Model-Based Inversion of Dynamic Range Compression

Stanislaw Gorlow, Joshua D. Reiss

Reverse Audio Engineering: Model-Based Inversion of Dynamic Range Compression

Stanislaw Gorlow, *Graduate Student Member, IEEE* and Joshua D. Reiss, *Member, IEEE*

***Abstract*—Reverse audio engineering so far has been quite an understudied research topic, which is attested by the irrefutable fact that existing literature on that particular subject is scarce. In the present work it is shown how a dynamic nonlinear time-variant operator, such as a digital compressor, can be inverted in a mathematical sense using an explicit data model. Knowing the model parameters which were used during compression, one is competent to recover the original (uncompressed) signal from the broadcast signal with high numerical accuracy and also low computational complexity. A complete compressor-decompressor scheme is worked out and described in detail, and the proposed approach is further tested on synthetic signals for some typical compressor settings with considerable success.**

***Index Terms*—Dynamic range compression, model-based inversion, reverse audio engineering.**

I. Introduction

SOUND or *audio engineering* is an established discipline employed in many areas that are part of our everyday life without us taking notice of it. Most of us watch television or listen to the radio and even more use the Web as the number one source for music and entertainment. But not many know how the audio was produced. If we take sound recording and reproduction or broadcasting as an example, we may imagine that a prerecorded signal from an acoustic source is altered by an audio engineer in such a way that it corresponds to certain criteria when played back. The number of these criteria may be large and usually depends on the context. In general, the said alteration of the input signal is a sequence of numerous forward transformations, the reversibility of which is of little or no interest. But what if one wished to do exactly this, that is to reverse the transformation chain, and what is more, in a systematic and repeatable manner? Literature on that issue is few and far between. More research that deals with the topic of *reverse audio engineering* (RAE) is therefore needed.

The research objective of RAE can be twofold: to identify the transformation parameters given the input and the output signals, as in [1], or else to regain the input signal that goes with the output signal given the transformation parameters. In both cases, an explicit signal model is mandatory. The latter case might seem trivial, but only if the applied transformation is linear and orthogonal and as such perfectly invertible. Yet enough times the forward transform is neither linear nor has it an inverse. This is the case for *dynamic range compression* (DRC) which is commonly described by a dynamic nonlinear time-variant system. The classical linear time-invariant (LTI) system theory at large does not apply here, so that a tailored solution to the problem at hand must be found instead.

Although a minimum amount of side information to invert dynamics compression is promised by the title, the solution in [2] requires an instantaneous gain value to be transmitted for each sample of the compressed signal. To provide a means to control the data rate, the gain signal is subsampled and also entropy coded. Not relying on a gain model and thus being extremely generic, this approach is highly inefficient, simply because transmitting the uncompressed signal in conjunction with some few typical compression parameters like threshold, ratio, attack, and release requires a much smaller capacity on the one hand and yields the best possible signal quality with regard to any thinkable measure on the other hand.

Nonetheless, the more realistic scenario is without a doubt when the uncompressed signal is not available on consumer side. This is usually the case for studio music recordings and broadcast material. There, the listener is offered a signal that is meant to sound "good" to everyone. The loudness war [3] however had an adverse effect: Most of, if not all, latter-day audio material is over-compressed. Over-compression makes a song lose its artistic features like excitingness or liveliness and desensitizes the ear thanks to a louder volume. There are imaginably quite a few good reasons why somebody would appreciate the option to restore the original signal's dynamic range and to experience audio free of compression.

In addition to the normalization of the program's loudness level, the Dolby solution [4], [5] includes also dynamic range expansion. The expansion parameters that help reproduce the the original program's dynamic range are tuned on broadcaster side and transmitted as metadata together with the broadcast signal. This is a very convenient solution for broadcasters not least because the metadata is quite compact. Dynamic range expansion is yet another forward transformation rather than a true inversion.

Evidently, none of the previous approaches satisfy the RAE objective as it was formulated earlier. The goal of the present work, hence, is to *invert* dynamic range compression, which is a vital element not only in broadcasting but also in mastering.

This work was partially funded by the "Agence Nationale de la Recherche" (ANR) within the scope of the DReaM project (ANR-09-CORD-006) as well as the laboratory with which the first author is affiliated (see below) as part of the "mobilité juniors" program.

S. Gorlow is with the Computer Science Research Laboratory of Bordeaux (LaBRI), CNRS, University of Bordeaux 1, 33405 Talence Cedex, France (e-mail: stanislaw.gorlow@labri.fr).

J. Reiss is with the Centre for Digital Music (C4DM), Queen Mary, University of London, London E1 4NS, UK (e-mail: josh.reiss@eecs.qmul.ac.uk).

The remaining part of the paper looks as follows: Section II provides a brief introduction to the basics of dynamic range compression and presents the compressor model upon which my considerations are based. The data model, the formulation of the problem, and the pursued approach are described next in Section III. The inversion is discussed in detail in Section IV. Section V illustrates how an integral step of the inversion procedure, namely the search for the zero-crossing of a nonlinear function, can be solved in an iterative manner by means of linearization. The complete algorithm is given in the form of pseudocode in Section VI and its performance is evaluated for different compressor settings in Section VII. Conclusions are drawn in Section VIII and some thinkable directions for future work are also mentioned.

II. Dynamic Range Compression

Dynamic range compression or simply "compression" is a sound processing technique that attenuates loud sounds and/or amplifies quiet sounds, which in consequence leads to a reduction of an audio signal's dynamic range[1]. In what follows, I will use the word "compression" having "downward" compression in mind.[2] Downward compression means attenuating sounds above a certain threshold while leaving sounds below the threshold unchanged. Using digital signal processing, compression is applied in the form of an algorithm which often emulates an equivalent analog technology. Such an algorithm is called a "compressor". A sound engineer might use a compressor to reduce the dynamic range of source material for purposes of aesthetics, intelligibility, recording or broadcast limitations. Fig. 1 illustrates the *basic* compressor model from [6, ch. 2] which is moreover compatible with the compressor/limiter model in [7, ch. 4]. I will hereafter restrict my considerations to this basic model, as the purpose of the present work is to demonstrate a general approach rather than a solution to a specific problem—although a solution for the model under consideration is also offered.

The *broadband* compressor in Fig. 1 operates in the following manner: First, the input signal is split and a copy is sent to the *side chain*. The sidechain signal, alias "key" or "trigger", is therefore deemed to be always equal to the input signal. The *detector* then calculates the magnitude or level of the sidechain signal using the *root mean square* (RMS) or *peak* as a measure for how loud a sound is [7, ch. 4]. The sound level is then compared with the *threshold* level, and in case it exceeds the threshold a scale factor is calculated which corresponds to the *ratio*. The *knee* parameter determines how quick the compression ratio is reached. At the end of the side chain, the scale factor is fed to a *smoothing filter* that yields the gain. The response of the filter is controlled by the *attack* and *release* parameters. Finally, the *gain control* applies the smoothed gain to the input signal and adds a fixed amount of *makeup gain* to bring the output signal to a desired level. Such a broadband compressor operates on the input signal's full bandwidth, treating all frequencies from zero through the highest frequency equally. A detailed overview of all sidechain controls of a basic *gain computer* is given in [6, ch. 3].

At this point, I would like to highlight the fact that neither Volterra nor Wiener model approaches [8] are applicable to the compressor model from Fig. 1, and neither are describing functions [9], [10]. These are useful tools when identifying a *time-invariant* nonlinear system or analyzing the limit cycle behavior of a *feedback* system with a static nonlinearity. Yet none of the italicized criteria apply to the referenced model.

III. Data Model, Problem Formulation, and Proposed Solution

A. Data Model

The used data model is based upon the compressor from Fig. 1. The following simplifications are additionally made: the knee parameter is ignored ("hard" knee) and so is the makeup gain (fixed at 0 dB). The compressor is further deemed to be a single-input single-output (SISO) system, that is both the input and the output are mono. What follows is a description of each block by means of a dedicated function.

The RMS/peak detector builds upon a first-order (one-pole) lowpass filter given by the recurrence relation

$$v(n) = \alpha u(n) + \bar{\alpha} v(n-1), \tag{1}$$

where α is a *non-zero* smoothing factor, $0 < \alpha \leqslant 1$, and $\bar{\alpha} \triangleq 1-\alpha$. u and v represent the input and the output, respectively, and n is the discrete-time index. Using (1) the sound level or envelope $e(n)$ is obtained by

$$\begin{aligned} \hat{x}(n) &= \beta |x(n)|^p + \bar{\beta}\hat{x}(n-1) \\ e(n) &= \sqrt[p]{\hat{x}(n)} \end{aligned} \qquad \text{with } p \in \{1, 2\}. \tag{2}$$

If $p = 2$ one speaks of an RMS detector or else of a peak detector if $p = 1$. The smoothing factor β may further take on different values, depending on whether the detector is in *attack* or *release phase*. A formula that converts a time constant into a corresponding smoothing factor can be found in [7, ch. 4]. The static nonlinearity in the gain computer is usually modeled in the logarithmic domain as a continuous piecewise linear function:

$$F(n) = \begin{cases} -\mathrm{SLP} \cdot [E(n) - \mathrm{THR}] & \text{if } E(n) > \mathrm{THR}, \\ 0 & \text{otherwise}, \end{cases} \tag{3}$$

where SLP is the *slope*, $E(n) = 20 \log_{10} e(n)$, and THR is the threshold in decibel. The slope is further derived from the desired compression ratio RAT according to [7, ch. 4]:

$$\mathrm{SLP} = 1 - \frac{1}{\mathrm{RAT}}. \tag{4}$$

Note that with a compression ratio of $1:1$ the slope in the second piece of (3) is zero, and so is F. Also, compression sets in only when the envelope exceeds the threshold. Equation (3) can be equivalently expressed in the linear domain as

$$f(n) = \begin{cases} K e^{-\mathrm{SLP}}(n) & \text{if } e(n) > thr, \\ 1 & \text{otherwise}, \end{cases} \tag{5}$$

[1] In audio, dynamic range is the difference between the loudest and quietest sound measured in decibel.

[2] The discussed approach is likewise applicable to upward compression.

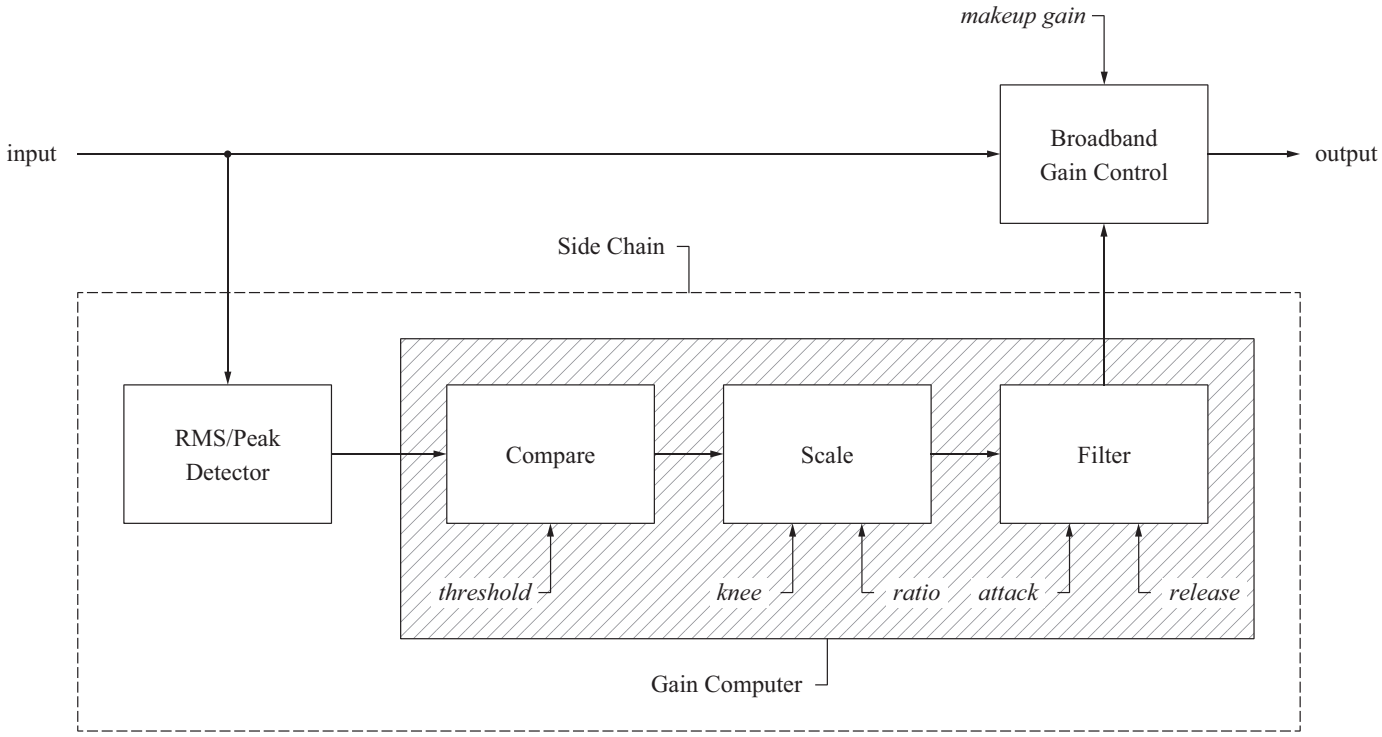

Fig. 1. Basic broadband compressor model (feed forward)

where $K = 10^{\mathrm{SLP\,THR}/20}$, $thr = 10^{\mathrm{THR}/20}$, and f is the scale factor before filtering. The smoothed gain g is then calculated as the exponentially-weighted moving average as in (1),

$$g(n) = \gamma f(n) + \bar{\gamma} g(n-1). \tag{6}$$

The output signal is finally obtained by multiplying the above gain with the input signal

$$y(n) = g(n) \cdot x(n). \tag{7}$$

B. Problem Formulation

Due to the fact that the gain g is strictly positive, $0 < g \leqslant 1$, it follows that

$$|y(n)| = g(n) \cdot |x(n)| \tag{8}$$

and further

$$\mathrm{sgn}(y) = \mathrm{sgn}(x), \tag{9}$$

where sgn is the signum or sign function. In consequence, it is convenient to factorize the input signal as a product of the sign and the modulus according to

$$x(n) = \mathrm{sgn}(x) \cdot |x(n)| \tag{10}$$

with $\mathrm{sgn}(x)$ being known due to (9). The problem to be solved can hence be formulated as follows: Given the compressed signal $y(n)$ and the model parameters $\boldsymbol{\theta}$, recover the modulus of the original signal $|x(n)|$ from $|y(n)|$ based on $\boldsymbol{\theta}$ using the one-sample history of $\hat{x}$ and g, that is $\hat{x}(n-1)$ and $g(n-1)$, respectively.

C. Proposed Solution

The output of the side chain, that is the gain of $|x(n)|$, given $\boldsymbol{\theta}$, $\hat{x}(n-1)$, and $g(n-1)$, may be written as

$$g(n) = G[|x(n)| \mid \boldsymbol{\theta}, \hat{x}(n-1), g(n-1)]. \tag{11}$$

In (11), G denotes a *nonlinear dynamic* operator that maps the modulus of the input signal $|x(n)|$ onto a sequence of instantaneous gain values $g(n)$ according to the compressor model represented by $\boldsymbol{\theta}$. The compressor shall be completely described by the input parameters listed below.

TYP	The detector type (corresponds to p)
ATT_1	The attack time of the envelope filter in ms
REL_1	The release time of the envelope filter in ms
THR	The threshold in dB
RAT	The compression ratio (input : output)
ATT_2	The attack time of the gain filter in ms
REL_2	The release time of the gain filter in ms

Using (11), (8) can be solved for $|x(n)|$ yielding

$$|x(n)| = G^{-1}[g(n) \mid \boldsymbol{\theta}, \hat{x}(n-1), g(n-1)] \cdot |y(n)| \tag{12}$$

subject to invertibility of G. In order to solve the above equation one requires the knowledge of $g(n)$, which is unavailable. However, since g is a function of $|x|$, we can express $|y|$ in (8) as a function of *one* independent variable $|x|$, and in that manner we obtain an equation with a single unknown:

$$|y(n)| = H[|x(n)| \mid \boldsymbol{\theta}, \hat{x}(n-1), g(n-1)], \tag{13}$$

where H represents the entire compressor. If H is invertible, i.e. *bijective* for all n, $|x(n)|$ can be obtained from $|y(n)|$ by

$$|x(n)| = H^{-1}[|y(n)| \mid \boldsymbol{\theta}, \hat{x}(n-1), g(n-1)], \tag{14}$$

or more explicitly:

$$|x(n)| = \begin{cases} H^{-1}[|y(n)| \mid \boldsymbol{\theta} \dots] & \text{if } e(n) > thr, \\ |y(n)| & \text{otherwise.} \end{cases} \tag{15}$$

And yet, since $e(n)$ is unknown, the condition for applying decompression must be predicted from $y(n)$, $\hat{x}(n-1)$, and $g(n-1)$, and so needs the condition for toggling between the attack and release phases. Depending on the quality of the prediction, the recovered modulus may differ somewhat at transition points from the original modulus, so that in the end

$$x(n) \approx \operatorname{sgn}(y) \cdot H^{-1}[|y(n)| \mid \boldsymbol{\theta}, \hat{x}(n-1), g(n-1)]. \tag{16}$$

In the next section it is shown how such an *inverse compressor* or *decompressor* is derived.

IV. Inversion of Dynamic Range Compression

A. Characteristic Function

For reason of a somewhat simpler calculus, let us choose the instantaneous envelope value $e(n)$ instead of $|x(n)|$ as the independent variable in (13); the relation between the two items is given by (2). We start out from (8):

$$\begin{aligned} |y(n)| &= g(n) \cdot |x(n)| \\ &\overset{(6)}{=} [\gamma f(n) + \bar{\gamma} g(n-1)] \cdot |x(n)| \qquad (17) \\ &\overset{(5)}{=} \left[\gamma K e^{-\mathrm{SLP}}(n) + \bar{\gamma} g(n-1)\right] \cdot |x(n)|. \qquad (18) \end{aligned}$$

At this point we solve (2) for $|x(n)|$ in terms of $e(n)$ and plug the result in into (18), which yields

$$\begin{aligned} |y(n)| = \left[\gamma K e^{-\mathrm{SLP}}(n) + \bar{\gamma} g(n-1)\right] \\ \times \sqrt[p]{\left[e^p(n) - \bar{\beta}\hat{x}(n-1)\right]/\beta}, \end{aligned} \tag{19}$$

or equivalently (note that $\beta \neq 0$ by definition)

$$\begin{aligned} \beta |y(n)|^p = \left[\gamma K e^{-\mathrm{SLP}}(n) + \bar{\gamma} g(n-1)\right]^p \\ \times \left[e^p(n) - \bar{\beta}\hat{x}(n-1)\right]. \end{aligned} \tag{20}$$

Moreover, (20) has a unique solution if G and also H are invertible. Moving the expression on the left-hand side over to the right-hand side, we may define

$$\begin{aligned} z_p(e) \triangleq \left[\gamma K e^{-\mathrm{SLP}}(n) + \bar{\gamma} g(n-1)\right]^p \\ \times \left[e^p(n) - \bar{\beta}\hat{x}(n-1)\right] - \beta |y(n)|^p, \end{aligned} \tag{21}$$

which shall be termed the *characteristic function*. The *zero-crossing* of $z_p(e)$ hence represents the sought-after envelope value $e(n)$. Once $e(n)$ is found (see Section V), the current values of $\hat{x}$, $|x|$, and g are updated as per

$$\begin{aligned} \hat{x}(n) &= e^p(n) \\ |x(n)| &= \sqrt[p]{\left[\hat{x}(n) - \bar{\beta}\hat{x}(n-1)\right]/\beta} \\ g(n) &= |y(n)|/|x(n)| \end{aligned} \tag{22}$$

and the decompressed sample is then calculated as

$$x(n) = \operatorname{sgn}(y) \cdot |x(n)|. \tag{23}$$

B. Attack-Release Phase Toggle

1) Envelope Smoothing: When a peak detector is in use, β takes on two different values. The condition for the attack phase is $|x(n)| > e(n-1)$ [7, ch. 4], or equivalently

$$|x(n)|^p \overset{!}{>} \hat{x}(n-1). \tag{24}$$

Assuming that the past value of $\hat{x}$ is known at time n, what we need to do is to express the unknown $|x|$ in terms of $|y|$, such that the above equation still holds true. So let us recall (17) which says that

$$|y(n)| = [\gamma f(n) + \bar{\gamma} g(n-1)] \cdot |x(n)|. \tag{25}$$

If γ is very small—which is the usual case for the gain filter ($\gamma \sim 0.001 \ll 1$), the term $\gamma f(n)$ in (25) is negligible, so that in good approximation

$$|y(n)| \approx g(n-1) \cdot |x(n)|. \tag{26}$$

Solving (26) for $|x(n)|$ and plugging the result in into (24), we finally obtain

$$\left[\frac{|y(n)|}{g(n-1)}\right]^p > \hat{x}(n-1), \tag{27}$$

or equivalently

$$|y(n)| > \sqrt[p]{\hat{x}(n-1)} \cdot g(n-1). \tag{28}$$

If (28) is true, the detector is assumed to be in attack phase, or in release phase otherwise.

2) Gain Smoothing: Just like the peak detector, the gain smoothing filter may be in either attack or release phase. The condition for the attack phase is now $f(n) < g(n-1)$ [7, ch. 4], or also

$$e(n) \overset{!}{>} \left[\frac{K}{g(n-1)}\right]^{1/\mathrm{SLP}}. \tag{29}$$

But since the current envelope value is unknown, we need to substitute $e(n)$ in the above inequality by something that we know. With this in mind we rewrite (17) as

$$\begin{aligned} |y(n)| &= [\gamma f(n) + \bar{\gamma} g(n-1)] \cdot |x(n)| \\ &= \left[\gamma \frac{f(n)}{g(n-1)} + \bar{\gamma}\right] g(n-1) \cdot |x(n)| \\ &= \left[1 - \gamma\left(1 - \frac{f(n)}{g(n-1)}\right)\right] g(n-1) \cdot |x(n)|. \end{aligned} \tag{30}$$

Provided that $f(n) < g(n-1)$, and due to the fact that $0 < \gamma \leqslant 1$, the expression in square brackets in (30) is smaller than one, and thus during attack

$$|y(n)| < g(n-1) \cdot |x(n)|. \tag{31}$$

Substituting $|x(n)|$ by $\sqrt[p]{\left[e^p(n) - \bar{\beta}\hat{x}(n-1)\right]/\beta}$ using (22), and solving (31) for $e(n)$ results in

$$e(n) > \sqrt[p]{\beta\left[\frac{|y(n)|}{g(n-1)}\right]^p + \bar{\beta}\hat{x}(n-1)}. \tag{32}$$

If $e(n)$ in (29) is substituted by the expression on the right-hand side of (32), (29) still holds true, so that the following

sufficient condition is used to predict the attack phase of the gain filter:

$$\sqrt[p]{\beta\left[\frac{|y(n)|}{g(n-1)}\right]^p + \bar{\beta}\hat{x}(n-1)} > \left[\frac{K}{g(n-1)}\right]^{1/\mathrm{SLP}}. \tag{33}$$

Rearranging (33) for $|y(n)|$ finally yields

$$|y(n)| > \sqrt[p]{\frac{1}{\beta}\left\{\left[\frac{K}{g(n-1)}\right]^{p/\mathrm{SLP}} - \bar{\beta}\hat{x}(n-1)\right\}} \times g(n-1). \tag{34}$$

Note that the values of all variables are known whenever (34) is evaluated.

C. Envelope Predictor

An instantaneous estimate of the envelope value $e(n)$ is required not only to predict the period when compression is active, formally $e(n) > thr$, but also to initialize the iterative search algorithm in Section V. We resort once more to (17) and note that in the opposite case where $e(n) \leqslant thr$, $f(n) = 1$, and so

$$|x(n)| = \frac{|y(n)|}{\gamma + \bar{\gamma}g(n-1)}. \tag{35}$$

The sound level of the input signal at time n is therefore

$$e(n) = \sqrt[p]{\beta\left[\frac{|y(n)|}{\gamma + \bar{\gamma}g(n-1)}\right]^p + \bar{\beta}\hat{x}(n-1)}. \tag{36}$$

Using the expression from (36) we can formally state that the condition

$$\sqrt[p]{\beta\left[\frac{|y(n)|}{\gamma + \bar{\gamma}g(n-1)}\right]^p + \bar{\beta}\hat{x}(n-1)} > thr \tag{37}$$

must be fulfilled for compression to set in, or analogously

$$|y(n)| > \sqrt[p]{\frac{1}{\beta}\left[thr^p - \bar{\beta}\hat{x}(n-1)\right]} \cdot \left[\gamma + \bar{\gamma}g(n-1)\right] \tag{38}$$

with β and γ selected based on (28) and (34), respectively.

D. Stereo Linking

When dealing with stereo signals, one might want to apply the same amount of gain reduction to both channels to prevent image shifting. This is achieved through *stereo linking*. One way is to calculate the required amount of gain reduction for each channel independently and then apply the larger amount to both channels. The question which arises in this context is which of the two channels was the gain derived from. To give an answer resolving the dilemma of ambiguity, one thinkable solution would be to signal which of the channels carries the applied gain. One could then decompress the marked sample and use its gain for the other channel. Although very simple to implement, this approach provokes an additional data rate of 22 kbps at 44.1-kHz sampling. A rate-efficient alternative that comes with a higher computational cost is realized in the following way: First, one decompresses both the left and the right channel independently and in so doing one obtains two estimates $\tilde{x}_l(n)$ and $\tilde{x}_r(n)$, where subscript l shall denote the left channel and subscript r the right channel, respectively. In a second step, one calculates the compressed values of $\tilde{y}_l(n)$ and $\tilde{y}_r(n)$ which correspond to the estimates and selects the channel for which $\tilde{y}(n) = y(n)$ holds true. In a final step, one updates the remaining variables using the proper gain.

E. General Remarks

A compressor with a *look-ahead* function, i.e. with a delay unit in the main signal path as in [7, ch. 4], uses past input samples to calculate the output sample. For a causal system, however, all input samples are zero for negative time. Now that future samples are required to invert the process, which are all zero in the beginning, the inversion renders impossible. $g(n)$ and $x(n)$ must thus be in sync for the above approach to be applied.

Another point worth mentioning is that "hard" clipping and "brick-wall" limiting are special cases of compression with at least the attack time set to zero and the compression ratio set to $\infty : 1$. The static nonlinearity F, in that particular case, is a *one-to-many* mapping, which by definition is *non-invertible*.

V. Numerical Solution of the Characteristic Function

An approximate solution to the characteristic function can be found, e.g., by means of *linearization*. In that respect, we may approximate (21) at a given point using the equation of a straight line, $z = m \cdot e + c$, where m is the slope and c is the z-intercept. The estimate from (36) may moreover serve as a starting point for an iterative search of an optimum:

$$e_{\mathrm{init}} = \sqrt[p]{\beta\left[\frac{|y(n)|}{\gamma + \bar{\gamma}g(n-1)}\right]^p + \bar{\beta}\hat{x}(n-1)}. \tag{39}$$

The criterion for optimality is further chosen as the deviation of the characteristic function from zero initialized to

$$\Delta_{\mathrm{init}} = |z_p(e_{\mathrm{init}})|. \tag{40}$$

The slope and the z-intercept of the linear approximation at iteration i are then calculated as

$$\begin{aligned} m_i &= \frac{z_p(e_i + \Delta_i) - z_p(e_i)}{\Delta_i} \\ c_i &= z_p(e_i) \end{aligned} \tag{41}$$

with e_i as reference point. Quite evidently, the zero-crossing is characterized by the equation

$$z_i = \frac{z_p(e_i + \Delta_i) - z_p(e_i)}{\Delta_i} \cdot e + z_p(e_i) \stackrel{!}{=} 0, \tag{42}$$

as is shown in Fig. 2. The new and hopefully better estimate of the optimal e is hence found using (42) as

$$e_{i+1} = e_i - \frac{\Delta_i \cdot z_p(e_i)}{z_p(e_i + \Delta_i) - z_p(e_i)}. \tag{43}$$

If e_{i+1} is less optimal than e_i, the iteration is stopped and e_i is the final estimate. The iteration is also stopped, if Δ_{i+1} is smaller than some ϵ. In the latter case, e_{i+1} has the optimal

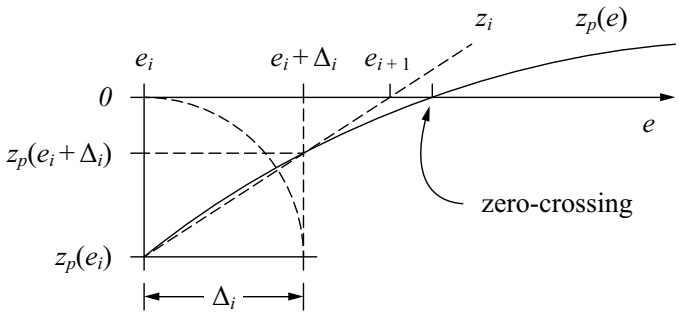

Fig. 2. Graphical illustration for the iterative search of the zero-crossing

Algorithm 1 The compressor

function COMP($x_n, \boldsymbol{\theta}, f_s$)
  $\hat{x}_n \leftarrow 0$
  $g_n \leftarrow 1$
  for $n \leftarrow 1, N$ **do**
    if $|x_n|^p > \hat{x}_n$ **then**
      $\beta \leftarrow 1 - \exp(-2.2/\mathrm{ATT}_1/f_s)$
    else
      $\beta \leftarrow 1 - \exp(-2.2/\mathrm{REL}_1/f_s)$
    end if
    $\hat{x}_n \leftarrow \beta|x_n|^p + \bar{\beta}\hat{x}_n$
    $e_n \leftarrow \sqrt[p]{\hat{x}_n}$
    if $e_n > thr$ **then**
      $f_n \leftarrow K e_n^{-\mathrm{SLP}}$
    else
      $f_n \leftarrow 1$
    end if
    if $f_n < g_n$ **then**
      $\gamma \leftarrow 1 - \exp(-2.2/\mathrm{ATT}_2/f_s)$
    else
      $\gamma \leftarrow 1 - \exp(-2.2/\mathrm{REL}_2/f_s)$
    end if
    $g_n \leftarrow \gamma f_n + \bar{\gamma} g_n$
    $y_n \leftarrow g_n x_n$
  end for
  return y_n
end function

value with respect to the chosen criteria. Otherwise, e_i is set to e_{i+1} and Δ_i is set to Δ_{i+1} after every iteration step and the procedure is repeated until e_{i+1} has converged to a more optimal value.

VI. The Algorithm

The complete algorithm is divided into three parts each of them given as pseudocode further below. Algorithm 1 outlines the compressor that corresponds to the model from Sections II–III. Algorithm 2 illustrates the decompressor described in Section IV, and the iterative search from Section V is finally summarized in Algorithm 3.

Algorithm 2 The decompressor

function DECOMP($y_n, \boldsymbol{\theta}, \epsilon, f_s$)
  $\hat{x}_n \leftarrow 0$
  $g_n \leftarrow 1$
  for $n \leftarrow 1, N$ **do**
    if $|y_n| > \sqrt[p]{\hat{x}_n} \cdot g_n$ **then**
      $\beta \leftarrow 1 - \exp(-2.2/\mathrm{ATT}_1/f_s)$
    else
      $\beta \leftarrow 1 - \exp(-2.2/\mathrm{REL}_1/f_s)$
    end if
    if $|y_n| > \sqrt[p]{\left[(K/g_n)^{p/\mathrm{SLP}} - \bar{\beta}\hat{x}_n\right]/\beta} \cdot g_n$ **then**
      $\gamma \leftarrow 1 - \exp(-2.2/\mathrm{ATT}_2/f_s)$
    else
      $\gamma \leftarrow 1 - \exp(-2.2/\mathrm{REL}_2/f_s)$
    end if
    if $|y_n| > \sqrt[p]{\left(thr^p - \bar{\beta}\hat{x}_n\right)/\beta} \cdot (\gamma + \bar{\gamma} g_n)$ **then**
      $e_n \leftarrow \sqrt[p]{\beta[|y_n|/(\gamma + \bar{\gamma} g_n)]^p + \bar{\beta}\hat{x}_n}$
      $e_\mathrm{o} \leftarrow \mathrm{CHARFZERO}(e_n, \epsilon)$
      $|x_n| \leftarrow \sqrt[p]{\left(e_\mathrm{o}^p - \bar{\beta}\hat{x}_n\right)/\beta}$
      $\hat{x}_n \leftarrow e_\mathrm{o}^p$
      $g_n \leftarrow |y_n|/|x_n|$
    else
      $g_n \leftarrow \gamma + \bar{\gamma} g_n$
      $|x_n| \leftarrow |y_n|/g_n$
      $\hat{x}_n \leftarrow \beta|x_n|^p + \bar{\beta}\hat{x}_n$
    end if
    $x_n \leftarrow \mathrm{sgn}(y_n) \cdot |x_n|$
  end for
  return x_n
end function

Algorithm 3 The iterative search of the zero-crossing

function CHARFZERO(e_n, ϵ)
  $e_i \leftarrow e_n$
  repeat
    $\Delta_i \leftarrow |z_p(e_i)|$
    $e_i \leftarrow e_i - \Delta_i \cdot z_p(e_i)/[z_p(e_i + \Delta_i) - z_p(e_i)]$
    if $|z_p(e_i)| > \Delta_i$ **then**
      return e_n
    end if
    $e_n \leftarrow e_i$
  until $|z_p(e_i)| < \epsilon$
  return e_i
end function

VII. Performance Evaluation

A. Performance Metrics

For the purpose of evaluation of the inverse approach, the following quantities were measured: the ℓ^2 error norm given in decibel relative to full scale (dBFS) and the execution time of the decompressor relative to realtime (RT). In addition, the number of iterations required for the search to converge to a satisfactory solution was also tracked. The computations were run on an Intel Core i5-520M processor using MATLAB.

B. Computational Results

Fig. 3 shows the inverse output signal $z(n)$ for a synthetic input signal $x(n)$ using an RMS detector. The inverse signal was derived from the output signal $y(n)$ with high accuracy, i.e. an error as low as -84.5 dBFS. The error values as well as execution times and iteration numbers for white Gaussian noise were computed on top of this for different compressor settings, among which the threshold was fixed at -20 dBFS and the ratio was invariably set to $4:1$. The noise signal was zero-mean with $\sigma = 0.25$. The ϵ value in the break condition of Algorithm 3 was further set to 1×10^{-12}. A summary of compressor settings and corresponding results is provided in Tables I–II.

As can be seen from Table II, the largest inversion error is associated with column F, followed by column D. There, all time constants are nonzero, which makes up a very dynamic compression behavior. Also, if an RMS detector is in use, the characteristic function has a stronger curvature in comparison to the peak detector, which explains the difference between F and D. By choosing the distance in (40) as $\Delta = \sqrt[p]{|z_p(e)|}$, it is yet possible to obtain a similar error value for F as for D, -63 dBFS to be exact, at the expense of a somewhat longer runtime: 0.53 RT with 1.84 iterations on average. Moreover, comparing the columns A, B, and C, one may speculate that the toggle lever between the attack and release phases works more reliably for the gain smoothing filter than for the level detector. The choice of the time constants also seems to have an impact on decompressor's accuracy.

Fig. 4 depicts the error surface as a function of attack and release times for both the peak and the RMS detector. It can be observed that the shorter is the attack time chosen for the envelope filter, the smaller is the committed error. This is the case not only for the peak but also the RMS detector over a wide range of attack time values; see (a) and (c). Just about the opposite is the case for the gain filter in (b) and (d): after reaching a local peak, the error decreases as the attack or the release time becomes longer. The release time seems to be of little relevance in (a) and (c), whereas a kind of "resonance" peaks appear in (b) for certain release time values. However, their origin remains a mystery at this point. In (d), the error descent is smoother and less sensitive to the attack time.

VIII. Conclusion

By the present work I have displayed how an inverse to a nonlinear dynamic operator such as a digital compressor can be derived by the help of elementary mathematics and basic rationale. The presented approach is characterized by the fact that it uses an explicit signal model to solve the problem. To find the "dry" or uncompressed signal with high accuracy, it is sufficient to know the model parameters, which may be as few as seven in number or even less. These parameters must be transmitted together with the "wet" or compressed signal as side information or estimated from the compressed signal, which might be a possible direction for future work. Another could be to apply the approach to more sophisticated models that include a soft knee, parallel and multiband compression, or perform gain smoothing in the logarithmic domain.

In conclusion, I want to draw the reader's attention to the fact that the presented figures suggest that the decompressor is realtime capable which can pave the way for exciting new applications.

Acknowledgment

This work was carried out in part at the Centre for Digital Music (C4DM).

References

[1] D. Barchiesi and J. Reiss, "Reverse engineering of a mix," *J. Audio Eng. Soc.*, vol. 58, no. 7/8, pp. 563–576, Jul. 2010.

[2] B. Lachaise and L. Daudet, "Inverting dynamics compression with minimal side information," in *Proc. DAFx-08*, Sept. 2008.

[3] E. Vickers, "The loudness war: Background, speculation and recommendations," in *AES Convention 129*, Nov. 2010.

[4] *Dolby Digital and Dolby Volume Provide a Comprehensive Loudness Solution*, Dolby Laboratories, Jan. 2007.

[5] *Broadcast Loudness Issues: The Comprehensive Dolby Approach*, Dolby Laboratories, 2011.

[6] R. Jeffs, S. Holden, and D. Bohn, *Dynamics Processor — Technology & Application Tips*, Rane Corporation, 2005.

[7] U. Zölzer, *DAFX: Digital Audio Effects*, 2nd ed. John Wiley & Sons, 2011.

[8] T. Ogunfunmi, *Adaptive Nonlinear System Identification: The Volterra and Wiener Model Approaches*. Springer, 2007.

[9] A. Gelb and W. E. Vander Velde, *Multiple-Input Describing Functions and Nonlinear System Design*. McGraw Hill, 1968.

[10] P. W. J. M. Nuij, O. H. Bosgra, and M. Steinbuch, "Higher-order sinusoidal input describing functions for the analysis of non-linear systems with harmonic responses," *Mech. Syst. Signal Process.*, vol. 20, no. 8, pp. 1883–1904, 2006.

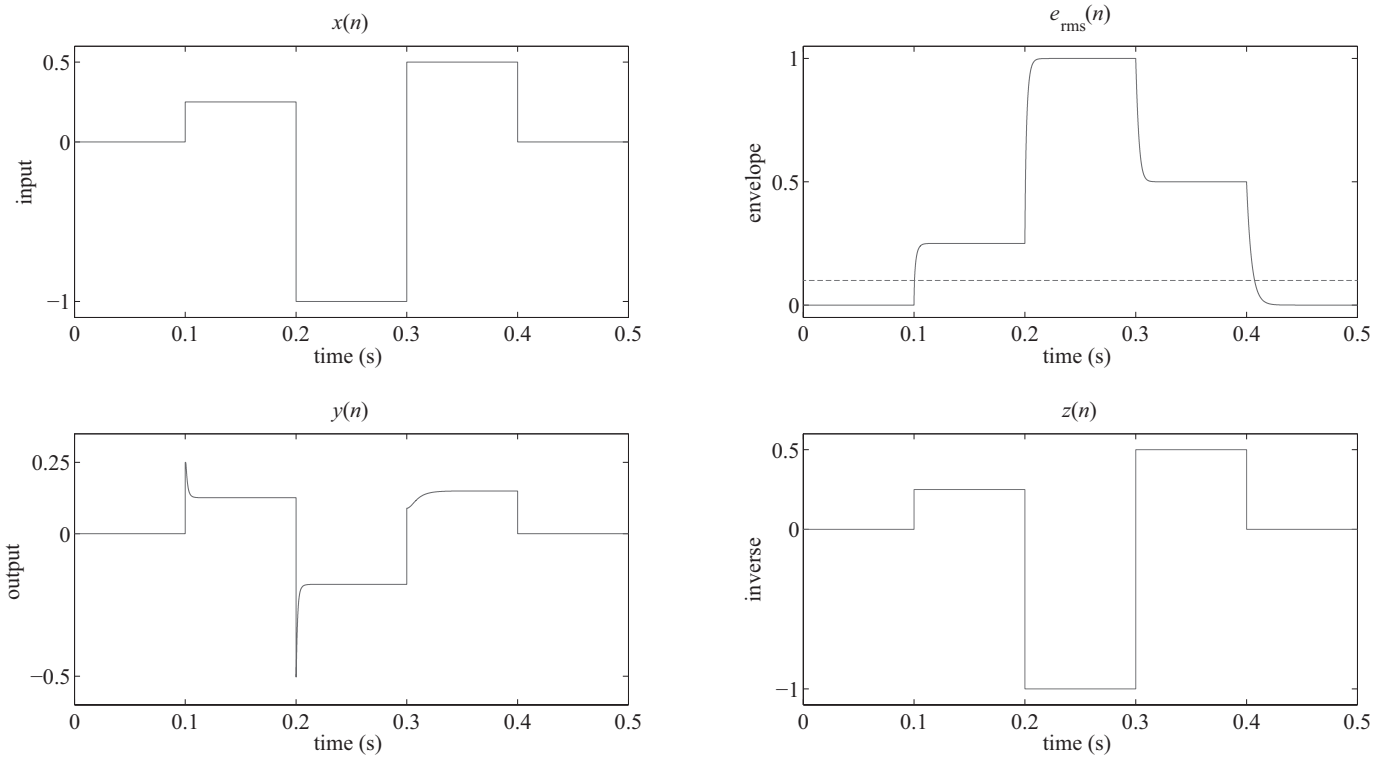

Fig. 3. An illustrative example using an RMS detector with β set to 0.01, a threshold of -20 dBFS (dashed line in the upper right corner), a compression ratio of $4:1$, and the gain smoothing factor γ set to 0.03 for attack and 0.003 for release, respectively.

TABLE I
COMPRESSOR SETTINGS USED IN THE EXPERIMENT (THRESHOLD AND RATIO FIXED)

	A	B	C	**D**	E	**F**
TYP	peak	peak	peak	**peak**	RMS	**RMS**
ATT_1	0 ms	0 ms	0.14 ms	**0.14 ms**	4.96 ms	**4.96 ms**
REL_2	0 ms	0 ms	4.96 ms	**4.96 ms**	4.96 ms	**4.96 ms**
ATT_1	0 ms	1.64 ms	0 ms	**1.64 ms**	0 ms	**1.64 ms**
REL_2	0 ms	16.6 ms	0 ms	**16.6 ms**	0 ms	**16.6 ms**

TABLE II
RESULTS FOR WHITE GAUSSIAN NOISE UNDER DIFFERENT SETTINGS (100 RUNS)

	A	B	C	**D**	E	**F**
Mean inversion error (dBFS)	-195	-179	-159	$\mathbf{-62}$	-134	$\mathbf{-49}$
Mean execution time (RT)	0.65	0.40	0.61	**0.41**	0.59	**0.41**
Mean number of iterations	4.26	1.98	2.56	**1.25**	2.42	**1.22**
Maximum number of iterations	5	2	5	**2**	4	**2**

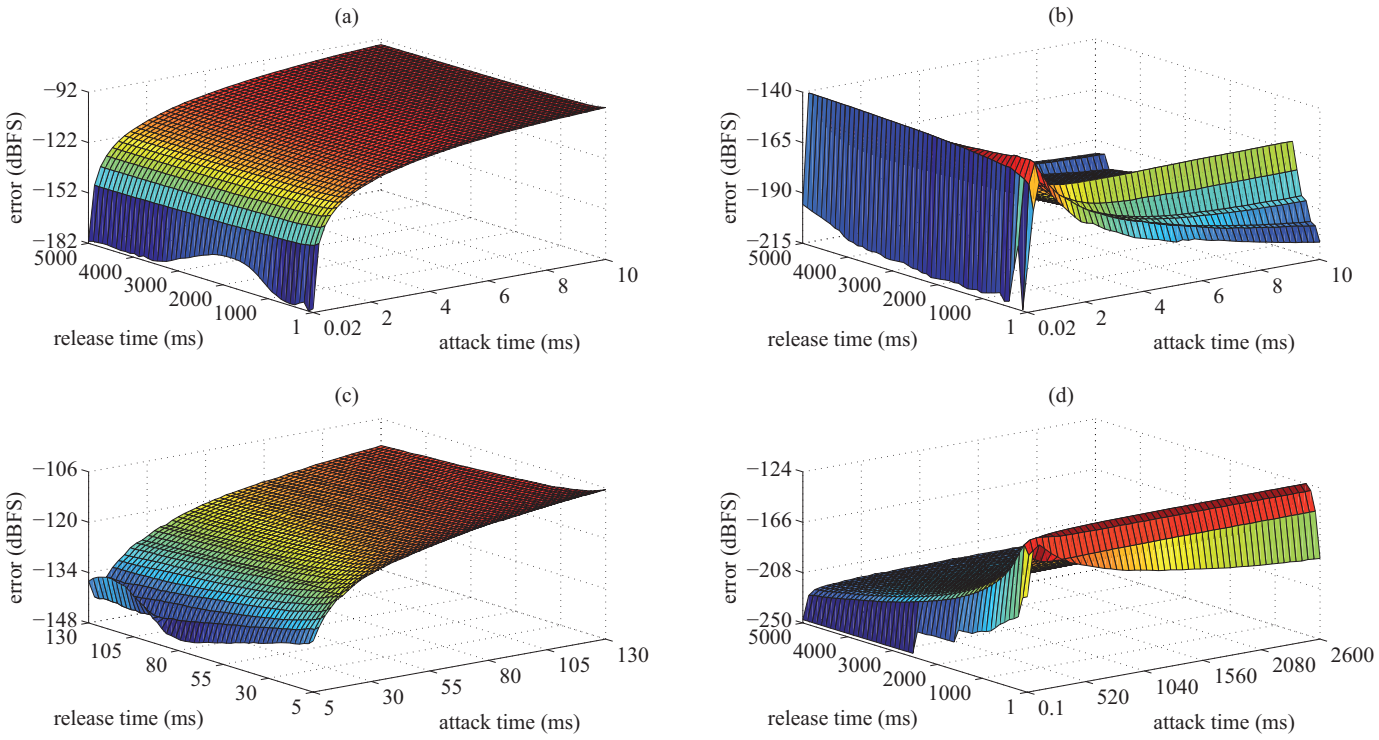

Fig. 4. Error surface as a function of typical attack and release times using the peak (upper row) and the RMS (lower row) detector. In the left column, the time constants of the envelope filter are varied while the time constants of the gain filter are fixed at zero. The right column shows the reverse case.