SDR}_{\hat{\boldsymbol{x}}}$) signals:

$$\mathrm{SDR}_{\boldsymbol{y}} = 20\log_{10}\frac{\|\left[\begin{smallmatrix}\boldsymbol{M}_{\mathrm{c}}^{+}\\ \boldsymbol{M}_{\mathrm{c}}^{-}\end{smallmatrix}\right]\boldsymbol{x}\|_2}{\|\left[\begin{smallmatrix}\boldsymbol{M}_{\mathrm{c}}^{+}\\ \boldsymbol{M}_{\mathrm{c}}^{-}\end{smallmatrix}\right]\boldsymbol{x}-\left[\begin{smallmatrix}\boldsymbol{M}_{\mathrm{c}}^{+}\\ \boldsymbol{M}_{\mathrm{c}}^{-}\end{smallmatrix}\right]\boldsymbol{y}\|_2}, \mathrm{SDR}_{\hat{\boldsymbol{x}}} = 20\log_{10}\frac{\|\left[\begin{smallmatrix}\boldsymbol{M}_{\mathrm{c}}^{+}\\ \boldsymbol{M}_{\mathrm{c}}^{-}\end{smallmatrix}\right]\boldsymbol{x}\|_2}{\|\left[\begin{smallmatrix}\boldsymbol{M}_{\mathrm{c}}^{+}\\ \boldsymbol{M}_{\mathrm{c}}^{-}\end{smallmatrix}\right]\boldsymbol{x}-\left[\begin{smallmatrix}\boldsymbol{M}_{\mathrm{c}}^{+}\\ \boldsymbol{M}_{\mathrm{c}}^{-}\end{smallmatrix}\right]\hat{\boldsymbol{x}}\|_2}$$

Hence, only the samples corresponding to clipped indices are taken into account. Concerning *SPADE*, this choice makes no difference, since the remainder of the estimate $\hat{\boldsymbol{x}}$ perfectly fits the observations $\boldsymbol{y}$. However, it may favor the other two algorithms that do not share this feature.

Audio examples consist of 10 music excerpts taken from RWC database [10], which significantly differ in tonal and vocal content. The excerpts are of approximately similar duration ($\sim$ 10s), and are sampled at 16kHz with 16bit encoding. The inputs are generated by artificially clipping the audio excerpts at five levels, ranging from severe ($\mathrm{SDR}_{\boldsymbol{y}} = 1$dB) towards mild ($\mathrm{SDR}_{\boldsymbol{y}} = 10$dB).

According to the results presented in figure 1, the *SPADE* algorithms yield highest improvement in SDR among the four considered approaches. As assumed, S-SPADE and A-SPADE achieve similar results in a non-redundant setting, but when the overcomplete frames are considered, the synthesis version performs somewhat better. Interestingly, the overall best results for the analysis version are obtained for the twice-redundant frame, while the performance slightly drops for the redundancy four. This is probably due to the absolute choice of the parameter ε, and suggests that in the analysis setting, this value should be replaced by a relative threshold instead. In the non-redundant case, declipping by A-SPADE and Consistent IHT took (on the average) 3min and 7min, respectively, while the other two algorithms were much slower[4] (on the order of hours).

6 Conclusion

We presented a novel algorithm for non-convex regularization of the declipping inverse problem. The algorithm is flexible in terms that it can easily accommodate sparse (S-SPADE) or cosparse (A-SPADE) prior, and as such has been used to compare the recovery performance of the two data models. The empirical results are slightly in favor of the sparse synthesis data model. However, the analysis version does not fall far behind, which makes it attractive for practical applications. Indeed, due to the natural way of imposing clipping consistency

[3] We use the implementation kindly provided by the authors.

[4] All algorithms were implemented in Matlab®, and run in single-thread mode.

constraints, it can be implemented in an extremely efficient way, even allowing for a real-time signal processing. Benchmark on real audio data demonstrates that both versions outperform considered state-of-the-art algorithms in the field.

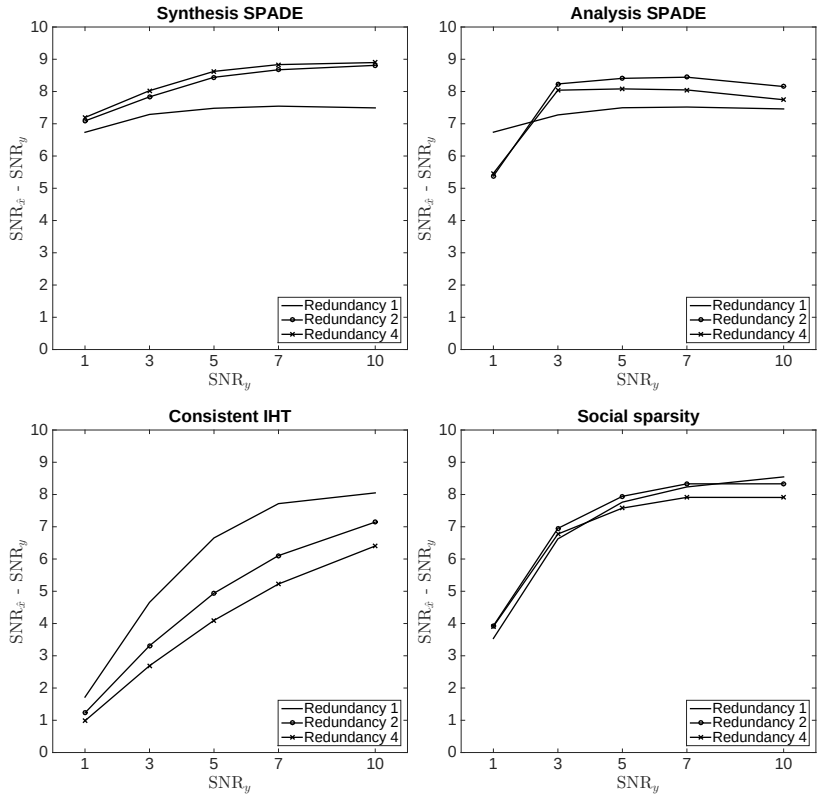

Fig. 1. Declipping performance in terms of the SDR improvement.

Future work will be dedicated to theoretical analysis of the algorithm, with emphasis on convergence. A possible extension is envisioned by introducing structured (co)sparsity priors in the presented algorithmic framework.

References

1. A. Adler, V. Emiya, M. G. Jafari, M. Elad, R. Gribonval, and M. D. Plumbley. Audio inpainting. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):922–932, 2012.
2. T. O. Aydin, R. Mantiuk, K. Myszkowski, and H. Seidel. Dynamic range independent image quality assessment. In *ACM Transactions on Graphics (TOG)*, volume 27, page 69. ACM, 2008.

3. D. P. Bertsekas. *Nonlinear programming.* Athena scientific Belmont, 1999.
4. T. Blumensath and M. E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.
5. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
6. B. Defraene, N. Mansour, S. De Hertogh, T. van Waterschoot, M. Diehl, and M. Moonen. Declipping of audio signals using perceptual compressed sensing. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(12):2627–2637, 2013.
7. M. Elad, P. Milanfar, and R. Rubinstein. Analysis versus synthesis in signal priors. *Inverse problems*, 23(3):947, 2007.
8. Y. C. Eldar and G. Kutyniok. *Compressed sensing: theory and applications.* Cambridge University Press, 2012.
9. S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing.* Springer, 2013.
10. M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical and jazz music databases. In *ISMIR*, volume 2, pages 287–288, 2002.
11. M. J. Harvilla and R. M. Stern. Least squares signal declipping for robust speech recognition. In *INTERSPEECH*, 2014.
12. A. Janssen, R. Veldhuis, and L. Vries. Adaptive interpolation of discrete-time signals that can be modeled as autoregressive processes. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(2):317–330, 1986.
13. M. Kahrs and K. Brandenburg. *Applications of digital signal processing to audio and acoustics*, volume 437. Springer Science & Business Media, 1998.
14. S. Kitić, N. Bertin, and R. Gribonval. Audio declipping by cosparse hard thresholding. In *iTwist-2nd international-Traveling Workshop on Interactions between Sparse models and Technology.*
15. S. Kitić, L. Jacques, N. Madhu, M. P. Hopwood, A. Spriet, and C. De Vleeschouwer. Consistent iterative hard thresholding for signal declipping. In *IEEE ICASSP*, pages 5939–5943. IEEE, 2013.
16. M. Kowalski, K. Siedenburg, and M. Dorfler. Social sparsity! neighborhood systems enrich structured shrinkage operators. *Signal Processing, IEEE Transactions on*, 61(10):2498–2511, 2013.
17. X. Li and L. J. Cimini. Effects of clipping and filtering on the performance of OFDM. In *47th IEEE Vehicular Technology Conference*, volume 3, pages 1634–1638. IEEE, 1997.
18. S. K. Naik and C. A. Murthy. Hue-preserving color image enhancement without gamut problem. *IEEE Transactions on Image Processing*, 12(12):1591–1598, 2003.
19. S. Nam, M. E. Davies, M. Elad, and R. Gribonval. The cosparse analysis model and algorithms. *Applied and Computational Harmonic Analysis*, 34(1):30–56, 2013.
20. M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies. Sparse representations in audio and music: from coding to source separation. *Proceedings of the IEEE*, 98(6):995–1005, 2010.
21. K. Siedenburg, M. Kowalski, and M. Dorfler. Audio declipping with social sparsity. In *IEEE ICASSP*, pages 1577–1581. IEEE, 2014.
22. Y. Tachioka, T. Narita, and J. Ishii. Speech recognition performance estimation for clipped speech based on objective measures. *Acoustical Science and Technology*, 35(6):324–326, 2014.
23. A. J. Weinstein and M. B. Wakin. Recovering a clipped signal in sparseland. *arXiv preprint arXiv:1110.5063*, 2011.