PARAMETRIC APPROACH TO BLIND DECONVOLUTION OF NONLINEAR CHANNELS

Jordi SOLE (1), Christian JUTTEN (2), Anisse TALEB (2, 3)
(1) University of Vic, Sagrada Família 7, 08500, Vic (Catalunya, Spain)
(2) INPG, LIS, 46 Avenue Félix Viallet, 38031 Grenoble Cedex, France
(3) ATRI-Curtin Univ. of Technology, GPO Box U1987, PERTH WA 6848 Australia

Abstract

A parametric procedure for the blind inversion of nonlinear channels is proposed, based on a recent method of blind source separation in nonlinear mixtures. Experiments show that the proposed algorithms perform efficiently, even in the presence of hard distortion. The method, based on the minimization of the output mutual information, needs the knowledge of log-derivative of input distribution (the so-called *score function*). Each algorithm consists of three adaptive blocks: one devoted to adaptive estimation of the score function, and two other blocks estimating the inverses of the linear and nonlinear parts of the channel, (quasi-)optimally adapted using the estimated score functions. This paper is mainly concerned by the nonlinear part, for which we propose two parametric models, the first based on a polynomial model and the second on a neural network, while [14, 15] proposed non-parametric approaches.

1 Introduction

When linear models fail, nonlinear models appear to be powerful tools for modeling practical situations. Many researches have been done in the identification and/or the inversion of nonlinear systems. These works assume that both the input and the output of the distortion are available [13]; they are based on higher-order input/output cross-correlation [3], bispectrum estimation [11, 12] or on the application of the Bussgang and Prices theorems [4, 10] for nonlinear systems with Gaussian inputs. However, in a real world situations, one often does not have access to the distortion input. In this case, blind identification of the nonlinearity becomes the only way to solve the problem. This paper is concerned by a particular class of nonlinear systems, composed by a linear filter followed by a memoryless nonlinear distortion (figure 1, left). This class of nonlinear systems, also known as a Wiener system, is a nice and mathematically

attracting model, but also a realistic model used in various areas, such as biology [8], industry [2], sociology and psychology (see also [9] and the references therein). Despite its interest, today, there does not exist fully blind procedure for inverting such systems.

In this paper, we proposed a fully blind inversion method inspired of recent advances in source separation of nonlinear mixtures. Although deconvolution can be viewed as a single input/single output (SISO) source separation problem in convolutive mixtures (which are consequently not cited in this paper), the current approach is actually very different. It is mainly based on equivalence between instantaneous postnonlinear mixtures and Wiener systems, provided a well-suited parameterization. The paper is organized as follows. The Wiener model and its parameterization are described in Section 2. The cost function based on statistical independence is derived in Section 3. The gradient of the cost function with respect to the inversion structure parameters are detailed in Section 4, according two parametric (polynomial and neural) models. Section 5 contains a few simulation results. In Section 6, we discuss on what happen if the input signal is not i.i.d., before concluding in Section 7.

2 Model and assumptions

We suppose that the input of the system $S=\{s(t)\}$ is an unknown non-Gaussian independent and identically distributed (i.i.d.) process, and that subsystems h, f are a linear filter and a memoryless nonlinear function, respectively, both unknown and invertible. We would like to estimate $s(t)$ by only observing the system output. This implies the blind estimation of the inverse structure (figure 1, right), composed of similar subsystems: a memoryless nonlinear function g followed by a linear filter w. Such a system is known as a Hammerstein system. Let $\mathbf{s}$ and $\mathbf{e}$ be the vectors of infinite dimension, whose t-th entries are $s(t)$ or $e(t)$, respectively. The unknown input-output transfer can be written as:

$$\boldsymbol{e} = f\left(\mathbf{H}\boldsymbol{s}\right) \tag{1}$$

where:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & h(t+1) & h(t) & h(t-1) & \cdots \\ \cdots & h(t+2) & h(t+1) & h(t) & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \tag{2}$$

is an infinite dimension Toeplitz matrix which represents the action of the filter h to the signal $s(t)$. The matrix $\mathbf{H}$ is non-singular provided that the filter h is invertible, i.e. satisfies $h^{-1}*h(t) = h*h^{-1}(t) = \delta(t)$, where $\delta(t)$ is the Dirac impulse. The infinite dimension of vectors and matrix is due to the lack of assumption on the filter order. If the filter h is a finite impulse response (FIR) filter of order N_h, the matrix dimension can be reduced to the size N_h. Practically, because infinite-dimension equations are not tractable, we have to choose a pertinent (finite) value for N_h.
Equation (1) corresponds to a post-nonlinear (pnl) model [14]. This model has been recently studied in nonlinear source separation, but only for a finite dimensional case. In fact, with the above parameterization, the i.i.d. nature of $s(t)$ implies the spatial independence of the components of the infinite vector s. Similarly, the output of the inversion structure can be written $\boldsymbol{y} = \mathbf{W}\boldsymbol{x}$ with $x(t) = g(e(t))$. Following [14, 15] the inverse system (g, w) can be estimated by minimizing the output mutual information, i.e. spatial independence of $\boldsymbol{y}$ which is equivalent to the i.i.d. nature of $y(t)$.

3 Cost function

The mutual information of a random vector of dimension n, defined by

$$I(\mathbf{z}) = \sum_{i=1}^{n} H(z_i) - H(z_i, z_2, ..., z_n) \tag{3}$$

can be extended to a vector of infinite dimension, using the notion of *entropy rates* of stationary stochastic processes [6]

$$I(Z) = \lim_{T\to\infty} \frac{1}{2T+1} \left\{ \sum_{t=-T}^{T} H(z(t)) - H(z(-T), ..., z(T)) \right\} = H(z(t)) - H(Z) \tag{4}$$

where t is arbitrary due to the stationarity assumption. We can notice that $I(Z)$ is always positive and vanishes iff $z(t)$ is i.i.d. Since S is stationary, and h and w are time-invariant filters, then Y is stationary too, and $I(Y)$ is defined by

$$I(Y) = H(y(t)) - H(Y) \tag{5}$$

Using the Lemma 1 of [15], the last right term of equation (5) becomes

$$H(Y) = H(X) + \frac{1}{2p} \int_0^{2p} \log \left| \sum_{t=-\infty}^{+\infty} w(t) e^{-jtq} \right| dq \tag{6}$$

Moreover, using $x(t)=g(e(t))$ and the stationarity of $E=\{e(t)\}$:

$$H(X)=\lim_{T\to\infty}\frac{1}{2T+1}\left\{H(e(-T),...,e(T))+\sum_{t=-T}^{T}E[\log g'(e(t))]\right\}= \\ =H[E]+E[\log g'(e(t))] \quad (7)$$

Combining (6) and (7) in (5) leads finally to:

$$I(Y)=H(y(t))-\frac{1}{2p}\int_0^{2p}\log\left|\sum_{t=-\infty}^{+\infty}w(t)e^{-jtq}\right|dq-E[\log g'(e(t))]-H[E] \quad (8)$$

4 Theoretical derivation of the inversion algorithm

For deriving the optimization algorithm, the derivatives of *I*(*Y*) (8), with respect to the parameters of the linear part *w* and of the nonlinear function *g*, are required.

4.1 Linear subsystem

The linear subsystem is parameterized by the coefficients of the filter *w*. The derivative of *I*(*Y*) with respect to the coefficient *w*(*t*), corresponding to the *t*-th lag of the filter *w*, is computed as follow (see [15] for details).

The derivative of the first right side term of (8) is:

$$\frac{\partial H(y(t))}{\partial w(t)}=-E\left[\frac{p'_Y(y(t))}{p_Y(y(t))}\frac{\partial(w*x)(t)}{\partial w(t)}\right]=-E[y_Y(y(t))x(t-t)] \quad (9)$$

where $y_Y(u)=\frac{p'_Y}{p_Y}(u)$ is the so-called *score* function.

The derivative of the second term is:

$$\frac{\partial}{\partial w(t)}\left\{\frac{1}{2p}\int_0^{2p}log\left|\sum_{t'=-\infty}^{+\infty}w(t')e^{-jt'q}\right|dq\right\}=\frac{1}{2p}\int_0^{2p}\frac{1}{W(e^{jq})}e^{-jtq}dq \quad (10)$$

Clearly, this second term is the {-*t*}-th coefficient of the inverse filter of *w*. In the following, it will be noted $\overline{w}(-t)$.

The two last terms of (8) does not depends on *w*, then their contribution to the gradient of *I*(*Y*) with respect to *w* is zero. Combining (9) and (10) leads to:

$$\frac{\partial I(Y)}{\partial w(t)} = -E\left[x(t-t)y_Y(y(t))\right] - \overline{w}(-t) \qquad (11)$$

From (11), we could deduce a simple gradient algorithm, but we prefer derive the *natural* or *relative* gradient descent algorithms [1, 5], which exhibit uniform performance. Denoting $g_{y,y_Y(y)} = E\left[yy_Y(y)\right]$ the high order cross-correlation between y and $y_Y(y)$, the natural gradient, derived from (11), leads to the symbolic algorithm

$$w \leftarrow w + m\left\{g_{y,y_Y(y)} + d\right\} * w \qquad (12)$$

More details on this algorithm are given in [15]. Moreover, an interactive JAVA simulator is available on the author's Web pages, whose address is provided at the end of the paper. Matlab routines can also be downloaded from this server.

4.2 Nonlinear subsystem

For this subsystem, we propose two different parametric models for the function *g*. The first one is based on a polynomial model of *g*, and the second one on a 1-hidden layer multilayer perceptron (MLP).

4.2.1 Polynomial parameterization

Modeling *g* with a N-degree polynomial leads to:

$$g(t) = \sum_{n=1}^{N+1} a_n t^{n-1} \qquad (13)$$

From figure 1 and (13), we obtain:

$$x(t) = g(e(t)) = \sum_{n=1}^{N+1} a_n e^{n-1}(t) \qquad (14)$$

$$y(t) = (w * x)(t) = \sum_{i=-\infty}^{\infty} w(i)g(e(t-i)) \qquad (15)$$

The gradient descent algorithm for estimating g requires the derivatives of $I(Y)$ (5) with respect to the coefficients a_n of the polynomial. The derivatives of the right-side term of (5), with respect of the p-th coefficient, are:

$$\frac{\partial H(y(t))}{\partial a_p} = -E\left[\frac{\partial}{\partial a_p} log\ p_Y(y(t))\right] = -E\left[y_Y(y(t))\frac{\partial y(t)}{\partial a_p}\right] = \\ = -E\left[y_Y(y(t))\sum_{t=-\infty}^{\infty} w(t)e(t-t)^{p-1}\right] \tag{16}$$

and:

$$\frac{\partial H(Y)}{\partial a_p} = \frac{\partial}{\partial a_p} E\left[log\left(\sum_{n=1}^{N+1} a_n(n-1)e(t)^{n-2}\right)\right] = E\left[\left(\sum_{n=1}^{N+1} a_n(n-1)e(t)^{n-2}\right)^{-1}\frac{\partial g(e(t))}{\partial a_p}\right] = \\ = E\left[\left(\sum_{n=1}^{N+1} a_n(n-1)e(t)^{n-2}\right)^{-1}(p-1)e(t)^{p-2}\right] \tag{17}$$

where $y_Y(u) = \frac{p'_Y}{p_Y}(u)$ is the so-called *score* function.

Combining (16) and (17) leads to:

$$\frac{\partial I(Y)}{\partial a_p} = -E\left[y_Y(y(t))\sum_{t=-\infty}^{+\infty} w(t)e(t-t)^{p-1}\right] - E\left[\left(\sum_{n=1}^{N+1} a_n(n-1)e(t)^{n-2}\right)^{-1}(p-1)e(t)^{p-2}\right] \tag{18}$$

Equation (18) is the gradient of $I(Y)$ with respect to polynomial coefficients a_p, and will be used for estimating g according to the gradient descent algorithm:

$$a \leftarrow a - m\frac{\partial I(Y)}{\partial a} \tag{19}$$

4.2.2 Neural network parameterization

In this subsection, we model g using a multilayer perceptron with one hidden layer:

$$g[\boldsymbol{a},\boldsymbol{c},\boldsymbol{b},u]=\sum_{i=1}^{N}a_i s\left(c_i u-b_i\right) \tag{20}$$

and from figure 1 we obtain:

$$x(t)=g(e(t))=\sum_{i=1}^{N}a_i s\left(c_i e(t)-b_i\right) \tag{21}$$

$$y(t)=(w*x)(t)=\sum_{i=-\infty}^{\infty}w(i)g(e(t-i)) \tag{22}$$

The gradient descent algorithm of g requires the derivatives of $I(Y)$ (5) with respect to the network parameters **a**, **c** and **b**:

a) the derivatives of $H(Y)$ with respect to the various parameters are given below:

- with respect of a_p:

$$\frac{\partial H(Y)}{\partial a_p}=\frac{\partial}{\partial a_p}E[log(g'(e(t)))]=$$

$$=E\left[\frac{1}{\sum_{i=1}^{N}a_i c_i s'(c_i e(t)-b_i)}\frac{\partial}{\partial a_p}g'(e(t))\right]=E\left[\frac{c_p s'(c_p e(t)-b_p)}{\sum_{i=1}^{N}a_i c_i s'(c_i e(t)-b_i)}\right] \tag{23}$$

- with respect of c_p:

$$\frac{\partial H(Y)}{\partial c_p}=E\left[\frac{a_p\left(e(t)c_p s''(c_p e(t)-b_p)+s'(c_p e(t)-b_p)\right)}{\sum_{i=1}^{N}a_i c_i s'(c_i e(t)-b_i)}\right] \tag{24}$$

- with respect of b_p:

$$\frac{\partial H(y)}{\partial b_p}=E\left[\frac{-a_p c_p s''(c_p e(t)-b_p)}{\sum_{i=1}^{N}a_i c_i s'(c_i e(t)-b_i)}\right] \tag{25}$$

b) the derivatives of $H(y(t))$ with respect to the various parameters are given below:

- with respect of a_p:

$$\frac{\partial H(y(t))}{\partial a_p} = -E\left[\frac{\partial}{\partial a_p} log\left(p_Y(y(t))\right)\right] = -E\left[y_Y(y(t))\frac{\partial y(t)}{\partial a_p}\right] =$$
$$= -E\left[y_Y(y(t))\frac{\partial}{\partial a_p}\sum_{t=-\infty}^{\infty} w(t)\left(\sum_{i=1}^{N} a_i s\left(c_i e(t-t)-b_i\right)\right)\right] =$$
$$= -E\left[y_Y(y(t))\sum_{t=-\infty}^{\infty} w(t)s\left(c_p e(t-t)-b_p\right)\right] \tag{26}$$

- with respect of c_p:

$$\frac{\partial H(y(t))}{\partial c_p} = -E\left[y_Y(y(t))\sum_{t=-\infty}^{\infty} w(t)a_p e(t-t)s'\left(c_p e(t-t)-b_p\right)\right] \tag{27}$$

- with respect of b_p:

$$\frac{\partial H(y(t))}{\partial b_p} = +E\left[y_Y(y(t))\sum_{t=-\infty}^{\infty} w(t)a_p s'\left(c_p e(t-t)-b_p\right)\right] \tag{28}$$

By combining these derivatives, we obtain the gradient of the mutual information $I(Y)$ (5) with respect to the network coefficients:

$$\frac{\partial I(Y)}{\partial a_p} = -E\left[\frac{c_p s'\left(c_p e(t)-b_p\right)}{\sum_{i=1}^{N} a_i c_i s'\left(c_i e(t)-b_i\right)}\right] - E\left[y_Y(y(t))\sum_{t=-\infty}^{\infty} w(t)s\left(c_p e(t-t)-b_p\right)\right] \tag{29}$$

$$\frac{\partial I(Y)}{\partial c_p} = -E\left[\frac{a_p\left(e(t)c_p s''\left(c_p e(t)-b_p\right)+s'\left(c_p e(t)-b_p\right)\right)}{\sum_{i=1}^{N} a_i c_i s'\left(c_i e(t)-b_i\right)}\right] -$$
$$- E\left[y_Y(y(t))\sum_{t=-\infty}^{\infty} w(t)a_p e(t-t)s'\left(c_p e(t-t)-b_p\right)\right] \tag{30}$$

$$\frac{\partial I(Y)}{\partial b_p} = E\left[\frac{a_p c_p s''(c_p e(t) - b_p)}{\sum_{i=1}^{N} a_i c_i s'(c_i e(t) - b_i)}\right] + E\left[y_Y(y(t)) \sum_{t=-\infty}^{\infty} w(t) a_p s'(c_p e(t-t) - b_p)\right] \quad (31)$$

Equations (29-31) will be used for deriving gradient descent algorithms for estimating the MLP, which models *g*:

$$a \leftarrow a - m_a \frac{\partial I(Y)}{\partial a}\ ,\ c \leftarrow c - m_c \frac{\partial I(Y)}{\partial c}\ ,\ b \leftarrow b - m_b \frac{\partial I(Y)}{\partial b} \quad (32)$$

5. Algorithm

5.1. Practical issues

For implementing efficient algorithms, a few practical points must be addressed:

- order of filter *w*,
- normalisation,
- score function estimations,
- cross-correlation estimations.

As infinite dimension filters are not tractable, we constrain *w* as a FIR filter of order $N_w = 2N_0 + 1$. Without assumptions on the orders of causal and noncausal parts, we choose a symmetrical filter with N_0 entries for the causal part, N_0 entries for the strictly causal part. Then, at the beginning of the algorithm, the initialisation of *w* is $w(t) = d(t - N_0 - 1)$.

The solutions based on independence cannot be unique. In fact, independence is satisfied if $(g \circ f)(u) = au, \forall a$ and if $w(t) * h(t) = bd(t - t), \forall b, t$. For cancelling indeterminacies, one must add two normalisations, consisting to scaling to one the signal power, just after $g(.)$ and after *w*. We also normalize the filter so that the central coefficent is equal to one, according to $w(t) \leftarrow \frac{w(t)}{w(N_0 + 1)}$. The delay indeterminacy is cancelled by initializing the filter as explained above.

Many methods can be used for estimating the score functions (first step in the main loop). Although direct estimation is possible [16], in this paper we used indirect estimation, based on the well known kernel estimators [7] of the density and of its derivative.

Finally, the cross-correlation $g_{y,y_Y(y)} = Eyy_Y(y)$, used in (12), is estimated according to an empirical mean.

5.2 Algorithm

Denoting $E = \{e(1), e(2), ..., e(T)\}$ the observation sequence, the blind inversion algorithm based on the polynomial model (and similarly based on the MLP model) writes as:

Require E: $e(t)$, t=1,..T

Initialization

$g(.) = 1$

$w(t) = d(t - N_0 - 1)$

Main loop

do

estimation of the *score* function, $y_Y(y)$

estimation of the gradient, $\frac{\partial I(Y)}{\partial a_n}$, according to (18), for each n

updating of a_n, according to (19), for each n

output of the nonlinear subsystem: $x(t) = \sum a_n e^n(t)$

normalization of $x(t)$

empirical estimation of cross-correlation $g_{y,y_Y(y)}$

updating of the filter coefficients according to (12)

normalisation of the filter

computation of the current output $y(t)$

normalisation of $y(t)$

until convergence

5.3. Experiments

In order to prove the efficiency of the previous algorithms, we consider an input i.i.d. random sequence $s(t)$, filtered by a non-minimum phase FIR filter h=[–0.082, 0, –0.1793, 0, 0.6579, 0, –0.1793, 0, –0.082] and then distorted with $f(u) = 0.1u + \tanh(5u)$ (see figure 2). Note on figure 2-right, the saturation due to the function tanh(·) and which strongly distorts the input signal. The frequency response of h (figure 3) emphasizes on the non-minimum phase nature of h.

The algorithm was trained with a sample size of T = 1000. The length of the impulse response of w is set to 21 with equal length for the causal and anti-causal parts. Estimation results, shown in figures 4 and 5, prove the efficiency of the two algorithms (with polynomial or neural models) for estimating the inverse of nonlinear function f.

The performance can be directly measured with the output signal noise ratio $S/N = E\left[y^2(t)\right] / E\left[(s(t) - y(t))^2\right]$, where N is the error power and S is the estimated signal power. After adequate processing (delaying and re-scaling of $y(t)$), one obtains $S/N \approx +18dB$ with both polynomial and neural models.

6 Discussion

In the previous experiments, the input signal was an i.i.d. random sequence. However, in real world situations, we usually handle non i.i.d. signals. What will happen with our inversion method if the input signal is no longer i.i.d.? In such a situation, we will recover an i.i.d. version of the original, i.e. the innovation process of the original signal, if it exists. In fact, supposing that the signal can be modeled as an i.i.d. sequence filtered with an AR filter, we can merge the AR filter and the channel filter in a single filter, as shown in figure 6. Then, the inversion system will recover the inverse of the cascade, and the output of the inversion system will be the i.i.d. sequence at the input of the AR filter, i.e. the so-called innovation process $s_r(t)$ instead $s(t)$.

This result is validated by another experiment (figure 8), using (as input signal $s(t)$) a non i.i.d. signal, obtained by filtering an i.i.d. random sequence, $s_r(t)$, with an AR filter. As expected, the inverse filter w estimated by the method is the inverse of the cascade $\frac{h(z)}{AR(z)}$, and its output is the innovation process generating the data, as showed in figure 8. In all figures, the spectrums are computed with FFT Matlab function with 1024 points, and only one of the two symmetric parts is showed.

If we know the AR model of the signal $s(t)$, we can easily recover this signal using from the i.i.d. output of the inversion system, $y(t) \approx s_r(t)$, by filtering $y(t)$ with the AR filter (figure 7). Figure 8-c shows the recovered signal spectrum, after this post-filtering, becomes very close to the spectrum of $s(t)$.

In the next example we present a result obtained with a real music signal, which is not an i.i.d. signal. We use the first 1000 samples for estimating the inverse of the Wiener system. In fact, due to the non i.i.d. nature of the input signal, the output $y(t)$ should be an estimate of the innovation process, $s_r(t)$, of the music $s(t)$ (figure 9-b and d). Then, by filtering $y(t)$

with an AR filter (of course, it requires prior information), one can obtain a good estimate of the original music signal, $s(t)$, from this innovation process, as we can see in figure 9-c. The main problem of the method is to get a good prior information on the AR model of the source. This point, including the robustness of the estimation of the AR filter obtained with similar signals instead the true one $s(t)$, must be further investigated.

7 Conclusions

In this paper, two fully blind parametric procedures for the inversion of a nonlinear channel (Wiener system) were proposed. Contrary to other blind identification procedures, the system input is assumed to be non Gaussian. In fact, the method fails for Gaussian signals, because it requires higher (than 2) order statistics. The inversion procedure, in both cases, is based on the minimization of the mutual information rate of the inverse system output. For achieving optimal implementation of this criterion, whatever the signal distribution, the score function is estimated for providing the best statistics. The estimation of g is done according to a parametric model, using either a polynomial model or a neural MLP model. Both models lead to good results, even in difficult situations (hard nonlinearity and non-minimum phase filter). If the input is non i.i.d., but is a linear filtering of an i.i.d. noise (the so-called innovation), the output provides the innovation instead of the input signal $s(t)$. The restitution of the input then requires the prior knowledge (or the estimation) of the AR filter generating the signal $s(t)$.

Web server. Java demo and Matlab codes are available at the following addresses :
`http://www.uvic.es/projectes/SeparationSources`
`http://195.83.79.145/demo/ICAdemo/index.html`

Acknowledgment. This work has been in part supported by the Direcció General de Recerca de la Generalitat de Catalunya, by the European Union (BLInd Source Separation and applications: BLISS, European project IST-1999-14190) and by the French project Statistiques avancées et signal (SASI).

References

1. S. I. Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, Vol. 10, n° 2, pp. 251-276, 1998
2. R. Bars, I. Bèzi, B. Pilipar, B. Ojhelyi. Nonlinear and long range control of a distillation pilot plant. In *Identification and Syst. Parameter Estimation. 9th IFAC/IFORS Symp.*, pp. 848-853, Budapest (1990)
3. S.A. Bellings, S.Y. Fakhouri. Identification of a class of nonlinear systems using correlation analysis. *Proc. IEEE*, 66 pp. 691-697 (1978)
4. E.D. Boer. Cross-correlation function of a bandpass nonlinear network. *Proc. IEEE*, 64 pp. 1443-1444 (1976)
5. J.-F. Cardoso, B. Laheld. Equivariant Adaptive Source Separation. *IEEE Trans. on S.P.*, Vol. 44, n° 12, 3017-3030, 1996
6. T.M. Cover, J.A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications (1991)
7. W. Härdle. *Smoothing Techniques with implementation in S.* Springer-Verlag (1990)
8. I.W. Hunter. Frog muscle fibber dynamic stiffness determined using nonlinear system identification techniques. *Biophys. J.*, pp. 49-81a (1985)
9. I.W. Hunter, M.J. Korenberg. The identification of nonlinear biological systems: Wiener and Hammerstein cascade models. *Biol Cybern.*, 55, pp. 135-144 (1985)
10. G. Jacoviti, A. Neri, R. Cusani. Methods for estimating the autocorrelation function of complex stationary process. *IEEE Trans. ASSP*, 35, pp. 1126-1138 (1987)
11. C.L. Nikias, A.P. Petropulu. *Higher-Order Spectra Analysis – A Nonlinear Signal processing Framework.* Englewood Cliffs, NJ: Prentice-Hall (1993)
12. C.L. Nikias, M.R.Raghuveer. Bispectrum estimation: A digital signal processing framework. *Proc. IEEE*, 75 pp. 869-890 (1987)
13. S. Prakriya, D. Hatzinakos. Blind identification of LTI-ZMNL-LTI nonlinear channel models. *Biol. Cybern.*, 55 pp. 135-144 (1985)
14. A. Taleb, C. Jutten. Source separation in postnonlinear mixtures. *IEEE Trans. on S.P.*, Vol. 47, n°10, pp.2807-20 (1999).
15. A. Taleb, J. Solé, C. Jutten. Quasy-Nonparametric Blind Inversion of Wiener Systems. *IEEE Trans. on S.P.*, Vol. 49, n°5, pp.917-924 (2001).
16. A. Taleb, C. Jutten. Entropy Optimization. Application to Blind Source Separation. ICANN'97, Lausanne (Switzerland), October 1997. In *Lecture Notes in Computer Science 1327*, Springer, pp. 528-534.

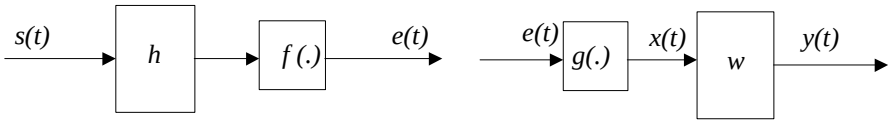


Figure 1: The unknown nonlinear convolution system (left)
and the proposed inversion structure (right).

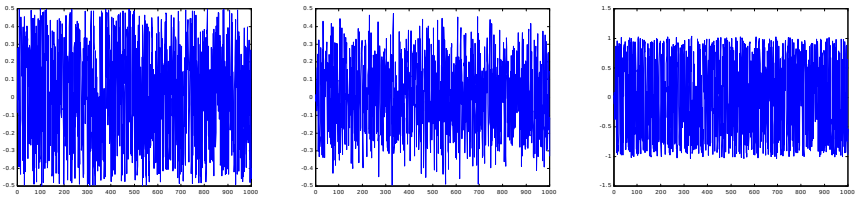


Figure 2. From left to right, input signal $s(t)$, filtered signal and distorted signal $e(t)$

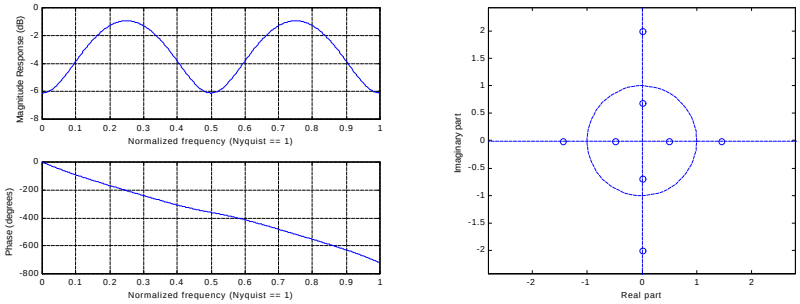


Figure 3. Frequency domain response and zeros of h

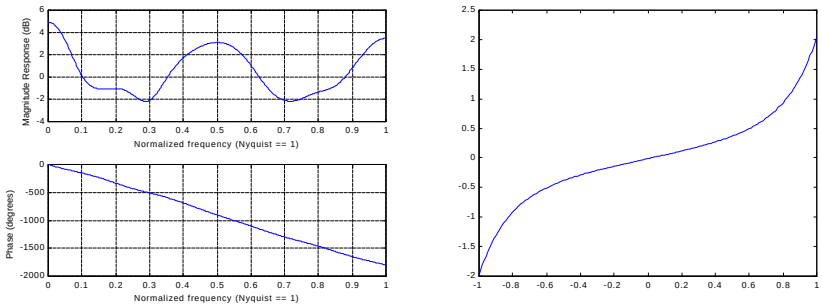


Figure 4. On the left, estimated inverse of h: w frequency domain response. On the right, estimated inverse of f using a 10-degree polynomial model: $x(t)$ vs. $e(t)$

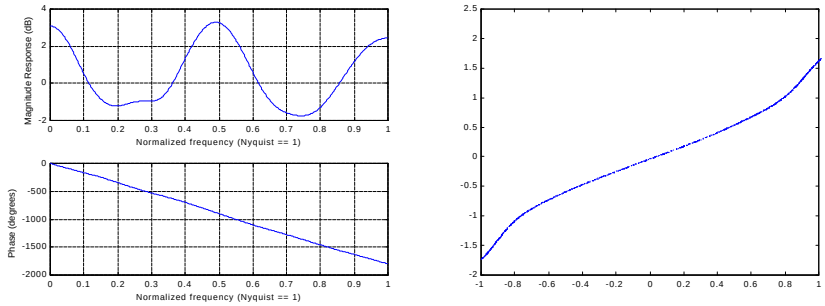


Figure 5. On the left, estimated inverse of *h*: *w* frequency domain response. On the right, estimated inverse of *f* using a multiplayer perceptron with 6 neurons in the hidden layer: $x(t)$ vs. $e(t)$

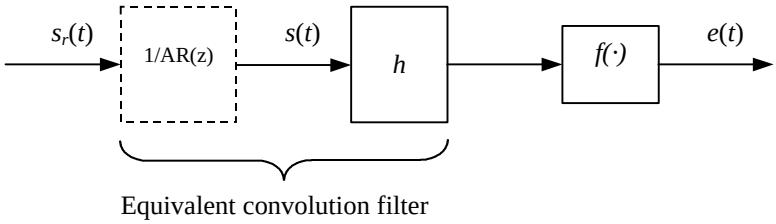


Figure 6. Block diagram of the equivalent convolution system for non i.i.d. signals.

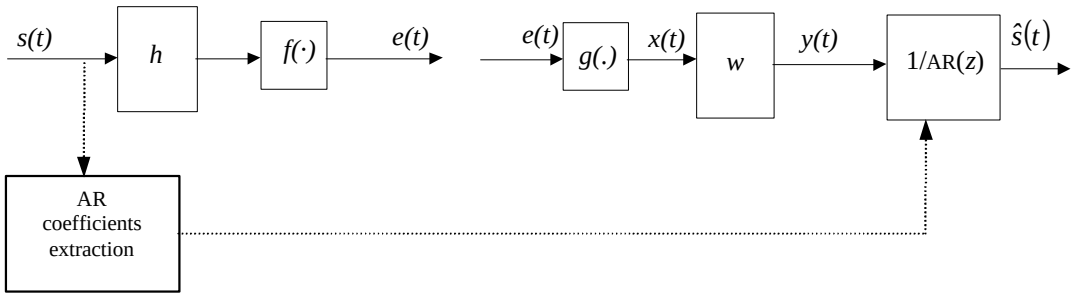
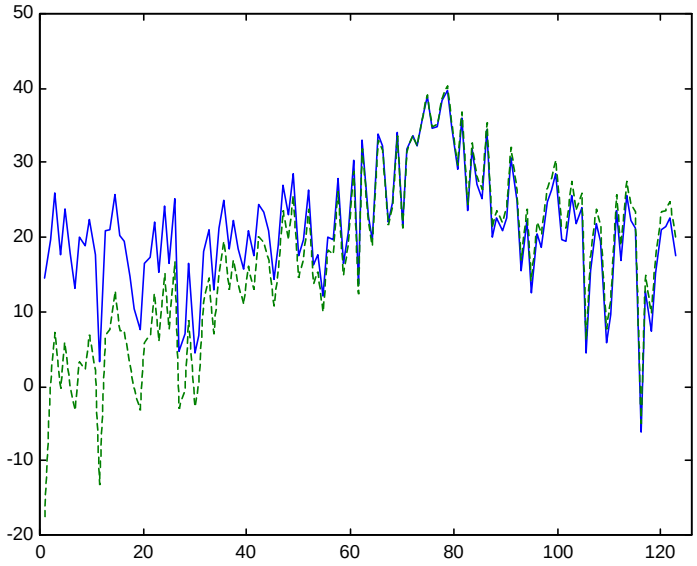
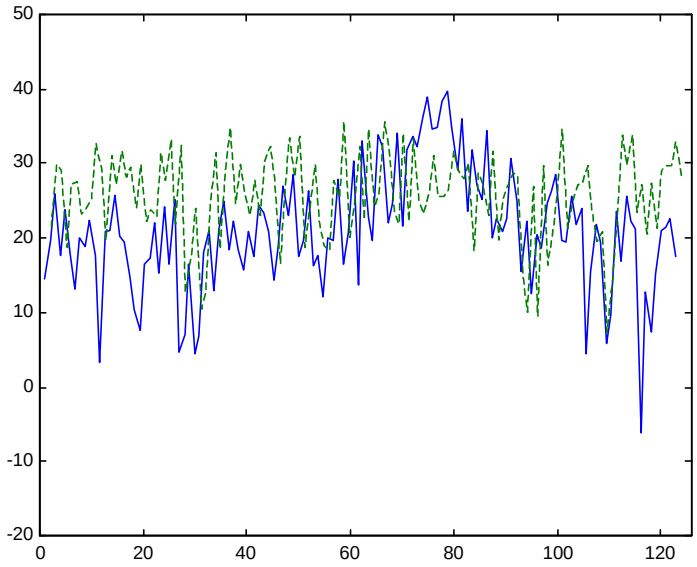


Figure 7. Block diagram of the inversion signal for non i.i.d. signals. The true original signal $s(t)$ is recovered by filtering the i.i.d. output $y(t)$ with the AR filter, which requires prior information on the signal.

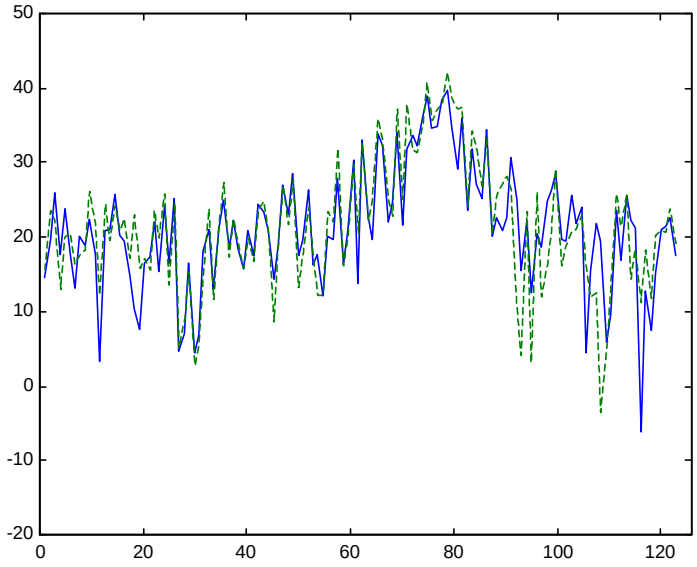
Figure 8. Inversion of a non i.i.d. signal



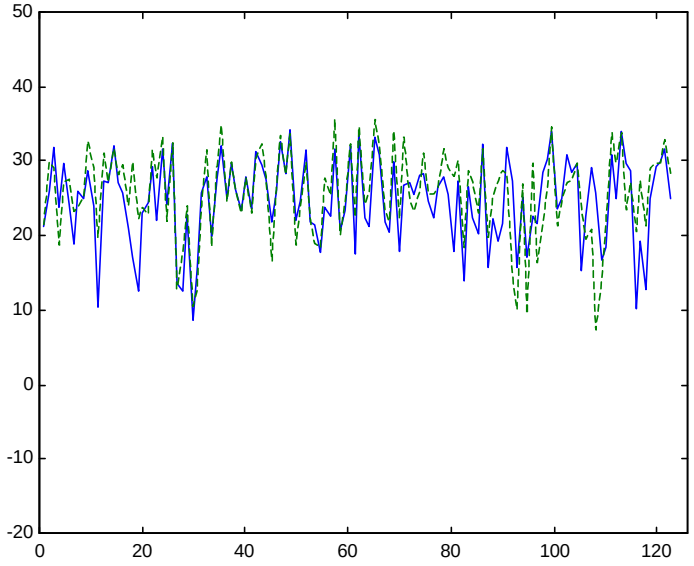
a) Spectra of the original signal $s(t)$ in solid line, and the observed signal $e(t)$ in dashed line.



b) Spectra of the original signal $s(t)$ in solid line, and the deconvolved signal $y(t)$ in dashed line. One can observe that $y(t) \neq s(t)$.

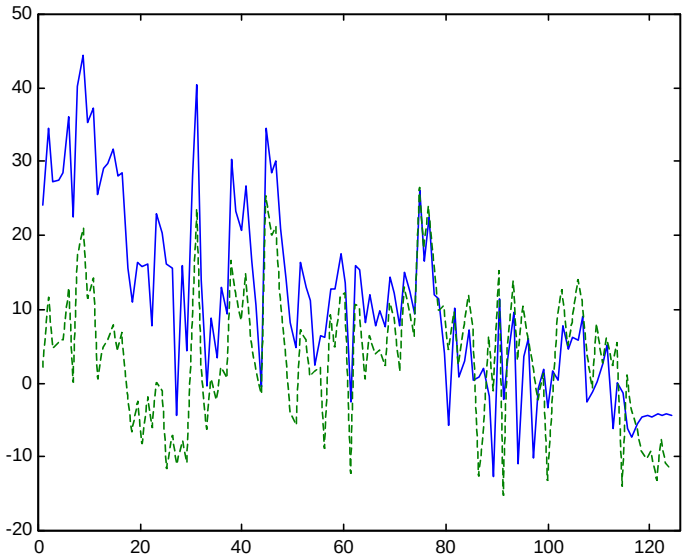


c) Spectra of the original signal $s(t)$ in solid line, and the post-processed deconvolved signal $\left[\frac{1}{AR(z)}\right] y(t) = \hat{s}(t)$ in dashed line. We can observe that the spectrum of $\hat{s}(t)$ is almost equal to the spectrum of $s(t)$

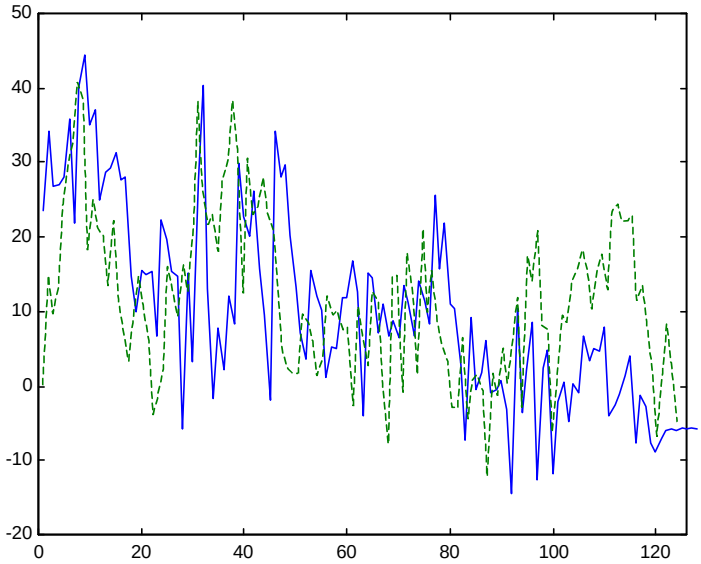


d) Spectra of the innovation process $s_r(t)$ of the original signal $s(t)$ in solid line, computed with Matlab LPC function, and the deconvolved signal $y(t)$ in dashed line. We can observe that the two spectrums are very similar.

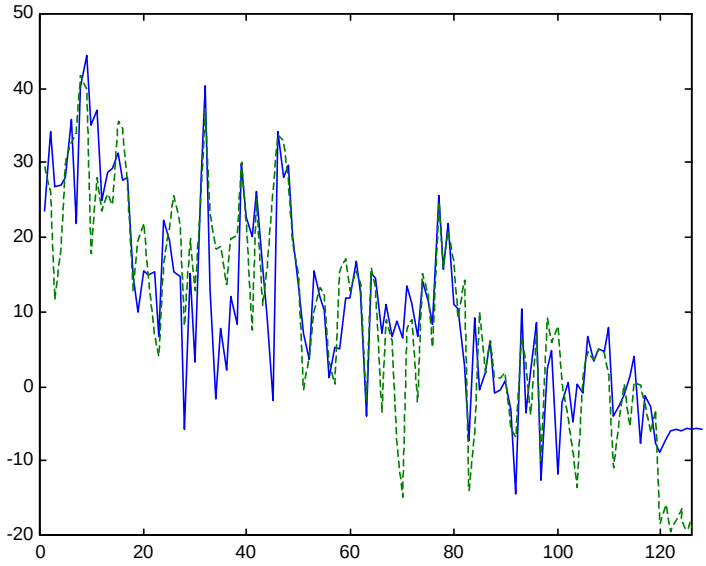
Figure 9. Inversion of a real music signal



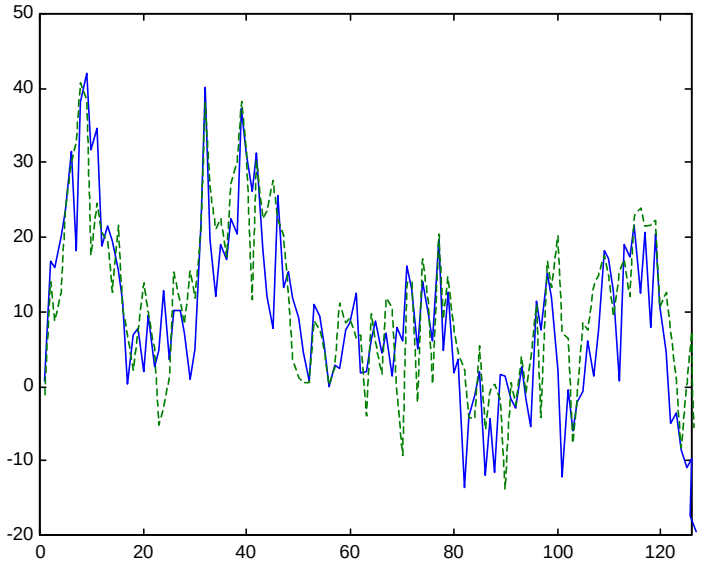
a) Spectra of the original signal $s(t)$ in solid line, and the observed signal $e(t)$ in dashed line



b) Spectra of the original signal $s(t)$ in solid line, and the deconvolved signal $y(t) \approx s_r(t)$ in dashed line



c) Spectra of the original signal $s(t)$ in solid line, and the post-processed deconvolved signal $\left[\frac{1}{AR(z)}\right] y(t) = \hat{s}(t)$ in dashed line. We can observe that the spectrum of $\hat{s}(t)$ is almost equal to the spectrum of $s(t)$



d) Spectra of the innovation process of the original signal $s(t)$ in solid line, computed with Matlab LPC function, and of the deconvolved signal $y(t)$ in dashed line. We can observe that the two spectra are very similar.