CCIT Report #444
October 2003

Blind Deconvolution with Relative Newton Method

Alexander Bronstein Michael Bronstein
Michael Zibulevsky

Department of Electrical Engineering,
Technion–Israel Institute of Technology,
Haifa 32000, Israel.

October 7, 2001

Abstract

Blind deconvolution is an important task for numerous applications in acoustics, signal processing, communications, control, etc. In this work, we study a relative optimization framework for quasi-maximum likelihood single-channel blind deconvolution and relative Newton method as its particular instance. A smooth approximation of the absolute value is considered for deconvolution of super-Gaussian sources. Special Hessian structure allows fast approximate Hessian construction and inversion with complexity comparable to that of gradient methods, and sequential optimization with gradual reduction of the smoothing parameter makes the proposed algorithm very accurate. We also propose the use of rational IIR restoration kernels, which constitute a richer family of filters than the traditionally used FIR kernels. Simulation results demonstrate the efficiency of the proposed methods.

Notation

The following notation is adopted in this work: time signals are denoted by lowercase italic and indexed starting from $n = 0$ unless stated otherwise. Z-transform domain representations of signals are denoted by uppercase italic and are exchangeable with the corresponding time-domain representations. Vectors and matrices are denoted by lowercase and uppercase italic, respectively, and indexed starting from $n = 0$ unless stated otherwise. The following notation is used:

$x,\ x_n,\ x(n)$	Time domain signals for $n = 0, 1, ...$
$X(z)$	Z-transform domain domain represenation of x_n.
$y_n = (p * x)_n$	Application of an LTI kernel p to signal x.
$y_n = P(z)\,[x_n]$	
$Y(z) = P(z)X(z)$	
I, I_N	Identity matrix of size $N \times N$.
A^T	Transpose of a matrix A.
A^*	Complex conjugate of a matrix A.
$A^H = (A^*)^T$	Hermitian transpose of a matrix A.
$\mathrm{diag}\,\{a_1, ..., a_N\}$	Diagonal matrix with elements a_i along the main diagonal.
$f'(t),\ \frac{df(t)}{dt}$	First-order derivative of $f(t)$ with respect to t.
$f''(t),\ \frac{d^2f(t)}{dt^2}$	Second-order derivative of $f(t)$ with respect to t.
$\partial_x f,\ \frac{\partial f}{\partial x}$	Partial derivative of f with respect to x.
$\partial^2_{xy} f,\ \frac{\partial^2 f}{\partial x \partial y}$	Second-order partial derivative of f with respect to x and y.
$\nabla_x f(x_0),\ \nabla f,\ g$	Gradient of f with respect to x at $x = x_0$.
$\nabla^2_{xx} f(x_0),\ \nabla^2 f,\ H_{xx},\ H$	Hessian of f with respect to x at $x = x_0$.
$\mathcal{A}\{x\},\ \mathcal{A}x$	Application of an operator $\mathcal{A}$ to signal x.
$\mathcal{J}x_n = x_{N-n}$	Mirror operator of sequence $\{x_n\}_{n=0}^{N-1}$.
$\delta(t)$	Dirac Delta function.
δ_n	Kroenecker delta sequence.
$\mathbb{E}\{x\}$	Expectation value of a random variable x.

The following abbreviations are used in this paper:

CDF	Cumulative Distribution Function
DFT	Discrete Fourier Transform
FIR	Finite Impulse Response
FFT	Fast Fourier Transform
i.i.d.	independent identically distributed
IIR	Infinite Impulse Response
ISI	Inter-Symbol Interference
ML	Maximum Likelihood
OLA	Overlap-and-Add
PDF	Probability Density Function
SIR	Signal to Interference Rate
SNR	Signal to Noise Rate

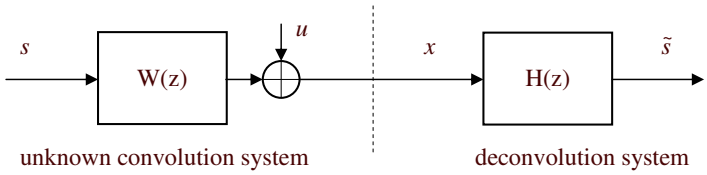

Figure 1: Schematic representation of the deconvolution problem.

1 Introduction

Blind deconvolution problem appears in various applications related to acoustics, optics, geophysics, communications, control, etc. In comunications, the term *blind channel equalization* is more common, as the main interest lies in retrieving the data s transmitted over a dispersive communication channel [1, 2, 3, 4, 5]. In control, blind deconvolution is usually known as *blind identification*, since the main interest lies in obtaining a model of the system [6, 7, 8], whereas in acoustics, optics and geophysics the term *blind deconvolution* is more adeguate, since the goal is to "undo" the influence of a system by finding its stable inverse.

The general setup of the single-channel blind deconvolution problem is presented in Figure 1. The observed sensor signal x is created from the *source signal* s passing through a causal convolutive system described by the impulse response w,

$$x_n \;=\; \sum_{k=0}^{\infty} w_k \, s_{n-k} + u_n, \tag{1}$$

where u is the additive sensor noise. The setup is termed *blind* if only x is accessible, whereas no knowledge on w, s and u is available. The problem of blind deconvolution aims to find such a deconvolution (or restoration) kernel h, that produces a possibly delayed waveform-preserving estimate of s:

$$\tilde{s}_n \;=\; \sum_{k=0}^{\infty} h_k \, x_{n-k} \approx c \cdot s_{n-\Delta}, \tag{2}$$

where c is a scaling factor and Δ is an integer shift. Equivalently, the *global system response* should be approximately a Kroenecker delta, up to scale factor:

$$g_n \;=\; (w * h)_n \approx c \cdot \delta_{n-\Delta}. \tag{3}$$

A commonly used assumption is that s is non-Gaussian.

1.1 Prior work

The majority of blind deconvolution methods described in literature focus on estimating the impulse response of the convolution system $W(z)$ from the observed signal x using a

causal finite length (FIR) model and then determining the source signals from this estimate [9, 10, 7, 11, 12]. Many of these methods use batch mode calculations and usually suffer from high computational complexity.

In their fundamental work, Amari *et al.* [13] introduced a time-domain blind deconvolution algorithm based on the natural gradient learning algorithm, which was originally proposed in context of blind source separation [14] and became very attractive due to the so-called *equivariant property* [15, 16]. Thanks to this property, convergence of natural gradient algorithm depends only on the current global system response. The natural gradient algorithm estimates directly the inverse kernel $H(z) = W^{-1}(z)$ and allows real-time processing. In [17], a generalization of the algorithm for multichannel case was presented. Efficient frequency-domain implementations were derived in [18, 19, 20, 21].

One of the most serious disadvantages of gradient methods is their relatively slow convergence [22, 23]. In this work, we present a blind deconvolution algorithm based on the relative Newton method, originally proposed in the context of sparse blind source separation in [24, 25]. We utilize special Hessian structure to derive a fast version of the algorithm with complexity comparable to that of gradient methods. We will first consider a batch mode version, which will be then extended for online processing. The online version of the algorithm is capable to handle time-varying convolutive systems.

We extend the classical works that limit their attention to causal FIR deconvolution kernels by considering the use of rational causal deconvolution kernels of the form

$$H(z) \;=\; \frac{B(z)}{A(z)} = \frac{b_0 + b_1\, z^{-1} + ... + b_{N-1}\, z^{-(N-1)}}{a_0 + a_1\, z^{-1} + ... + a_{M-1}\, z^{-(M-1)}}, \tag{4}$$

where, $N-1$ and $M-1$ are the respective orders of the numerator and denominator. The use of the all-pole part $A(z)$ together with the FIR part $B(z)$ yields a richer family of filters and generally allows to express the restoration kernel with less coefficients.

2 Quasi-maximum-likelihood blind deconvolution

Under the assumption that the restoration kernel $H(z)$ is stable [26] and the source signal is real and i.i.d., the normalized minus-log-likelihood function of the observed signal x is [13, 17, 27]

$$f(a, b; x) \;=\; -\frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|H(e^{i\theta})\right| d\theta + \frac{1}{T}\sum_{n=0}^{T-1} \phi\left(y_n\right), \tag{5}$$

where $y_n = H(z)\,[x_n]$ is a source estimate $\phi(\cdot) = -\log p\,(\cdot)$, where $p(\cdot)$ is the probability density function of the source s_n. In this paper, we consider a stable restoration kernel

$H(e^{i\theta}) = \frac{B(e^{i\theta})}{A(e^{i\theta})}$, where

$$\begin{aligned} A(e^{i\theta}) &= \sum_{n=0}^{M-1} a_n \, e^{-in\theta} \\ B(e^{i\theta}) &= \sum_{n=0}^{N-1} b_n \, e^{-in\theta} \end{aligned} \tag{6}$$

are the discrete-time Fourier transforms of the denominator and numerator polynomials, respectively. Cost function (5) can be also obtained using negative joint entropy [13, 17] and information maximization [28] considerations.

2.1 Choice of $\phi(\cdot)$

Consistent estimator can be obtained by minimizing $f(a, b; x)$ even when $\phi(\cdot)$ is not exactly equal to $-\log p(\cdot)$. Such *quasi-ML estimation* has been shown to be practical in instantaneous blind source separation when the source PDF is unknown or not well-suited for optimization. For example, when the source is super-Gaussian (e.g. it is sparse or sparsely representable), a smooth approximation of the absolute value function is a good choice for $\phi(\cdot)$ [29, 30, 31]. In this work, we focus our attention on super-Gaussian sources and use smooth approximation of the absolute value. To approximate the absolute value, we use a family of convex smooth functions

$$\phi_\lambda(t) = |t| - \lambda \log\left(1 + \frac{|t|}{\lambda}\right) \tag{7}$$

with λ a positive smoothing parameter [25]; $\phi_\lambda(t) \to |t|$ as $\lambda \to 0^+$ (see Figure 2). The derivatives of $\phi_\lambda(t)$ are

$$\phi'_\lambda(t) = \mathrm{sign}t \cdot \left(1 - \frac{1}{1 + \frac{|t|}{\lambda}}\right) \tag{8}$$

$$\phi''_\lambda(t) = \left(1 + \frac{|t|}{\lambda}\right)^{-2}, \tag{9}$$

and it can be shown that $\phi'_\lambda(t) \to \mathrm{sign}t$ and $\phi''_\lambda(t) \to \frac{1}{2\lambda}\delta(t)$ as $\lambda \to 0^+$. For convenience, we will henceforth omit λ from our notation and will refer to $\phi_\lambda(\cdot)$ as to $\phi(\cdot)$.

In case of sub-Gaussian sources $\phi_k(t) = |t|^k$ for $k > 1$ is a good choice for $\phi(\cdot)$. For example, when sources are uniformly distributed, increase of k refines the approximation and in the limit $k \to \infty$, $\phi_k(t)$ approaches minus log PDF (see Figure 3).

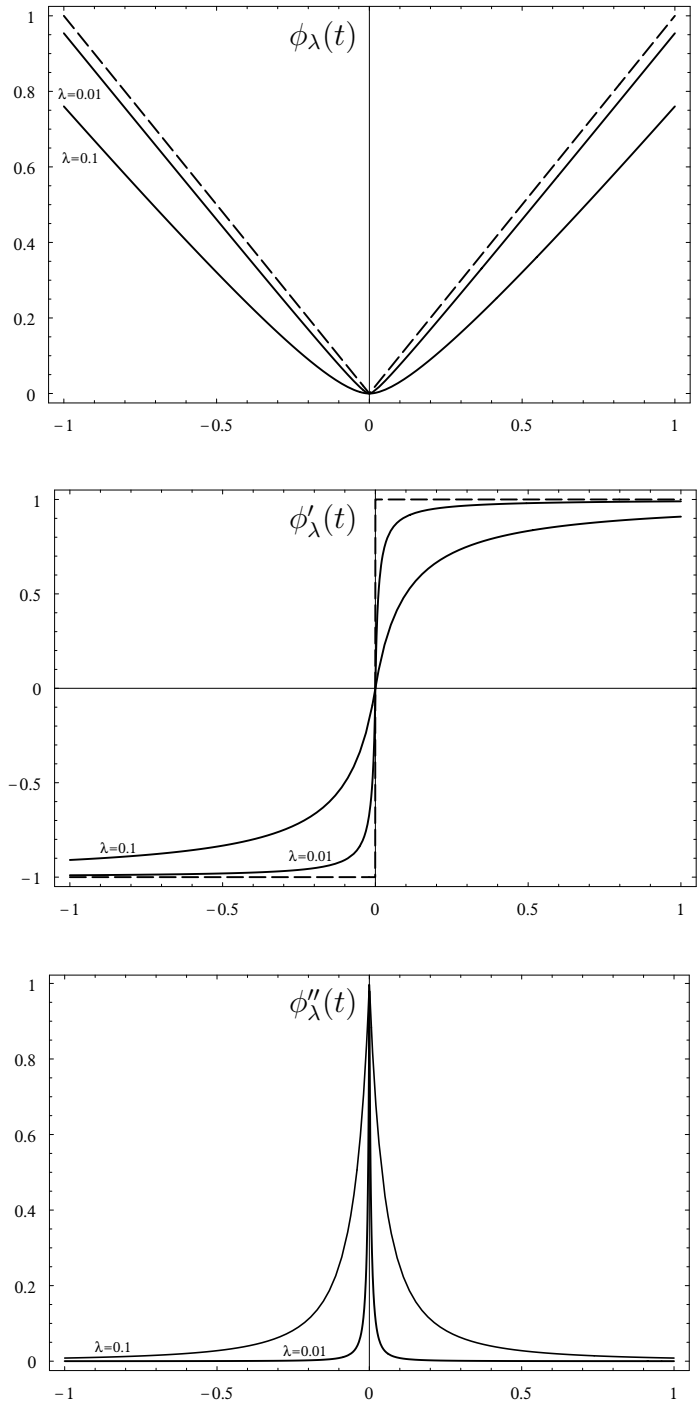

Figure 2: The smooth approximation of the absolute value and its first- and second-order derivatives for different values of λ. Dashed lines show the limit $\lambda \to 0^+$.

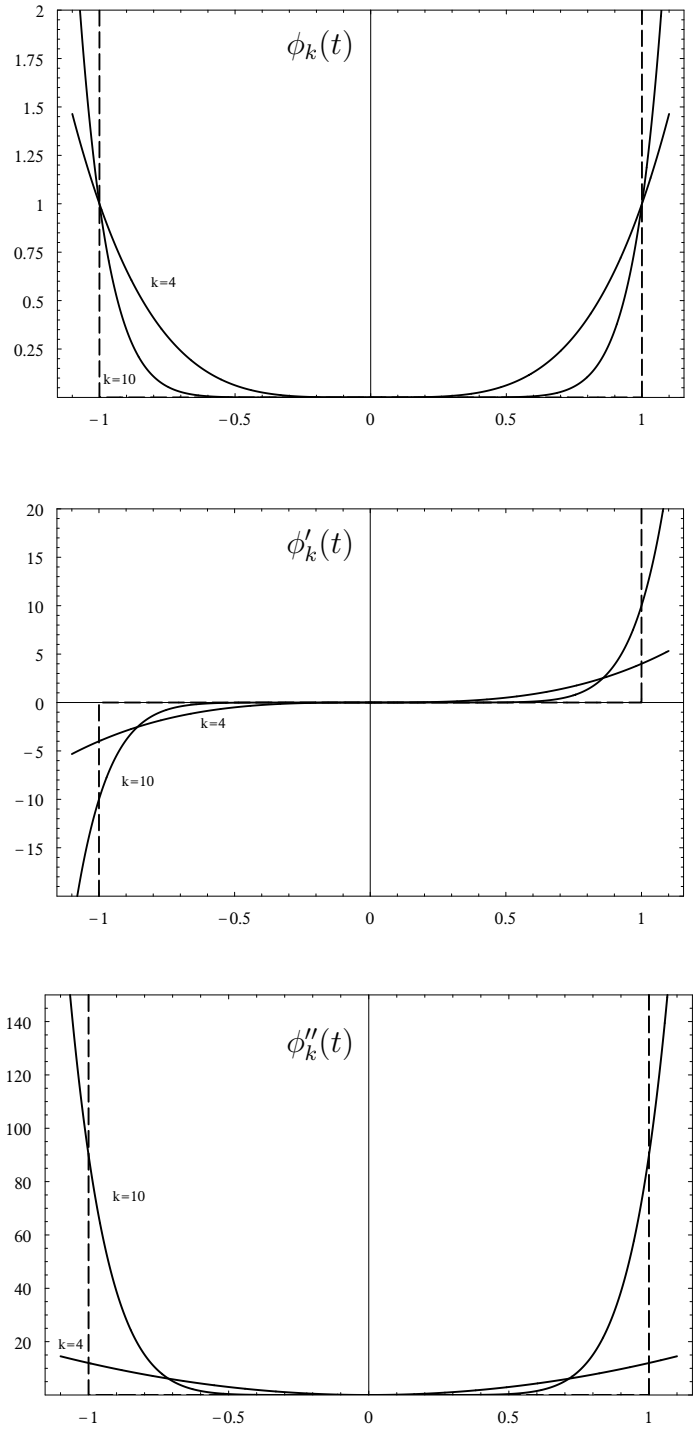

Figure 3: The non-linearity suitable for sub-Gaussian distributions and its first- and second-order derivatives for different values of k. Dashed lines show the limit $k \to \infty$.

2.2 Approximation of the log-likelihood function using the FFT

In practice, the first term of $f(a, b; x)$ containing the integral is difficult to evaluate; however, it can be approximated to any desired accuracy by

$$\frac{1}{2\pi}\int_{-\pi}^{\pi}\log\left|\frac{B(e^{i\theta})}{A(e^{i\theta})}\right|d\theta \;\approx\; \frac{1}{N_F}\sum_{k=0}^{N_F-1}\log\left|\frac{B_k}{A_k}\right| = \frac{1}{2N_F}\sum_{k=0}^{N_F-1}\log|B_k|^2 - \log|A_k|^2, \quad (10)$$

where

$$\begin{aligned} A_k &= A\left(e^{i\frac{2\pi k}{N_F}}\right) \\ B_k &= B\left(e^{i\frac{2\pi k}{N_F}}\right) \end{aligned} \quad (11)$$

are the DFT-N_F coefficients of the sequences $a = \{a_n\}_{n=0}^{M-1}$ and $b = \{b_n\}_{n=0}^{N-1}$, respectively, zero-padded to N_F. The approximation error vanishes as N_F grows to infinity. It is convenient to choose N_F to be an integer power of 2, since in this case A_k and B_k can be computed efficiently using the FFT. For convenience, we will henceforth refer to the approximate target function as to $f(a, b; x)$. Furthermore, we will define

$$f_1 = \sum_{k=0}^{N_F-1}\log|B_k|^2 - \log|A_k|^2 \quad (12)$$

$$f_2 = \sum_{n=0}^{T-1}\phi(y_n). \quad (13)$$

Using this notation, the objective function becomes

$$f = -\frac{1}{2N_F}f_1 + \frac{1}{T}f_2. \quad (14)$$

Optimization algorithm described in Section 3 requires knowledge of the gradient and the Hessian of f, which are given in the following propositions (for proofs see Appendix A.1 –A.3):

Proposition 1 *The gradient and the Hessian of $f_1(a, b; x)$ with respect to a, b are given by*

$$\nabla f_1 = \begin{bmatrix} -\Phi_A^H A'^* - \left(\Phi_A^H A'^*\right)^* \\ \Phi_B^H B'^* + \left(\Phi_B^H B'^*\right)^* \end{bmatrix}$$

and

$$\nabla^2 f_1 = \begin{bmatrix} H_{aa} & \\ & H_{bb} \end{bmatrix},$$

where

$$\begin{aligned} H_{aa} &= -\left(\Phi_A^H \operatorname{diag}\left\{A''\right\}^H\right)^* \Phi_A - \Phi_A^H \left(\operatorname{diag}\left\{A''\right\} \Phi_A\right)^* \\ H_{bb} &= \left(\Phi_B^H \operatorname{diag}\left\{B''\right\}^H\right)^* \Phi_B + \Phi_B^H \left(\operatorname{diag}\left\{B''\right\} \Phi_B\right)^*, \end{aligned}$$

Φ_A^H and Φ_B^H are DFT matrices of sizes $N_F \times M$ and $N_F \times N$, respectively; $A' = \left[\begin{array}{cccc} A_0^{-1} & \ldots & A_{N_F-1}^{-1} \end{array}\right]^T$, $B' = \left[\begin{array}{cccc} B_0^{-1} & \ldots & B_{N_F-1}^{-1} \end{array}\right]^T$, $A'' = -\left[\begin{array}{cccc} A_0^{-2} & \ldots & A_{N_F-1}^{-2} \end{array}\right]^T$ *and* $B'' = -\left[\begin{array}{cccc} B_0^{-2} & \ldots & B_{N_F-1}^{-2} \end{array}\right]^T$.

Proposition 2 *Let y be the source estimate given by $y_n = B(z)A^{-1}(z)\left[x_n\right]$. Its first- and second-order derivatives with respect to a_n and b_n are given by*

$$\begin{aligned} \partial_{a_i} y_n &= \partial_{a_0} y_{n-i} = -A^{-1}(z)\left[y_{n-i}\right] \\ \partial_{b_i} y_n &= \partial_{b_0} y_{n-i} = A^{-1}(z)\left[x_{n-i}\right] \\ \partial^2_{a_i a_j} y_n &= \partial^2_{a_0} y_{n-i-j} = 2A^{-2}(z)\left[y_{n-i-j}\right] \\ \partial^2_{a_i b_j} y_n &= \partial^2_{a_0 b_0} y_{n-i-j} = -A^{-2}(z)\left[x_{n-i-j}\right] \\ \partial^2_{b_i b_j} y_n &= 0. \end{aligned}$$

Proposition 3 *The gradient and the Hessian of $f_2(a, b; x)$ with respect to a, b are given by*

$$\nabla f_2 = \left[\begin{array}{c} \mathcal{J}\left(\partial_{a_0} y_n * \mathcal{J}\phi'\right)_0 \\ \vdots \\ \mathcal{J}\left(\partial_{a_0} y_n * \mathcal{J}\phi'\right)_{M-1} \\ \mathcal{J}\left(\partial_{b_0} y_n * \mathcal{J}\phi'\right)_0 \\ \vdots \\ \mathcal{J}\left(\partial_{b_0} y_n * \mathcal{J}\phi'\right)_{N-1} \end{array}\right]$$

and

$$\nabla^2 f_2 = \left[\begin{array}{cc} H_{aa}^1 + H_{aa}^2 & H_{ab}^1 + H_{ab}^2 \\ \left(H_{ab}^1 + H_{ab}^2\right)^T & H_{bb} \end{array}\right],$$

where

$$\begin{aligned} \left(H_{aa}^1\right)_{ij} &= \mathcal{J}\left(\partial_{a_0} y_n * \mathcal{J}\alpha^j\right)_i \\ \left(H_{aa}^2\right)_{ij} &= \mathcal{J}\left(\partial^2_{a_0} y_n * \mathcal{J}\phi'\right)_{i+j} \end{aligned}$$

for $i, j = 0, ..., M-1$*;*

$$\begin{aligned}\left(H^1_{ab}\right)_{ij} &= \mathcal{J}\left(\partial_{b_0} y_n * \mathcal{J}\alpha^j\right)_i \\ \left(H^2_{ab}\right)_{ij} &= \mathcal{J}\left(\partial^2_{b_0} y_n * \mathcal{J}\phi'\right)_{i+j}\end{aligned}$$

for $i = 0, ..., M-1; j = 0, ..., N-1$*;*

$$\left(H_{bb}\right)_{ij} = \mathcal{J}\left(\partial_{b_0} y_n * \mathcal{J}\beta^j\right)_i$$

for $i, j = 0, ..., N-1$*;* $\phi' = \left[\begin{array}{cccc}\phi'(y_0) & \phi'(y_1) & ... & \phi'(y_{T-1})\end{array}\right]^T$, $\alpha^j_n = \phi''(y_n)\,\partial_{a_0} y_{n-j}$, $\beta^j_n = \phi''(y_n)\,\partial_{b_0} y_{n-j}$ *and* $\mathcal{J}$ *denotes the mirror operator.*

2.3 Computational complexity

Application of the deconvolution filter can be split into application of an FIR filter $B(z)$ of order $N-1$ followed by application of an all-pole IIR filter $A(z)$ of order $M-1$. It is reasonable to assume that the length of the input signal is significantly larger than the order of the deconvolution kernel, which particularly implies $N \ll T$. In such a case, convolution with $B(z)$ can be efficiently implemented using the *overlap-and-add* (OLA) method [26], whose complexity is approximately $4T \log_2 N$, neglecting edge effects. Application of the all-pole filter $A(z)$ to the input signal requires about MT operations.

Computation of the gradient and Hessian of f_2 involves convolution of two sequences of length T and cropping the resulting sequence to L (which is either M or N). For $L < 6 \log T$, it is preferable the direct implementation of the convolution, which takes about LT operations. For larger L's, it is more efficient to perform the convolution using the FFT, which takes about $6T \log T$ operations. We will denote by

$$C(L, T) = \min\left\{L, 6 \log_2 T\right\} \cdot T \tag{15}$$

the complexity of the cropped convolution.

Evaluation of f_1 demands computation of N_F DFT coefficients A_k, B_k and application of $\log|\cdot|^2$, whose complexity will be denoted by k_L, to each of them. This results in $2N_F \log_2 N_F + 2k_L N_F$ operations. Evaluation of f_2 requires computation of y_n, which involves application of $A(z)$ and $B(z)$, followed by application of $\phi(\cdot)$ to T elements of y_n. This results in $4T \log_2 N + MT + kT$ operations, where k stands for the complexity of $\phi(\cdot)$.

Evaluation of ∇f_1 involves computation of the coefficients A'_k, B'_k and application of 4 FFTs of length N_F, which results in $2N_F + 4N_F \log_2 N_F$ operations, where complexity of complex division is considered equal to that of complex multiplication. Evaluation of ∇f_2 requires

1. Computation of the sequence ϕ'_n by applying $\phi'(\cdot)$ to the sequence y_n, which was computed previously.

2. Computation of the partial derivatives $\partial_{a_0} y_n$ and $\partial_{b_0} y_n$, which require application of $A^{-1}(z)$ to y_n and x_n, respectively.

3. Two cropped convolutions of the two partial derivative sequences with the sequence ϕ'_n.

Step 1 requires $k'T$ operations, where k' stands for the complexity of ϕ'; Step 2 takes $2MT$ operations; and Step 3 takes $C(M,T) + C(N,T)$ operations, resulting in total in $2MT + C(M,T) + C(N,T) + k'T$ operations.

Evaluation of $\nabla^2 f_1$ requires computation of the coefficients A''_k, B''_k and application of $8N_F$ FFTs of length N_F. This results in $2N_F + 8N_F^2 \log_2 N_F$ operations. Evaluation of $\nabla^2 f_2$ requires

1. Computation of the sequence ϕ''_n,

2. Computation of the sequences α_n^j, β_n^j.

3. $M + N$ cropped convolutions of the partial derivative sequences $\partial_{a_0} y_n$ and $\partial_{b_0} y_n$ (previously computed) with ϕ''_n.

4. Computation of the second-order partial derivative sequences $\partial^2_{a_0} y_n$ and $\partial^2_{a_0 b_0} y_n$, which require application of $A^{-1}(z)$ to $\partial_{a_0} y_n$ and $\partial_{b_0} y_n$, respectively.

5. Two cropped convolutions for computation of ξ_n and η_n.

Step 1 requires $k''T$ operations, where k'' stands for the complexity of ϕ''; Step 2 takes $(M + N)T$ operations; Step 3 takes $M \cdot C(M,T) + N \cdot C(N,T)$ operations; Step 4 takes $2MT$ operations; and Step 5 takes $C(2M-1,T) + C(M+N-1,T)$ operations, resulting in total in $(k''+3M+N)T + M \cdot C(M,T) + N \cdot C(N,T) + C(2M-1,T) + C(M+N-1,T)$ operations.

The overall complexity is summarized below:

$$
\begin{aligned}
f \quad &: \quad 2N_F \log_2 N_F + 2k_L N_F + 4T \log_2 N + MT + kT \\
\nabla f \quad &: \quad 2N_F + 4N_F \log_2 N_F + 2MT + C(M,T) + C(N,T) + k'T \\
\nabla^2 f \quad &: \quad 2N_F + 8N_F^2 \log_2 N_F + (k'' + 3M + N)T + \\
&\quad + M \cdot C(M,T) + N \cdot C(N,T) + C(2M-1,T) + C(M+N-1,T)
\end{aligned}
$$

In our implementation, the constants k_L, k, k' and k'' were experimentally evaluated as $k_L = 3.30 \pm 0.25\%$, $k = 8.80 \pm 1.10\%$, $k' = 1.57 \pm 1.65\%$ and $k'' = 2.10 \pm 0.77\%$.

Generally, $N \sim M$ and $N_F \sim 10N$. When $T \gg M, N$, one has the following complexity:

$$
\begin{aligned}
f \quad &: \quad (4 \log_2 N + M + k)\, T \\
\nabla f \quad &: \quad (3M + N + k')\, T \\
\nabla^2 f \quad &: \quad (M^2 + N^2 + 6M + 2N + k'' - 2)\, T
\end{aligned}
$$

In this case, Hessian construction is more computationally difficult than Hessian inversion, which is $\mathcal{O}\left(M^3 + N^3\right)$. It can also be seen that the deconvolution filter numerator $B(z)$ of length N contributes less to the overall computational complexity than the denominator $A(z)$, whose length is M.

When $T \sim M \sim N$ and $N_F = 10N$, one has the following complexity:

$$\begin{aligned} f \quad &: \quad N^2 + 24N \log_2 N + (20k_L + k + 20 \log_2 10)N \\ \nabla f \quad &: \quad 2N^2 + (20 + k' + 40 \log_2 10)N + 42N \log_2 N \\ \nabla^2 f \quad &: \quad 804N^2 \log_2 N + 4N^2 + 800 \log_2 10 \cdot N^2 + (20 + k'')N \end{aligned}$$

2.4 Normalization of the restoration kernel

Let a^*, b^* be a minimizer of $f(a, b; x)$. Then, for every constant $k \neq 0$, the coefficients $k \cdot a^*, k \cdot b^*$ are also minimizers of $f(a, b; x)$. For numerical stability, it is desirable to fix the arbitrary scaling of the nominator and the denominator polynomials to some constant value. This can be done by forcing, for example $a_0 = 1$. The gradient and the Hessian of the objective function have to be modified by removing the first element from the gradient, and the first row and column from the Hessian. Restoration kernel normalization does not affect the computational complexity except a negligible factor.

We will henceforth assume that the restoration kernel is normalized only whenever the normalization implies significant changes, which are not straightforward. In rest of cases, no normalization will be assumed for simplicty.

3 Relative Newton algorithm

In [25], a fast relative optimization algorithm for blind source separation based on the Newton method was introduced. Here we introduce a relative optimization framework for blind deconvolution. We will first describe the general relative optimization algorithm and the block relative optimization method for online processing. Next, the *relative Newton* algorithm, using a Newton step in the relative optimization framework will be introduced. A fast version of the relative Newton step will be studied in Section 5.

3.1 Relative optimization algorithm

The main idea of relative optimization is to iteratively produce source signal estimate and use it as the observed signal at the next iteration. For convenience, let us represent the restoration filter in the Z-transform domain, denoting by $H(z)$, $A(z)$ and $B(z)$ the Z-transforms of h, a and b, respectively. The relative optimization algorithm has the following form:

Relative optimization algorithm

1. Start with an initial estimate $B^{(0)}(z), A^{(0)}(z)$ of the restoration filter numerator and denominator, respectively; and the observed signal $x_n^{(0)} = x_n$.
2. **For** $k = 1, 2, ...,$ until convergence
3. Compute current source signal estimate: $x_n^{(k)} = \frac{B^{(k-1)}(z)}{A^{(k-1)}(z)} \left[x_n^{(k-1)} \right]$.
4. Starting with $A(z) = B(z) = 1$ (identity filter), compute the vectors of numerator and denominator coefficients $B^{(k)}(z), A^{(k)}(z)$ producing one or few steps of a conventional optimization method, which sufficiently decrease the objective function $f(A(z), B(z); x_n^{(k)})$.
5. Update the estimated restoration filter: $H^{(k)}(z) = \frac{B^{(k)}(z)}{A^{(k)}(z)} H^{(k-1)}(z)$
6. **End For**

This method allows to construct large restoration kernels of the form

$$H(z) \quad = \quad \frac{B^{(0)}(z)B^{(1)}(z) \cdot ... \cdot B^{(K)}(z)}{A^{(0)}(z)A^{(1)}(z) \cdot ... \cdot A^{(K)}(z)} \tag{16}$$

with high-order numerator and denominator using a set of relatively low-order factors $\left\{ \frac{B^{(k)}(z)}{A^{(k)}(z)} \right\}_{k=0}^{K}$. Another remarkable property of the relative optimization algorithm is its equivariance, stated in the following proposition:

Proposition 4 *The relative optimization algorithm is equivariant, i.e. its convergence at iteration k depends only on $G^{(k-1)}(z) = W(z)H^{(k-1)}(z)$.*

Proof of this proposition is straightforward, since Step 4 and the update in Step 5 do not depend explicitly on $W(z)$, but on the currents global system response [17].

3.2 Block relative optimization

Main disadvantage of the relative optimization approach is that it treats the observed signal x as a whole, not allowing thus on-line processing. In some cases the input signal might be very long, which makes the algorithm impractical. A possibility to overcome these difficulties is to partition the input into blocks and estimate the restoration kernel for the current block using the data of the previous block and the previous restoration kernel estimate. Let us assume for simplicity that the input signal is partitioned into equally sized blocks, denoted by $x^{[k]} = \{x_n\}_{n=kL}^{(k+1)L-1}$ for $k = 0, 1, 2, ...,$ where L is the block length. The block relative optimization algorithm has the following structure:

Block relative optimization algorithm

1. Initialize $H^{(0)}(z) = 1$, $x^{[0]} = \{x_n\}_{n=0}^{L-1}$.
2. **For** $k = 0, 1, 2, ...$
3. Starting with $A(z) = B(z) = 1$, compute the vectors of numerator and denominator coefficients producing one or few steps of a conventional optimization method, which sufficiently decrease the objective function $f(A(z), B(z); x^{[k]})$.
4. Update the estimated restoration filter: $H^{(k+1)}(z) = \frac{B(z)}{A(z)} H^{(k)}(z)$.
5. Update the next block: $x^{[k+1]} = H^{(k+1)}(z)\,[x_n]$ for $n = kL, ..., (k+1)L - 1$.
6. **End For**

The signal $\tilde{s} = \left[x^{[0]}, x^{[1]}, ..., x^{[n]}, ...\right]$ produced by the algorithm with delay of L samples with respect to the input is the adaptive estimate of the source signal s. Block relative optimization algorithm can also treat cases when the input signal x is produced as a result of s passing through a *time-varying* convolution system. In this case, the estimated restoration kernel will not converge to a constant filter, but will also vary with time.

The blocks should be long enough to provide sufficient statistics for faithful source signal estimation, yet they should be as short as possible to avoid long delays, undesired for online processing. Block length also dictates how fast the restoration kernel can vary with time; shorter blocks allow faster variability. A reasonable order for the block length is about 10 times the effective length of the restoration kernel impulse response; particularly, $L = 10N$ if the restoration kernel is a FIR filter.

3.3 Limited memory version

Both the batch mode and the block version of the relative optimization algorithm assume infinite memory and produce a restoration kernel of order growing at each iteration. In real applications it might be necessary to limit the numerator and denominator order to some finite number $K \leq L$. This can be done by replacing the update

$$\begin{aligned} A^{(k+1)}(z) &= A(z)A^{(k)}(z) \\ B^{(k+1)}(z) &= B(z)B^{(k)}(z) \end{aligned} \tag{17}$$

with a cropped version

$$\begin{aligned} a_n^{(k+1)} &= \left(a^{(k)} * a\right)_n \\ b_n^{(k+1)} &= \left(b^{(k)} * b\right)_n \end{aligned} \tag{18}$$

for $n = 0, ..., K - 1$.

3.4 Newton method

Newton method is an efficient tool for unconstrained optimization, which often provides very fast (quadratic) rate of convergence. We will first consider the standard Newton method; later we will see how its use in the relative optimization framework allows to overcome the difficulty emerging from the computationally expensive Newton iterations.

In the standard Newton approach, the direction d at each iteration is given by solution of the linear system

$$Hd = -g, \tag{19}$$

where $H = \nabla^2 f(v; x)$ is the Hessian of f, $g = \nabla f(v; x)$ is the gradient and $v = [a, b]^T$ is the vector of optimization variables. Since the objective function is non-convex, in order to guarantee descent direction, positive definiteness of the Hessian is forced by using modified Cholesky factorization, which finds such a diagonal matrix R, that the matrix $H + R$ is positive definite, and provides a solution to the modified system

$$(H + R)d \;\; = \;\; -g. \tag{20}$$

Having the direction d, the new iterate v^+ is given by

$$v^+ \;\; = \;\; v + \alpha v, \tag{21}$$

where α is the step size determined by either exact line search

$$\alpha \;\; = \;\; \mathrm{argmin} f(v + \alpha d; x), \tag{22}$$

or by backtracking line search:

Backtracking line search

$\alpha := 1$
While $f(v + \alpha d; x) > f(v; x) + \beta\alpha\nabla f(v; x)^T d$
    $\alpha := \gamma\alpha$
End While

where β and γ are constants. The use of line search guarantees monotonic decrease of the objective function at every iteration. In our implementation, we used the backtracking line search with $\beta = \gamma = 0.3$. It should be noted that when the gradient norm becomes very small (say, below 10^{-5}), computational inaccuracies make the line search inefficient. For this reason, we used the Newton direction as is (i.e. chose $\alpha = 1$) when the gradient norm fell below 10^{-5}.

When the Newton method is used to minimize f, where the IIR part of the restoration kernel is non-trivial (i.e. $M > 1$), special precautions must be taken to guarantee stability of $A(z)$, otherwise f_2 is liable to take huge, numerically untractable values. In Section 4 it will be described how to modify the backtracking line search in order not to "fall out" of the stability region.

3.4.1 Frozen Hessian

For medium-scale problems, the *frozen Hessian* method was found efficient [32, 33]. It consists of "freezing" the Hessian in the Newton method for some K iterations, i.e. once computed, the Hessian is used in the next $K-1$ iterations. The Hessian is therefore constructed and Cholesky factorization is performed every K*-th* iteration and the obtained factors are used to solve the Newton system in the next $K-1$ iterations, using the gradient computed at each iteration.

3.4.2 Block-coordinate update

Yet another method to avoid large computational complexity associated with the Newton system solution in medium-scale problems is to use *block-coordinate update*. The method consists of iteratively updating each time a different block of the optimization variables vector [34, 35]. The update is performed using the Newton step on the variables of the current block and assuming the rest of the variables fixed. The size of the Newton system is therefore equal to the block size. This allows to overcome the difficulties associated with solution of large Newton system.

Below is presented a particular case when equally-sized consequtive blocks are used. K denotes the block size and for simplicity we assume that the optimization variables vector v is composed of L such blocks. We also denote by w_l the l*-th* block of v, $w_l = \{v_i\}_{i=lK}^{(l+1)K-1}$.

Block-coordinate Newton algorithm

1. Initialize the algorithm with some initial guess $v^{(0)}$.
2. **For** $k = 1, 2, ...,$ until convergence
3. **For** $l = 0, 2, ..., L-1$
4. Update the l*-th* block of $v^{(k-1)}$, $w_l^{(k-1)}$ by performing a Newton step on a K-dimensional vector of optimization variables w_l, starting with $w_l^{(k-1)}$ and fixing the rest of the variables.
5. **End For**
6. **End For**

In [35], block-coordinate Newton step was used in the relative optimization framework for quasi-maximum likelihood blind source separation, and it was demonstrated that the block-coordinate method usually yields faster convergence. The issue will be discussed in Section 5.4.

3.5 Relative Newton method

Newton method can be used as is to find the restoration kernel that minimizes f. Another possibility is to use Newton method in Step 4 of the relative optimization algorithm (or on Step 3 of the block relative optimization algorithm). The latter possibility is advantageous, since it allows to construct a high-order restoration kernel using relatively low-order factors. This, in turn, implies solution of smaller optimization problems. Further advantage was found in using a single Newton step for unconditional optimization in the relative optimization algorithm [25] or in the block relative optimization algorithm. These algorithms will be termed henceforth as *relative Newton* and *block relative Newton* method, respectively.

However, for large T, Hessian construction becomes computationally difficult and for large M, N complexity of Hessian inversion required to compute the Newton direction may make the use of Newton method infeasible. In Section 5, we will show a fast version of the relative Newton step, which exploits the Hessian structure in the neighbourhood of the solution and the initialization $A(z) = B(z) = 1$. This significantly reduces computational complexity of both Hessian construction and inversion, and allows to perform the Newton step with complexity of the order of gradient descent methods.

4 Forcing stability of the restoration kernel

In Section 3 we have seen the relative optimization framework, in which filter coefficients a and b optimal in the sense of $f(a, b; x)$ were found over $\mathbb{R}^N \times \mathbb{R}^M$. However, in reality we are interested only in *stable* restoration filters, i.e. such filters, whose impulse response satisfies

$$\lim_{n\to\infty} h_n \;=\; 0. \tag{23}$$

The sufficient and necessary condition for restoration filter stability is that all the roots of the polynomial $A(z)$ are inside the unit circle [26]. We will henceforth term such a polynomial as *stable*.

In the relative optimization framework, the restoration filter found at the end of the algorithm is given by (16), where $A^{(k)}(z)$, $B^{(k)}(z)$ are polynomials of order $M-1$ and $N-1$, respectively. Since the kernels $\frac{B^{(k)}(z)}{A^{(k)}(z)}$ found at first iterations are nearly random, a necessary condition that the zeros and the poles of $\frac{B^{(k)}(z)}{A^{(k)}(z)}$ can be cancelled by the poles and the zeros, respectively, of $\frac{B^{(k+1)}(z)B^{(k+2)}(z)\cdot\ldots}{A^{(k+1)}(z)A^{(k+2)}(z)\cdot\ldots}$. This implies that the kernel $\frac{B^{(k)}(z)}{A^{(k)}(z)}$ has to be invertible for every k, i.e. both $A^{(k)}(z)$ and $B^{(k)}(z)$ has no roots outside the unit circle (Figure 4). When $M = N$, the subspace of filters of the form $\frac{B(z)}{A(z)}$ with stable non-zero $A(z)$ and $B(z)$ forms a *group* with the multiplication operator. We will first introduce a barrier function, which defines the stability region of an arbitrary polynomial $P(z)$. Later, we will show how to force stability of the restoration kernel using the stability barrier.

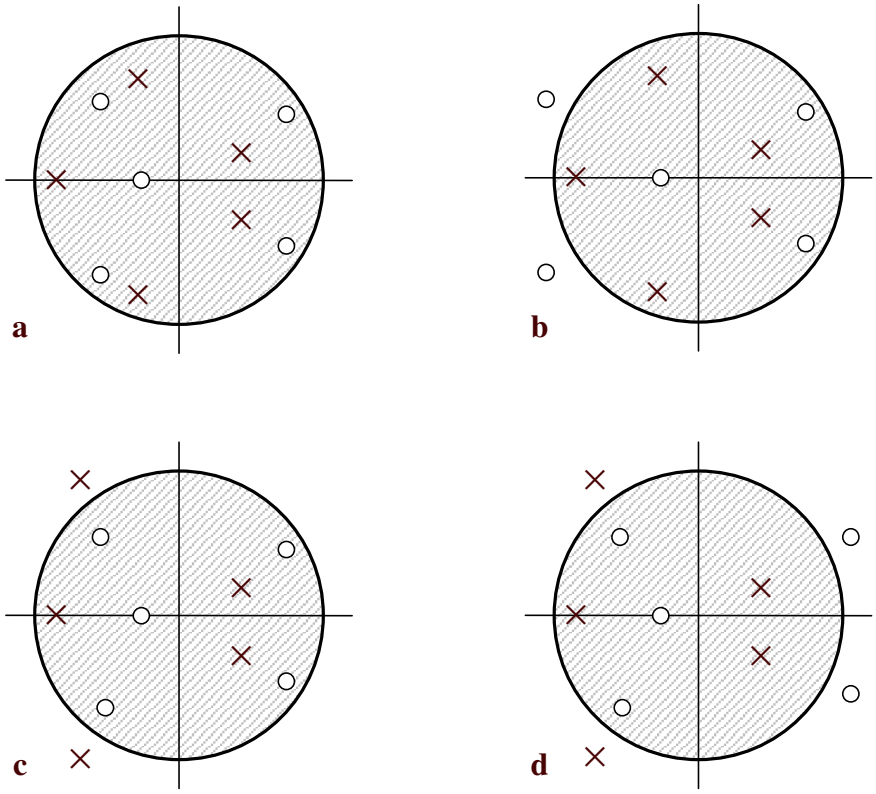

Figure 4: Z-plane representation of the restoration kernel zeros (circles) and poles (crosses): a. both $A(z)$ and $B(z)$ are stable; b. $A(z)$ is stable, $B(z)$ is unstable; c. $A(z)$ is unstable, $B(z)$ is stable; d. both $A(z)$ and $B(z)$ are unstable;

4.1 Stability barrier

Given a polynomial $P(z) = p_0 + p_1 z^{-1} + ... + p_K z^{-K}$ of order K, let us compute the impulse response q_n of the all-pole IIR kernel $P^{-1}(z)$ on some finite, yet sufficiently long time interval $t = 0, ..., N_S - 1$

$$q_n \;=\; \mathcal{Z}^{-1}\left\{P^{-1}(z)\right\}. \tag{24}$$

Obviously, when $P(z)$ has roots outside the unit circle, $|q_n|$ grows with n. Let us define a convex barrier function

$$f_S(P(z); N_S) \;=\; \frac{1}{N_S}\sum_{n=0}^{N_S-1}|q_n|. \tag{25}$$

When N_S is sufficiently large, f_S grows very fast if $P(z)$ has roots outside the unit circle, being typically

$$f_S \;\sim\; \frac{|r|^{N_S}-1}{N_S(|r|-1)}, \tag{26}$$

where $|r|$ is the largest root of $P(z)$. When $N_S \to \infty$, f_S approaches the ideal barrier function

$$f_S(P(z); \infty) \;=\; \begin{cases} 0 & : \;\; P(z) \text{ is stable} \\ \infty & : \;\; P(z) \text{ is unstable.} \end{cases} \tag{27}$$

Figure 5 depicts the behavior of f_S for a simple first-order kernel.

In practice, f_S given by (25) is difficult for optimization, since it contains absolute values. However, we can use the smooth approximation $\phi_{\lambda_S}(\cdot)$ to substitute the absolute value, namely,

$$f_S(P(z)) \;\approx\; \frac{1}{N_S}\sum_{n=0}^{N_S-1}\phi_{\lambda_S}(q_n). \tag{28}$$

Since the approximate absolute value is used to reflect merely the order of magnitude of $|q_n|$, the smoothing parameter λ_S in $\phi_{\lambda_S}(\cdot)$ does not need to be very small (in our implementation $\lambda_S = 1$ was used), which makes the optimization less difficult. To simplify the notation, we will denote by f_S the non-normalized version of f_S, i.e.

$$f_S(P(z)) \;=\; \sum_{n=0}^{N_S-1}\phi_{\lambda_S}(q_n), \tag{29}$$

and refer to the stability barrier function as to $\frac{1}{N_S}f_S$. We will also omit λ_S wherever possible.

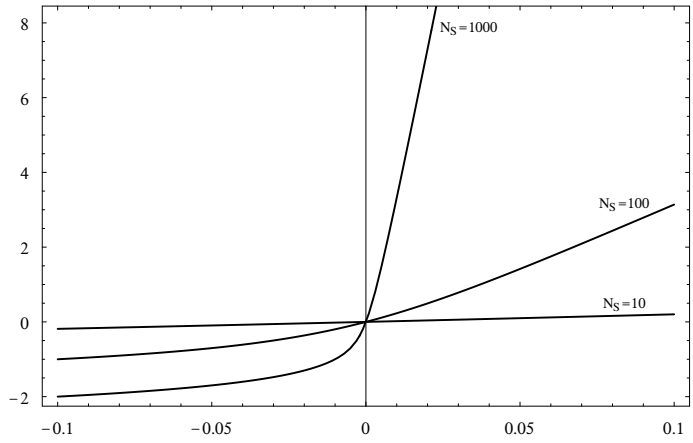

Figure 5: $\log_{10} f_S$ of a first-order kernel $P(z) = 1 + (1+\epsilon)z^{-1}$ plotted as a function of ϵ for different values of N_S.

It must be noted that unless $x \equiv 0, b \equiv 0$, the second term f_2 in the objective function acts as a stability barrier for the denominator of the restoration kernel. If $A(z)$ is unstable, y_n grows exponentially with n and the sum of absolute values $\sum_{n=0}^{T-1} |y_n|$ becomes very large. Therefore, the barrier has to be used to impose stability of $B(z)$ only, yielding the following modified objective function

$$f(a, b; x) \quad = \quad -\frac{1}{2N_F} f_1(a, b; x) + \frac{1}{T} f_2(a, b; x) + \frac{\nu_S}{N_S} f_S(b), \tag{30}$$

where $\nu_S \leq 1$ is a parameter that can be used in practice to reduce the influence of f_S without the need to increase N_S too much.

Figure 6 (left) depicts the value of f for $W(z) = 1 + 0.5z^{-1}$ applied to a 1000-sample long source signal and all-pole restoration kernel of the form $H(z) = \frac{1}{1+a_1 z^{-1}}$. It can be seen that when a_1 approaches stability margins ($a_1 = \pm 1$), the objective function grows very fast, becoming about 10^{39} for $a_1 = 1.1$. The minimum of f is obtained for $a_1 \approx 0.5$ as expected. Figure 6 (middle) depicts the value of f without stability barrier for $W(z) = \frac{1}{1+0.5z^{-1}}$ and FIR restoration kernel of the form $H(z) = 1 + b_1 z^{-1}$. In contrast to the previous case, here the function takes finite values on and beyond stability region of $H(z)$. When stability barrier is added to the objective function (Figure 6, right), f grows very fast as b_1 approaches ± 1, yet having negligible influence on the values of f inside the stability region and location of the minimum.

The gradient and the Hessian of $f_S(b)$ with respect to a are equal to zero, whereas the gradient and the Hessian with respect to b are given in the following proposition (for proof see Appendix A.4):

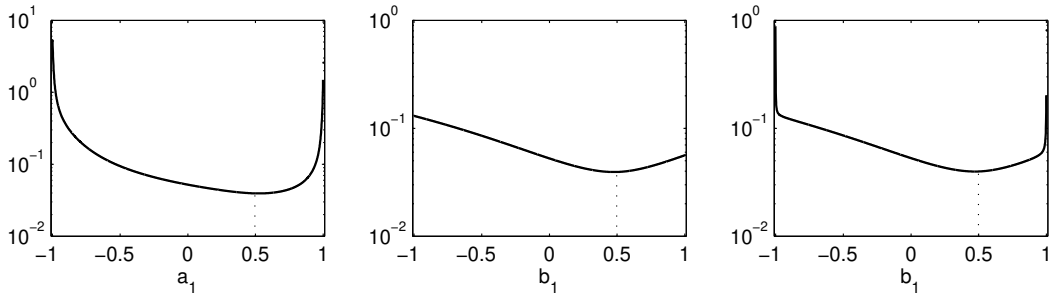

Figure 6: Left: $f(a = [1, a_1], b = 1; x)$ as function of a_1 for source filtered with the FIR kernel $W(z) = 1 + 0.5z^{-1}$. Middle: $f(a = 1, b = [1, b_1]; x)$ without stability barrier for $W(z) = (1 + 0.5z^{-1})^{-1}$. Right: the same for f with stability barrier.

Proposition 5 *The gradient and the Hessian of $f_S(b)$ with respect to b are given by*

$$\nabla f_S = \begin{bmatrix} \mathcal{J}\left(h_{\partial Q} * \mathcal{J}\phi'_S\right)_0 \\ \vdots \\ \mathcal{J}\left(h_{\partial Q} * \mathcal{J}\phi'_S\right)_{N-1} \end{bmatrix}$$

and

$$\left(\nabla^2 f_S\right)_{ij} = \mathcal{J}\left(h_{\partial Q} * \mathcal{J}\gamma^j\right)_i + \mathcal{J}\left(h_{\partial^2 Q} * \mathcal{J}\phi'_S\right)_{i+j}$$

for $i, j = 0, ..., N-1$; $\phi'_S = \left[\begin{array}{cccc} \phi'_{\lambda_S}(q_0) & \phi'_{\lambda_S}(q_1) & \ldots & \phi'_{\lambda_S}(q_{N_S-1}) \end{array}\right]^T$, $\gamma^j_n = \phi''_{\lambda_S}(q_n)\, h_{\partial Q}(n-j)$ and $\mathcal{J}$ denotes the mirror operator.

4.2 Stability-conditioned line search

When stability of $A(z)$ and $B(z)$ is imposed, the line search used at each Newton step has to be modified in such a way that no search will be performed outside the stability region of $A(z)$ and $B(z)$:

Stability-conditioned backtracking line search

$\alpha := 1$
While $f(v + \alpha d; x) > f(v; x) + \beta\alpha\nabla f(v; x)^T d$ **or** $v + \alpha d$ is unstable
    $\alpha := \gamma\alpha$
End While

The easiest way to determine polynomial stability is to evaluate f_S at each iteration. Hence, the condition "$v+\alpha d$ is unstable" reads $f_S(a) > R_\infty$ **or** $f_S(b) > R_\infty$ **or** $f(v+\alpha d; x) > R_\infty$,

where

$$\begin{aligned} a &= \{v + \alpha d\}_{k=0}^{k=M-1} \\ b &= \{v + \alpha d\}_{k=M}^{k=M+N-1} \end{aligned} \tag{31}$$

are current restoration filter coefficients and R_∞ is a very large real number, which is infinity for any practical use. The condition $f(v+\alpha d; x) > R_\infty$ is added to avoid numerical problems caused by very large values of the objective function, which occur outside the stability region.

4.3 Computational complexity

Evaluation of $f_S(B(z); N_S)$ involves at first application of an all-pole IIR filter of order N to a vector of length N_S with one at $n = 0$ and zero otherwise. This produces the sequence q_n and requires about $N_S N$ operations. Next, $\phi(\cdot)$ has to be applied to q_n, requiring in total kN_S operations. Therefore, evaluation of f_S requires $(N_S + k)N$ operations.

Evaluation of ∇f_S requires:

1. Application of $\phi'(\cdot)$ to the previously computed sequence q_n, which produces the sequence ϕ'_n.
2. Application of $B^{-1}(z)$ to the sequence δ_n, to produce the kernel $h_{\partial Q}$.
3. Cropped convolution of the kernel $h_{\partial Q}$ with the sequence ϕ'_n.

Step 1 requires $k'N_S$ operations; Step 2 takes about $N_S N$ operations; and Step 3 takes $C(N, N_S)$ operations, yielding in total about $(k'+N)N_S+C(N, N_S)$ operations for gradient evaluation.

Evaluation of $\nabla^2 f_S$ requires:

1. Computation of the sequence ϕ''_n,
2. Computation of the sequences γ_n^j.
3. N cropped convolutions of the kernel $h_{\partial Q}$ (previously computed) with ϕ''_n.
4. Computation of the kernel $h_{\partial^2 Q}$, which require application of $B^{-1}(z)$ to $h_{\partial Q}$.
5. Cropped convolution for computation of ζ_n.

Step 1 requires $k''N_S$ operations; Step 2 takes $N_S N$ operations; Step 3 takes $N \cdot C(N, N_S)$ operations; Step 4 takes $N_S N$ operations; and Step 5 takes $C(2N-1, N_S)$ operations, resulting in total in $(k''+N)N_S + N \cdot C(N, N_S) + C(2N-1, N_S)$ operations for Hessian evaluation.

Normally, N_S is desired to be at least one order of magnitude larger than N to faithfully reflect the impulse response of $B^{-1}(z)$. Therefore, we assume that $N_S \gg N$ and obtain the following computational complexities:

$$
\begin{aligned}
f_S &: N_S N \\
\nabla f_S &: 2N_S N + k' N_S \\
\nabla^2 f_S &: N_S N^2 + 3N_S N + (k'' - 1)N_S
\end{aligned}
$$

It can be concluded that the computational complexity of f_S, its gradient and Hessian is linear with N_S; f_S and ∇f_S are also linear with N, whereas $\nabla^2 f_S$ is quadratic with N.

4.4 Sequential increase of N_S

We have previously seen that the barrier f_S approaches the ideal stability barrier as N_S grows. Therefore, to minimize the influence of the barrier inside the stability region, it is desirable to use N_S as large as possible. On the other hand, computational complexity considerations suggest using the smallest N_S possible.

It is possible to start with a relatively small N_S, converging to some solution using, for example, the relative optimization algorithm to perform the unconstrained optimization. Then, N_S should be increased and the unconstrained optimization repeated, starting from the found solution. Usually, the latter requires less iterations to converge, which results in lower computational complexity compared to using large N_S from the beginning. In practice, to avoid computational complexity associated with large values of N_S, its highest value is limited by some $N_{\max}$ and the influence of the barrier function is reduced by decreasing ν_S. Modified relative optimization algorithm with sequential increase of N_S has the following form:

Relative optimization algoritm with sequentially increasing N_S

1. Start with $N_S^{(1)}$, $\nu_S = 1$, some initial guess of the restoration filter $H^{(0)}(z) = \frac{B^{(0)}(z)}{A^{(0)}(z)}$ and $x_n^{(0)} = x_n$.
2. **For** $k = 1, 2, ..., K$
3. Compute current source signal estimate: $x_n^{(k)} = \frac{B^{(k-1)}(z)}{A^{(k-1)}(z)} \left[x_n^{(k-1)} \right]$.
4. Starting with $A(z) = B(z) = 1$, find $A^{(k)}(z), B^{(k)}(z) = \text{argmin} f(A(z), B(z); x_n^{(k)}, N_S^{(k)}, \nu_S^{(k)})$.
5. Update the estimated restoration filter: $H^{(k)}(z) = \frac{B^{(k)}(z)}{A^{(k)}(z)} H^{(k-1)}(z)$.
6. Update $N_S^{(k+1)} = \mu_S N_S^{(k)}$.
7. Optional: update $\nu_S^{(k+1)} = \mu_\nu \nu_S^{(k)}$.
8. **End For**

The parameter $\mu_S > 1$ determines the growth rate of the stability barrier parameter N_S and it is generally set to $\mu_S \sim 10$. The parameter μ_ν determines the decrease rate of ν_S and is usually set to $\mu_\nu \sim 0.1$. The block relative optimization algorithm described in Section 3.2 can be modified in a similar way to allow sequential increase of N_S.

5 Fast relative Newton step

Practical use of the relative Newton step described in Section 3 is limited to small values of M, N and T, due to the complexity of Hessian construction ($\mathcal{O}\left((N^2 + M^2)T\right)$ when $T \gg M, N$ and $\mathcal{O}\left(N^2 \log N\right)$ when $T \sim M \sim N$), and solution of the Newton system ($\mathcal{O}\left(M^3 + N^3\right)$). This complexity can be significantly reduced if special Hessian structure is exploited. When the relative optimization framework is used, at each iteration the Hessian is evaluated at $a_n = b_n = \delta_n$ and in a sufficiently small neighbourhood of the minimum, $x_n^{(k)}$ becomes approximately s_n (supposing the deconvolution kernel is of a sufficient order to reproduce with sufficient accuracy the original signal from the observed one).

Proposition 6 *The Hessian of $f(a, b; x)$ with respect to a, b at $a = b = \delta_n$ and $x \approx s$ has an approximate tri-diagonal structure, with non-zero elements at the main diagonal and $\pm M$-th secondary diagonals.*

The proof is given in Appendix A.5 . Approximate tri-diagonal Hessian structure is depicted in Figure 7 (left). When the restoration kernel is normalized by forcing $a_0 = 1$, the first row and column are removed from the Hessian, still preserving the approximately tri-diagonal structure (Figure 7, middle).

5.1 Fast approximate solution of the Newton system

Exploiting the special Hessian structure, it is now possible to derive an efficient scheme for Newton system solution. Results presented in this section assume normalized restoration kernel (see Section 2.4); slightly different results are obtained in a similar way for a non-normalized kernel.

Let us denote by g and H the gradient and the Hessian of f, respectively, whose respective sizes are $(M + N - 1) \times 1$ and $(M + N - 1) \times (M + N - 1)$. Using the tri-diagonal Hessian approximation, the Newton direction d arising from solution of the Newton system (19) can be found approximately by solving $\min\{M, N\} - 1$ systems of linear equations of size 2×2 with respect to d_k and d_{k+M}

$$\begin{aligned} H_{k,\,k}\, d_k + H_{k,\,k+M}\, d_{k+M} &= -g_k \\ H_{k+M,\,k}\, d_k + H_{k+M,\,k+M}\, d_{k+M} &= -g_{k+M} \end{aligned} \tag{32}$$

for $k = 1, ..., \min\{M, N\} - 1$, and in case $M \neq N$, an additional set of $|M - N|$ single equations

$$H_{k,\,k}\, d_k \;\; = \;\; -g_k \tag{33}$$

for

$$k = \left\{ \begin{array}{lcl} M,\; 2M, ..., M + N - 1 & : & M < N \\ N, ..., M & : & M > N, \end{array} \right.$$

from where g_k can be found directly.

In order to guarantee decent direction and avoid saddle points, we force positive definiteness of the Hessian by inverting the sign of negative eigenvalues in system (32). In order to do so, we find analytically the eigenvalues λ_k^1,λ_k^2 and the diagonalizing matrices V_k of each of the 2×2 symmetric matrices

$$D_k \;\; = \;\; \left[\begin{array}{cc} H_{k,\,k} & H_{k,\,k+M} \\ H_{k+M,\,k} & H_{k+M,\,k+M} \end{array} \right], \tag{34}$$

namely,

$$\lambda_k^{1,\,2} \;\; = \;\; \frac{1}{2}\left(H_{k,\,k} \pm \sqrt{4H_{k,\,k+M}^2 + (H_{k,\,k} - H_{k+M,\,k+M})^2} + H_{k+M,\,k+M} \right) \tag{35}$$

and

$$V_k \;\; = \;\; \left[\begin{array}{cc} H_{k,\,k} - \sqrt{4H_{k,\,k+M}^2 + (H_{k,\,k} - H_{k+M,\,k+M})^2} - H_{k+M,\,k+M} & 2H_{k,\,k+M} \\ H_{k,\,k} + \sqrt{4H_{k,\,k+M}^2 + (H_{k,\,k} - H_{k+M,\,k+M})^2} - H_{k+M,\,k+M} & 2H_{k,\,k+M} \end{array} \right]^T \tag{36}$$

for $H_{k,\,k+M} \neq 0$ and $V_k = I$ otherwise. We invert the sign of negative $\lambda_k^{1,\,2}$ and force small eigenvalues to be above some positive threshold, say, $\epsilon = 10^{-8} \cdot \max\{|\lambda_k^1|, |\lambda_k^2|\}$:

$$\tilde{\lambda}_k^i \;\; = \;\; \max\left\{|\lambda_k^i|, \epsilon\right\} \;\;\;\; : \;\;\;\; i = 1, 2. \tag{37}$$

Then, a modified system

$$\tilde{D}_k \left[\begin{array}{c} d_k \\ d_{k+M} \end{array} \right] \;\; = \;\; \left[\begin{array}{c} g_k \\ g_{k+M} \end{array} \right], \tag{38}$$

where

$$\tilde{D}_k = V_k \left[\begin{array}{cc} \tilde{\lambda}_k^1 & \\ & \tilde{\lambda}_k^2 \end{array} \right] V_k^{-1}, \tag{39}$$

is solved analytically with respect to d_k and d_{k+M} for each k.

5.2 Delayed delta initialization

Sometimes, initialization $b_n = \delta_{n-R}$, where $0 < R < N$, is preferable to $b_n = \delta_n$ [13]. In this case, the term H_{ab} of the Hessian assumes the form depicted in Figure 7 (right), i.e. its non-zero main diagonal is shifted by R and a shifted anti-diagonal appears. In addition, the Hessian of f_1 with respect to b assumes a form of an anti-diagonal matrix with shifted anti-diagonal. However, the contribution of this term is generally negligible compared to the term coming from f_2. Neglecting the anti-diagonal term $\nabla_a^2 f_1$, fast approximate solution of the Newton system involves regularized solution of a set of 3×3 systems of the form:

$$\begin{bmatrix} d_k \\ d_{k+M-R} \\ d_{k+M+R} \end{bmatrix} = \begin{bmatrix} H_{k,\,k} & H_{k,\,k+M-R} & H_{k,\,k+M+R} \\ H_{k+M-R,\,k} & H_{k+M-R,\,k+M-R} & 0 \\ H_{k+M+R,\,k} & 0 & H_{k+M+R,\,k+M+R} \end{bmatrix} \begin{bmatrix} -g_k \\ -g_{k+M-R} \\ -g_{k+M+R} \end{bmatrix} \tag{40}$$

for $k = 1, ..., R$. This can be carried out similarly to solution of 2×2 systems described in Section 5.1, however, derivation of analytical expressions becomes more complicated and is note presented in this work. Another possibility is to consider techniques for solution of sparse symmetric systems. For example, one can use sparse modified Cholesky factorization for direct solution, or alternatively, conjugate gradient-type methods, possibly preconditioned by incomplete Cholessky factor, for iterative solution. In both cases, Cholesky factor is often not as sparse as the original matrix, but it becomes sparser, when appropriate matrix permutation is applied before factorization [25].

Yet another possibility is to limit attention to the particular case when $M = 1$ (FIR restoration kernel), where the Hessian in a neighbourhood of the solution is approximately diagonal if the anti-diagonal term $\nabla^2 f_1$ is neglected, or has an additional shifted anti-diagonal otherwise. In both cases approximate Newton direction can be found by either solving a set of independent linear equations, or a set of 2×2 systems.

5.3 Computational complexity

The approximate Hessian of f_1 and the Hessian of f_S at $a_n = b_n = \delta_n$ is very simple and does not depend on the data. Therefore, computational effort is required for their construction is $\mathcal{O}(1)$. The approximate Hessian of f_2 of a normalized restoration kernel is a tri-diagonal matrix with a total of $M + N + 2\min\{M, N\} - 3$ non-zero elements, of which, due to symmetry, only $M + N + \min\{M, N\} - 2$ different elements have to be computed. This requires evaluation of the main diagonals of H_{aa}, H_{ab} and H_{bb}, which consists of:

1. Computing the sequence ϕ_n''.

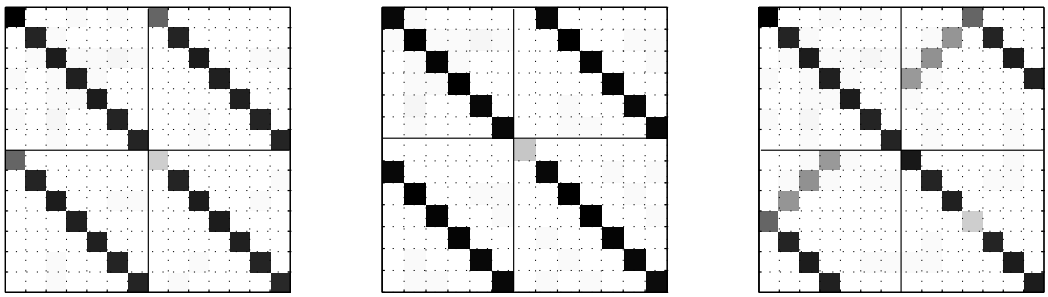

Figure 7: Structure of the Hessian of the non-normalized kernel (left) and the normalized kernel (middle) at the solution point for $a_n = b_n = \delta_n$. Right: non-normalized kernel with delayed initialization $b_n = \delta_{n-3}$. The parameters are $M = N = 7$, $T = 10^3$, $N_S = 256$ and $\lambda = 1$. White represents near-zero elements, whereas black stands for strong non-zero elements of the Hessian, either positive or negative.

2. Computing the sequences $\partial^2_{a_0} y_n$ and $\partial^2_{a_0 b_0} y_n$ by applying $A^{-1}(z)$ to $\partial_{a_0} y_n$, $\partial_{b_0} y_n$ already computed on the gradient computation stage.

3. Squaring the sequences $\partial_{a_0} y_n$ and $\partial_{b_0} y_n$, and computing the product $\partial_{a_0} y_n \, \partial_{b_0} y_n$.

4. Computing the products $\phi''_n \, \partial_{a_0} y_{n-i}$ for $n = 0, ...T-1;\ i = 1, ..., M-1$ and $\phi''_n \, \partial_{b_0} y_{n-i}$ for $n = 0, ...T-1;\ i = 0, ..., N-1$.

5. Computing the products $\phi'_n \, \partial^2_{a_0} y_{n-2i}$ for $n = 0, ...T-1;\ i = 1, ..., M-1$ and $\phi'_n \, \partial_{a_0 b_0} y_{n-2i}$ for $n = 0, ...T-1;\ i = 1, ..., \min\{M, N\} - 1$.

Step 1 requires $k''T$ operations; Step 2 takes $2MT$ operations; Step 3 takes $3T$ operations; Step 4 takes $T(M + N - 1)$ operations; and Step 5 requires $T\left(M + \min\{M, N\} - 2\right)$ operations. Totally, construction of the approximate Hessian requires about $\left(2M + \min\{M, N\} + N + k''\right) T$ operations.

When $T \sim M \sim N$, the complexity of constructing the approximate Hessian becomes $4N^2 + k''N$ and has the same order of magnitude as the complexity of gradient computation. See Table 1 for comparison of the computational complexity of the gradient, the exact Hessian and the approximate Hessian.

Exact solution of the Newton system using modified Cholesky decomposition requires $\frac{1}{6}(M + N - 1)^3$ operations for factorization and $(M + N - 1)^2$ operations for back/forward substitution, yielding in total about $\frac{1}{6}(M + N - 1)^3 + (M + N - 1)^2$ operations. Approximate solution of the Newton system requires solution of $\min\{M, N\} - 1$ modified systems (38) plus $|M - N|$ operations for obtaining the remaining elements of the vector d from the set of single equations (33). This requires about $k_N \cdot (\min\{M, N\} - 1) + |M - N|$ operations, where k_N stands for the complexity of a single 2×2 system solution. Efficient implementation allows to achieve $k_N \approx 40$.

Term	$\mathbf{T \gg M, N}$	$\mathbf{T \sim M \sim N, N_F = 10N}$
g	$(3M + N + k')\,T$	$2N^2 + (20 + k' + 40\log_2 10)N$ $+42N\log_2 N$
H	$(M^2 + N^2 + 6M + 2N + k'' - 2)\,T$	$804N^2\log_2 N + (20 + k'')N +$ $+(4 + 800\log_2 10)N^2$
$\tilde{H}$	$(2M + \min\{M, N\} + N + k'')\,T$	$4N^2 + k''N$

Table 1: Comparison of computational complexity of the gradient, the exact Hessian and the approximate Hessian for very large T and for T comparable with M, N.

5.4 Fast block-coordinate relative Newton step

In [35], it was shown that performing the fast relative Newton step in a block-coordinate manner is advantageous for quasi-maximum likelihood blind source separation. The same technique can be used in the proposed framework for quasi-ML blind deconvolution. Let us consider as an example the case when $M = 1$ and at each internal iteration a block of K coordinates is updated[1]. Let us also assume that $T \gg N$.

The use of line search requires evaluation of $f(a_n = 1, b_n = \delta_n + \Delta b_n; x)$, where Δb is the update of b in the current block. Coefficients B_k in f_1 are updated according to

$$\Delta B_k \;=\; \sum_{n \in \text{block}} \Delta b_n \, e^{-i\frac{2\pi n}{N_F}}. \tag{41}$$

The logarithms of $|B_k|^2$ have to be computed for all k's, resulting in total in $KN_F + k_L N_F$ operatins for updating f_1. y_n is updated according to

$$\Delta y_n \;=\; \sum_{k \in \text{block}} \Delta b_k \, x_{n-k}, \tag{42}$$

which results in about $4T\log_2 K + kT$ operations for updating f_2.

The most computationally difficult part of each internal iteration is the computation of K elements of the gradient vector and a $K \times K$ block from the approximate Hessian matrix, which for $T \gg N$ requires about $(2K + k' + k'')T$ operations. Internal iterations are repeated $\frac{N}{K}$ times at each external iteration, thus resulting in about $(2K + k' + k'')\frac{TN}{K}$ operations per external iteration. This is compared to the complexity of a "normal" fast relative Newton iteration, which requires about $(2N + k' + k'')T$ operations. Thus, block-coordinate update is advantageous if it makes the algorithm converge

$$\gamma \;=\; \frac{2NK + (k' + k'')N}{2NK + (k' + k'')K} = 1 + \frac{(k' + k'')(N - K)}{2NK + (k' + k'')K} \tag{43}$$

[1]For $M > 1$, it is not straightforward how to perform the fast update of the filtered signal y_n when only some coefficients of a_n are updated. Additional research is required to determine feasibility of the block-coordinate step in this case.

times faster than fast relative Newton method with full update. For $K \ll N$ and values of k', k'' from Section 2.3, γ becomes

$$\gamma \;=\; 1 + \frac{k' + k''}{2} \approx 4.67. \tag{44}$$

6 Sequential reduction of the smoothing parameter

When the source signal is sparse, the quality of the restored signal greatly improves with reduction of the smoothing parameter λ in the absolute value approximation. However, minimization of the quasi-ML function becomes more difficult and involves, in particular, very large values of the gradient and the Hessian. We address this problem by considering two methods of sequential refinement of the absolute value approximation: the first one based on sequential reduction of the smoothing parameter λ, and the second one involving the smoothing method of multipliers described in [36]. We limit our discussion to the case of super-Gaussian signals. Similar algorithms can be derived for sub-Gaussian signals.

6.1 Sequential optimization algorithm

Sequential reduction of the smoothing parameter consists of iterating the unconstrained minimization of the quasi-ML function, each time decreasing λ and initializing the minimization algorithm at the solution found in the previous iteration [25]. In combination with the relative optimization method described in Section 3.1, this results in the following sequential optimization algorithm:

Sequential optimization algoritm

1. Start with $\lambda^{(1)}$, some initial guess of $H^{(0)}(z) = \frac{B^{(0)}(z)}{A^{(0)}(z)}$, and $x_n^{(0)} = x_n$.
2. **For** $k = 1, 2, ..., K$
3. Compute current source signal estimate: $x_n^{(k)} = \frac{B^{(k-1)}(z)}{A^{(k-1)}(z)} \left[x_n^{(k-1)} \right]$.
4. Starting with $A(z) = B(z) = 1$, find
   $A^{(k)}(z), B^{(k)}(z) = \text{argmin} f(A(z), B(z); x_n^{(k)}, \lambda^{(k)})$.
5. Update the estimated restoration filter: $H^{(k)}(z) = \frac{B^{(k)}(z)}{A^{(k)}(z)} H^{(k-1)}(z)$.
6. Update the smoothing parameter $\lambda^{(k+1)} = \mu_\lambda \lambda^{(k)}$.
7. **End For**

The parameter $\mu_\lambda < 1$ determines the decay rate of the smoothing parameter and it is generally set to $\mu_\lambda \sim 10^{-1} \div 10^{-2}$. The block relative optimization algorithm can be modified similarly to allow sequential reduction of the smoothing parameter.

6.2 Smoothing method of multipliers

A substantial disadvantage of the sequential optimization algorithm is the fact that the smoothing parameter becomes extremely small in the neighbourhood of the solution, a fact that influences the accuracty of the obtained solution. An efficient alternative, known as *smoothing method of multipliers* was introduced in [36]. The proposed algorithm does not require extreme values of the smoothing parameter due to the use of multipliers and allows to converge to a very accurate solution.

Smoothing method of multipliers suggests to modify the absolute value approximation $\phi_\lambda(t)$ by forcing its derivative at $t = 0$ to be $\phi'_\lambda(0) = \mu$. The linear function μt is tangent to the graph of $\phi_{\lambda,\,\mu}(t)$ at the origin (see Figure 8). The parameter μ is used as an analog of Lagrange multiplier. The method was found especially efficient with the family of "soft" quadratic-logarithmic approximations of the absolute value, given by

$$\phi_{\lambda,\,\mu}(t) \;=\; \begin{cases} -t - p_1 \log \frac{t}{\tau_1} + q_1 & : \quad t < \tau_1 \le 0 \\ \frac{t^2}{2\lambda} + \mu t & : \quad \tau_1 \le t \le \tau_2 \\ t - p_2 \log \frac{t}{\tau_2} + q_2 & : \quad t < \tau_1 \le 0, \end{cases} \tag{45}$$

where

$$\begin{array}{ll} \tau_1 = -\frac{1}{2}(1+\mu)\lambda & \tau_2 = \frac{1}{2}(1-\mu)\lambda \\ p_1 = \lambda^{-1}\tau_1^2 & p_2 = \lambda^{-1}\tau_2^2 \\ q_1 = \frac{1}{2\lambda}\tau_1^2 + (\mu+1)\tau_1 & q_2 = \frac{1}{2\lambda}\tau_2^2 + (\mu-1)\tau_2. \end{array}$$

These functions have bounded third-order derivatives and hence are well-matched to the Newton method [36].

Using $\phi_{\lambda,\,\mu}(\cdot)$ for the absolute value approximation, the objective function becomes

$$f \;=\; -\frac{1}{2N_F} f_1 + \frac{\nu_S}{N_S} f_S + \frac{1}{T} \sum_{n=0}^{T-1} \phi_{\lambda,\,u_n}(y_n) \tag{46}$$

and it is analogous to an *augmented Lagrangian*, where $\{u_n : -1 \le u_n \le 1\}_{n=0}^{T-1}$ are Lagrange multipliers. Smoothing method of multipliers suggests the following iterative algorithm:

Smoothing method of multipliers

1. Start with $\lambda^{(1)}$, $u_n^{(1)} = 0$, $H^{(0)}(z) = \frac{B^{(0)}(z)}{A^{(0)}(z)}$ and $x_n^{(0)} = x_n$.

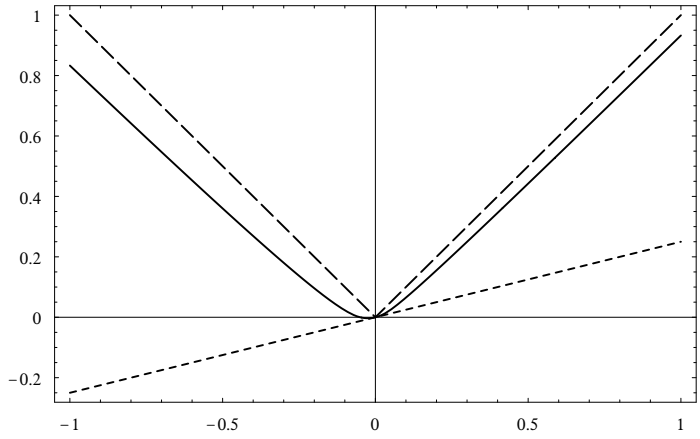

Figure 8: $\phi_{\lambda,\,\mu}(t)$ - solid line; $|t|$ - dashed line; μt - dotted line.

2. **For** $k = 1, 2, ..., K$
3. Compute current source signal estimate: $x_n^{(k)} = \frac{B^{(k-1)}(z)}{A^{(k-1)}(z)} \left[x_n^{(k-1)}\right]$.
4. Starting with $A(z) = B(z) = 1$, find
   $A^{(k)}(z), B^{(k)}(z) = \text{argmin} f(A(z), B(z); x_n^{(k)}, \lambda^{(k)}, u_n^{(k)})$.
5. Update the estimated restoration filter: $H^{(k)}(z) = \frac{B^{(k)}(z)}{A^{(k)}(z)} H^{(k-1)}(z)$.
6. Update the multipliers $u_n^{(k+1)} = \phi'_{\lambda^{(k)},\,u_n^{(k)}}(y_n)$, where $y_n = \frac{B^{(k)}(z)}{A^{(k)}(z)} \left[x_n^{(k)}\right]$.
7. Optional: update the smoothing parameter $\lambda^{(k+1)} = \mu_\lambda \lambda^{(k)}$.
8. **End For**

In practice, in order to stabilize the method, the relative change of the multipliers should be restricted:

$$\begin{aligned} \gamma_1 < \frac{u_n^{(k+1)} + 1}{u_n^{(k)} + 1} < \gamma_2 \\ \gamma_1 < \frac{1 - u_n^{(k+1)}}{1 - u_n^{(k)}} < \gamma_2 \\ \left|u_n^{(k+1)}\right| < 1 - \delta, \end{aligned} \tag{47}$$

where typically $\gamma_1 = \gamma_2^{-1} \sim 2$ and $\delta \sim 10^{-6}$. Furthermore, $\lambda^{(k)}$ is restricted to be no smaller than some $\lambda_{\min} \sim 10^{-3}$ [36].

7 Numerical results

In this section, we present several simulation results to examine performance of the algorithms described in this work. Source signals in all experiments were generated as i.i.d. processes with Gauss-Bernoully, generalized Laplacian or discrete uniform (PAM) distributions (see Appendix Appendix B). Unless stated otherwise, zero sensor noise was assumed. Convolutive systems were modeled either as FIR or rational IIR filters with different orders depending on the experiment. Coefficients were chosen randomly under the constraint of filter stability. In all tests, unless stated otherwise, $N_F = 256$ and $N_S = 1024$ were used, and gradient norm below 10^{-10} served as the stopping criterion for optimization algorithms. All algorithms were implemented in MATLAB and executed on an ASUS portable computer with Intel Pentium IV Mobile processor and 640MB RAM. All execution time measurements should be interpreted merely as upper bounds on execution time. Signal to interference ratio (SIR) was used as a quality measure. An exact definition of this measure is given in the following section.

7.1 Restoration quality measure

Estimated source signal $\tilde{s}$ can be described as source signal s (up to a delay Δ and multiplicative factor c), contaminated by q, which describes the *deconvolution error* [37]

$$\tilde{s}_n \;=\; (g * s)_n = c \cdot s_{n-\Delta} + q_n. \tag{48}$$

A common restoration quality measure known as *intersymbol interference* (ISI) or *signal to interference ratio* (SIR) is defined as the ratio of variances of $c \cdot s$ and q

$$SIR \;=\; \frac{c^2 \cdot \mathbf{E}\left\{s_n^2\right\}}{\mathbf{E}\left\{q_n^2\right\}}. \tag{49}$$

In simulations where the global system response is known, SIR can be expressed in terms of g as

$$SIR \;=\; \frac{g_\Delta^2}{\sum_{n\neq\Delta} g_n^2} = \frac{\|g\|_\infty^2}{\|g\|_2^2 - \|g\|_\infty^2}. \tag{50}$$

In reality, g_n is evaluated on a finite interval $n = 0, .., T_S - 1$. In our experiments $T_S = 10^3$ was used.

7.2 Test I: Rational restoration kernel

Most works on blind deconvolution focus their attention on FIR restoration kernels. In this work, we introduced a rational restoration kernel, containing besides the FIR part (numerator $B(z)$) also an all-pole IIR part (denominator $A(z)$). Importance of the all-pole part is

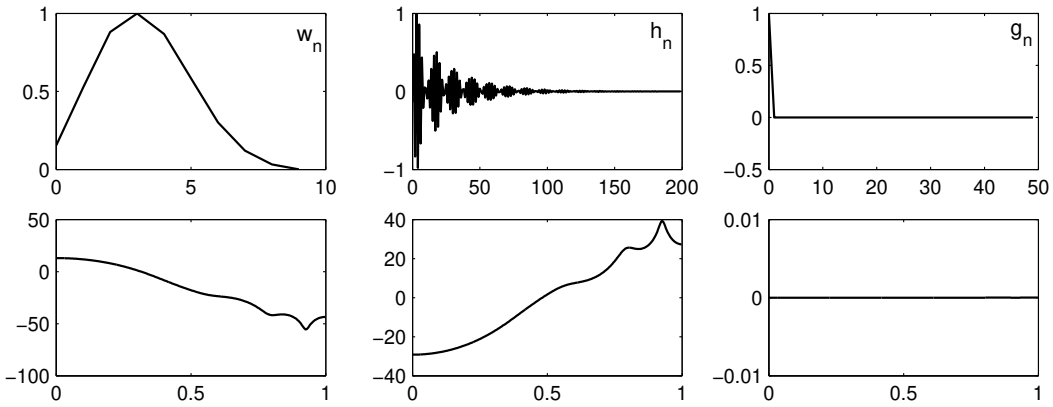

Figure 9: Impulse responses (top row) and transfer functions (bottom row) of $W(z)$ (left) and $H(z)$ (middle) and $G(z)$ (right) for SIR = 192.77 dB obtained in the experiment. The impulse response of $H(z)$ is truncated.

demonstrated in the following experiment. Input signal was generated by filtering an i.i.d. Gauss-Bernoully process (see Appendix B.1) with sparsity $\rho = 0.2$ and length $T = 1000$ samples by a short minimum phase FIR filter $W(z)$ of order 9 (Figure 9, left). Since the impulse response of $W^{-1}(z)$ decays relatively slowly (Figure 9, middle), restoration using an FIR kernel would require about 100 coefficients to achieve reasonable quality. On the other hand, an all-pole kernel of order 9 is sufficient to ideally restore the source signal.

Sequential optimization algorithm was used with λ decreasing from 1 to 10^{-10} with a rate of $\mu_\lambda = 0.1$, and ν_S decreasing from 1 with a rate of $\mu_\nu = 2$. Unconstrained minimization was carried out using Newton method. Three configurations were tested: pure FIR with unit denominator and numerator of order $n - 1$ ($M = 1, N = n$), all-pole IIR ($M = n, N = 1$) and a rational kernel with numerator and denominator of equal order ($M = N = \frac{n+1}{2}$), where n stands for number of variables in the problem. Different values of n were tested.

Figure 10 depicts the restoration SIR as function of the number optimization variables n for different assignments of the degrees of freedom to restoration kernel numerator and denominator. SIR higher than 10 dB was obtained for all-pole IIR kernel starting from $M \geq 8$, for the rational kernel starting from $N = M \geq 4$, and for the FIR kernel starting only from $N \geq 40$. It can be concluded that rational restoration kernels are generally advantageous than the FIR ones.

7.3 Test II: Convergence of the Newton method

The following test demonstrates convergence of the Newton method used for unconstrained minimization of f. Input signal was generated by filtering an i.i.d. Gauss-Bernoully process (see Appendix B.1) with sparsity $\rho = 0.2$ and length $T = 1000$ samples by a FIR filter of

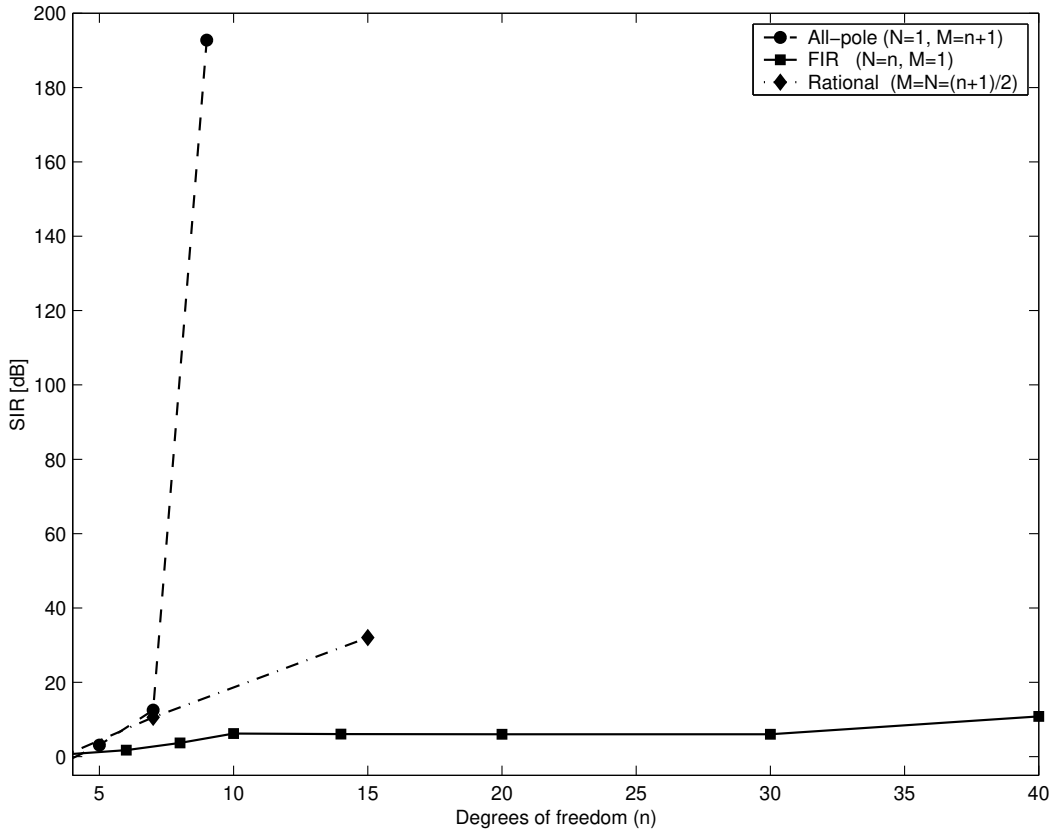

Figure 10: Restoration SIR as function of degrees of freedom (optimization variables in the problem) for different restoration kernel configurations. Solid: pure FIR ($M = 1, N = n$); dashed: all-pole IIR ($M = n, N = 1$); dash-dotted: rational kernel ($M = N = \frac{n+1}{2}$).

order 19 (Figure 11, left). Restoration was performed by a FIR filter of size $M = 1, N = 50$ and the smoothing parameter was set to $\lambda = 0.001$. No stability barrier was used. Convergence of steepest descent method with backtracking line search and Newton method were compared. Figure 12 reveals that Newton method has quadratic convergence at the end, which allows to achieve very small norm of gradient and SIR about 35 dB, whereas the steepest descent method achieves gradient norm no lower than 10^{-9} and SIR slightly higher than 15 dB. Figure 11 presents the restoration kernel response (middle) and the global system response (right) obtained using the Newton algorithm.

7.4 Test III: Sequential optimization

The following test demonstrates how the sequential optimization algorithm presented in Section 6 achieves very high restoration quality by gradual reduction the smoothing param-

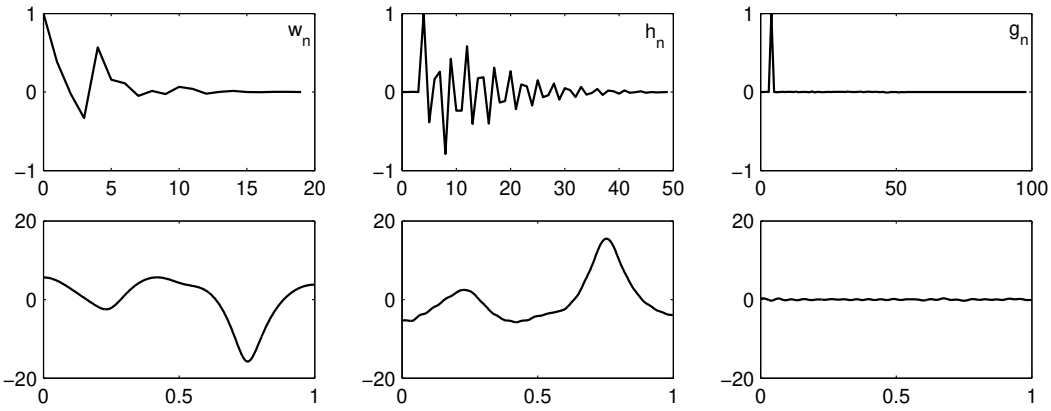

Figure 11: Impulse responses (top row) and transfer functions (bottom row) of $W(z)$ (left), $H(z)$ (middle) and $G(z)$ (right).

eter. Input signal was generated by filtering an i.i.d. Gauss-Bernoully process (see Appendix B.1) with sparsity $\rho = 0.2$ and length $T = 1000$ samples by an all-pole IIR filter of order 9. FIR restoration kernel of order 9 was used. Sequential optimization algorithm was used with λ decreasing from 1 to 3.28×10^{-11} with a rate of $\mu_\lambda = 0.2$, and ν_S decreasing from 1 with a rate of $\mu_\nu = 2$. Unconstrained minimization was carried out using Newton method. Problematic convergence was observed starting from $\lambda = 10^{-7}$. Figure 13 presents the SIR as function of λ. Restoration quality achieved for very small values of the smoothing parameter (SIR about 180 dB) can be considered ideal for any practical use.

7.5 Test IV: Sensitivity to noise

In order to study sensitivity of the algorithm to noise, Test III where practically ideal restoration was achieved, was repeated under presence of sensor noise with different SNRs (see Figure 14). Figure 15 presents the achieved restoration SIR as function of input SNR. Reasonable restoration quality is achieved for SNR higher than 15-20 dB.

7.6 Test V: Relative optimization

The following test demonstrates the relative optimization algorithm described in Section 3.1. Input signal was generated by filtering an i.i.d. Gauss-Bernoully process (see Appendix B.1) with sparsity $\rho = 0.2$ and length $T = 1000$ samples by an FIR filter of order 10. Restoration was performed by an FIR filter of size $M = 1, N = 50$ and a rational kernel of size $M = N = 5$ with smoothing parameter set to $\lambda = 10^{-3}$. Newton method, relative Newton method, fast relative Newton method and its limited memory version ($K = T$) were compared. Figure 18 depicts the convergence of the four methods for the FIR restoration kernel

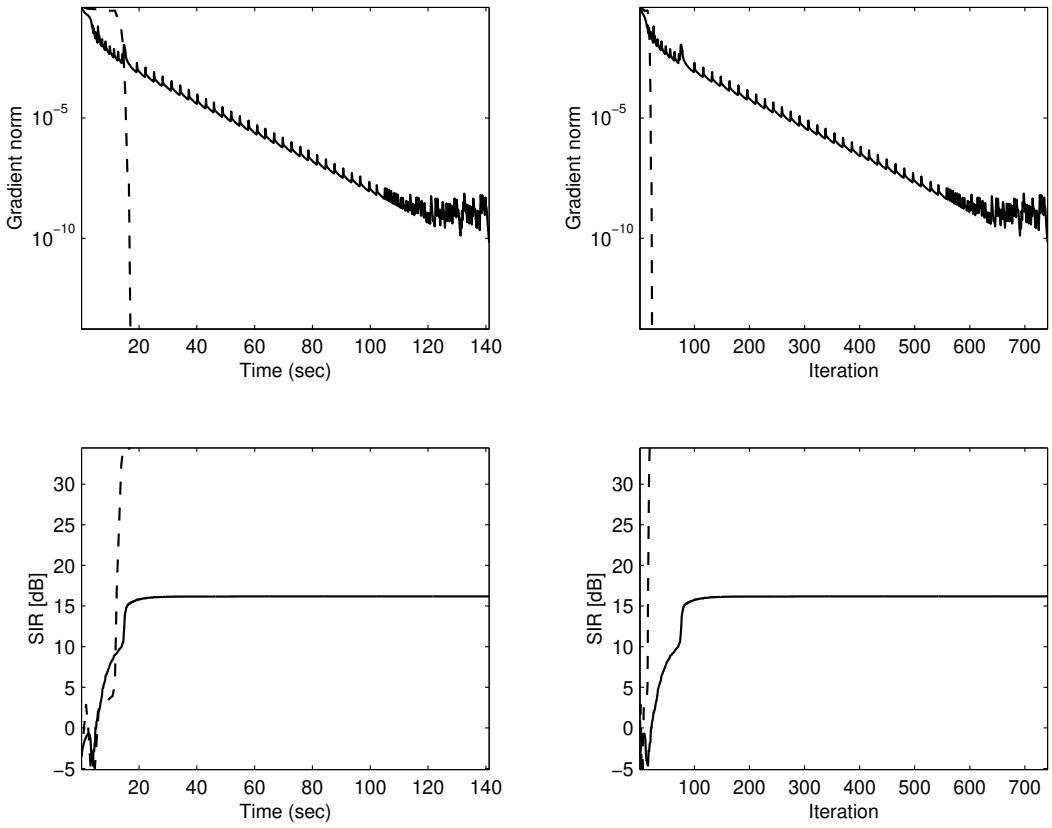

Figure 12: Convergence of the steepest descent method (solid) and the Newton method (dashed). Top row: norm of gradient; bottom row: signal to interference rate as function of time (left) and iteration number (right).

in sense of SIR and gradient norm as function of time and iteration number. In Figure 17, convergence for the IIR kernel is shown.

Fast relative Newton method and its limited memory version demonstrated virtually identical convergence both in sense of SIR and gradient norm. Newton and relative Newton methods converged in less iterations that the fast relative Newton method and its memory limited version and demonstrated quadratic convergence. However, the use of special Hessian structure significantly reduced the computational complexity of each iteration of the fast relative Newton method and its memory limited version and allowed these two methods to converge faster in the sense of execution time.

Rational restoration kernel achieved slightly lower SIR of 31.70 dB compared to 35.50 dB achieved by the FIR kernel, and demonstrated worse convergence in the number of iterations. However, the use of the rational kernel involved only 9 optimization variables, compared to the 32 variables required for the FIR kernel.

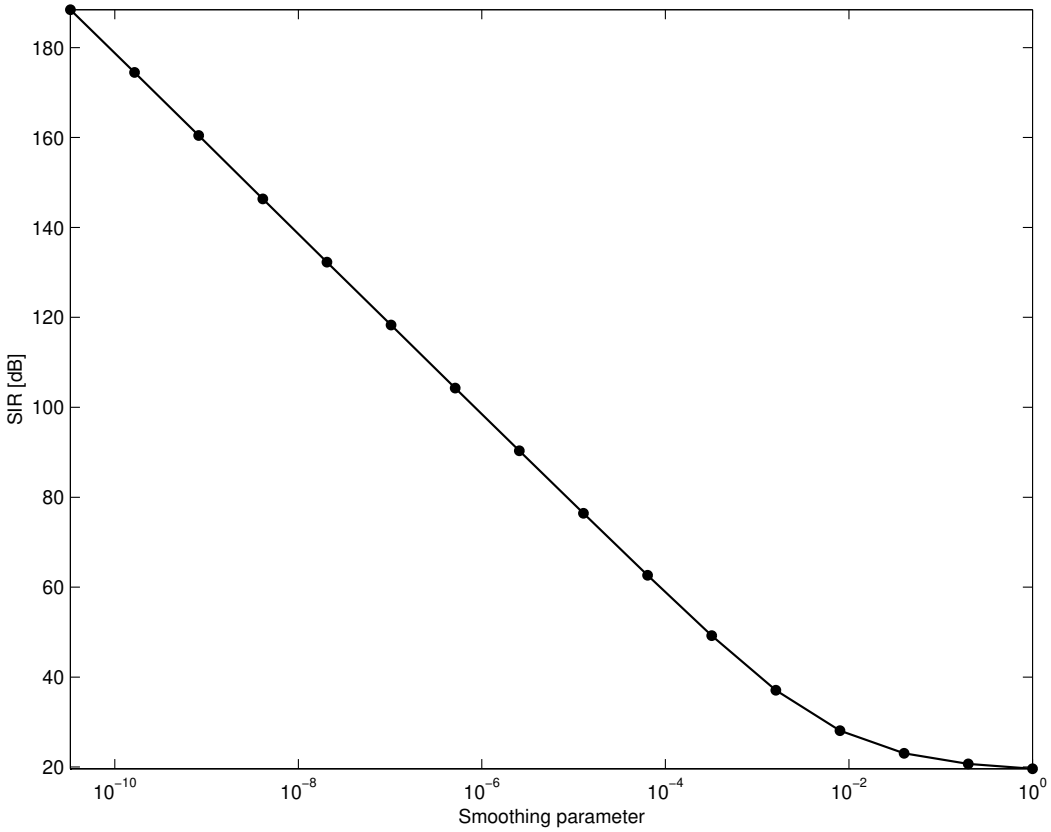

Figure 13: SIR obtained by sequential optimization algorithm presented as function of the smoothing parameter λ.

7.7 Test VI: Block relative optimization

The following test demonstrates the block relative optimization algorithm for online processing described in Section 3.2. Two super-Gaussian source signals with Gauss-Bernoully distribution with sparsity $\rho = 0.2$ and generalized Laplacian distribution with $\alpha = 0.5$, and one sub-Gaussian 2-level PAM process (see Appendix Appendix B), all of length $T = 2.5 \times 10^5$ were generated. Convolution system was modeled by an FIR filter of order 99.

FIR restoration kernel of order 31 ($M = 1, N = 32$) was used. Block relative optimization algorithm was applied with block size $L = 512$. Limited memory fast relative Newton step was used with kernel size limited to $K = 512$ samples. In case of super-Gaussian signals, λ was gradually reduced from 10^{-3} till 10^{-6} with a rate of $\mu_\lambda = 0.1$. In case of the sub-Gaussian process, k was reduced from 2 to 20. No stability barrier was used.

Figures 18–20 (dashed) show the restoration SIR as function of input signal sample number. The algorithm converged to an SIR of about 30–35 dB for the Gauss-Bernoully signal,

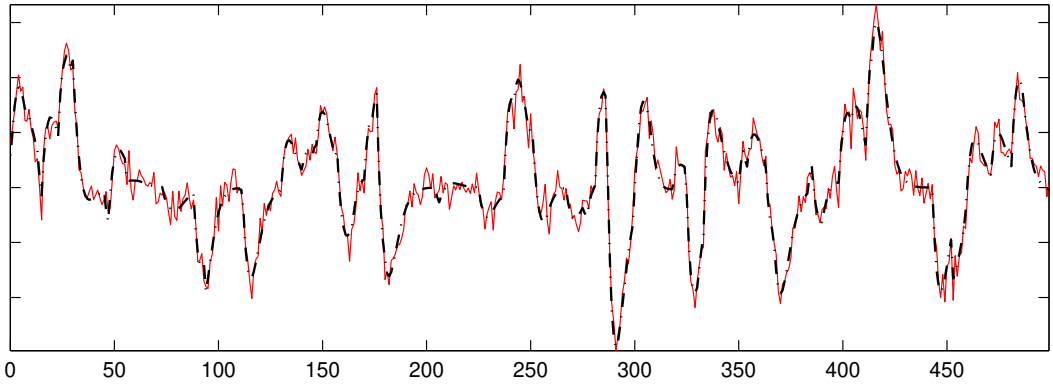

Figure 14: Part of the observed signal x in the noiseless case (dash-dotted) and at presence of noise, SNR = 15 dB (solid).

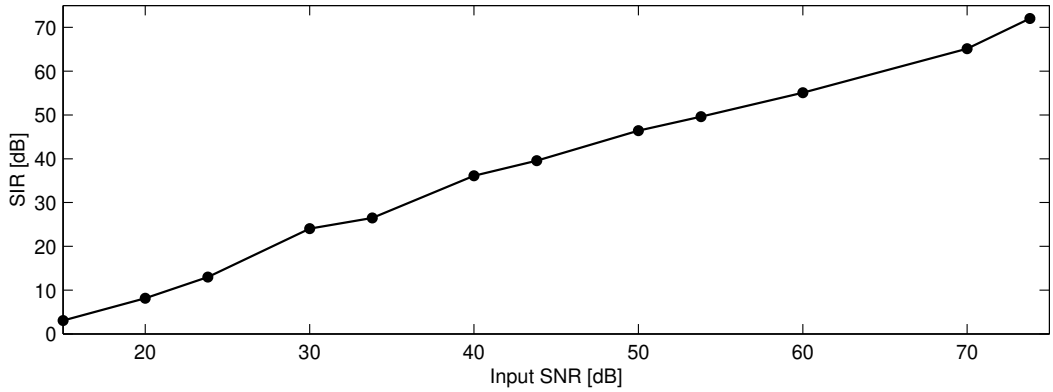

Figure 15: SIR obtained by sequential optimization algorithm as function of input SNR.

about 25–30 dB for the generalized Laplacian signal, and about 30 dB for the PAM signal. SIR of 33.77 dB, 33.11 dB and 30.92 dB for the Gauss-Bernoully, generalized Laplacian and PAM signals, respectively, was reached by the batch mode version of the algorithm applied to the first 4096 samples of the input. In Figure 21, output constellations of the limited memory fast block relative Newton algorithm for the PAM input signal at different blocks are depicted to visualize restoration quality.

For comparison, the natural gradient deconvolution algorithm proposed by Amari *et al.* [17, 13] and its block-wise frequency-domain version proposed by Joho *et al.* [19, 20], with block size of 512 samples and 32 FIR coefficients and step $\mu = 0.02$, achieved SIR of 10–20 dB for all source types (Figures 18–20). Moreover, the natural gradient and Joho's algorithms have slower convergence in terms of restoration SIR. MATLAB implementation of the memory limited block relative Newton algorithm demanded in average 1.59×10^{-4}

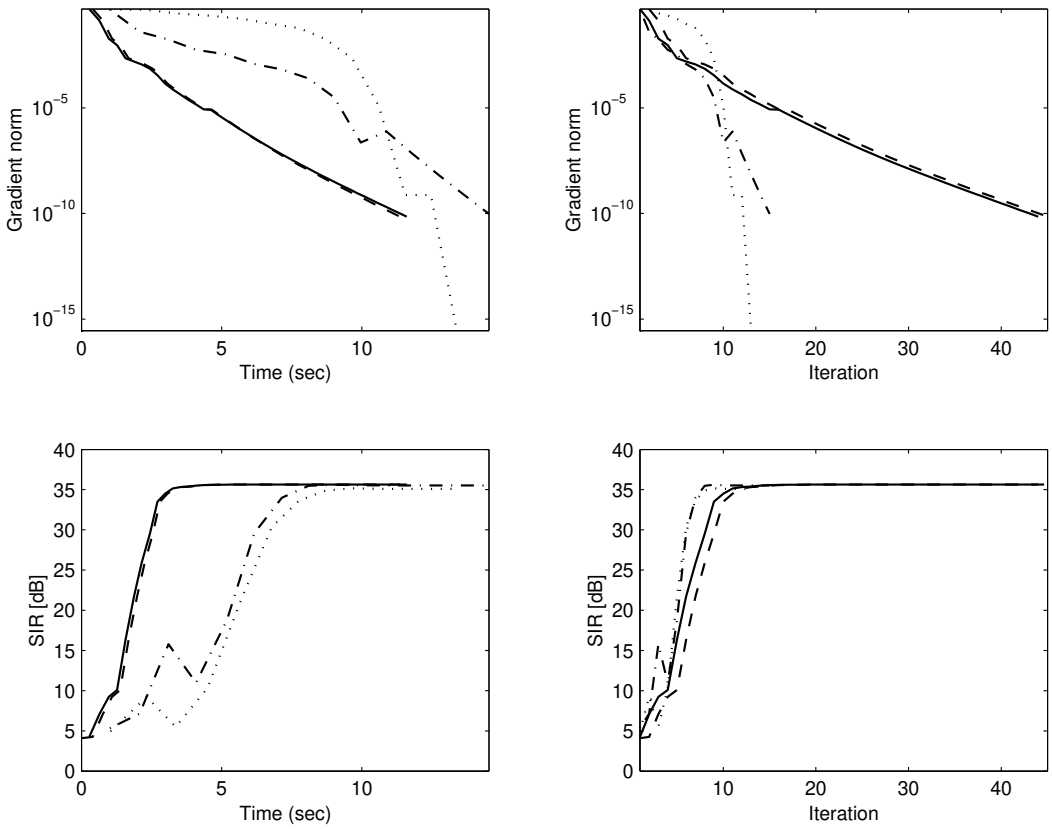

Figure 16: Convergence of the Newton method (dotted), the relative Newton method (dash-dotted), the fast relative Newton method (solid) and its limited memory version (dashed) for $M = 1, N = 32$.

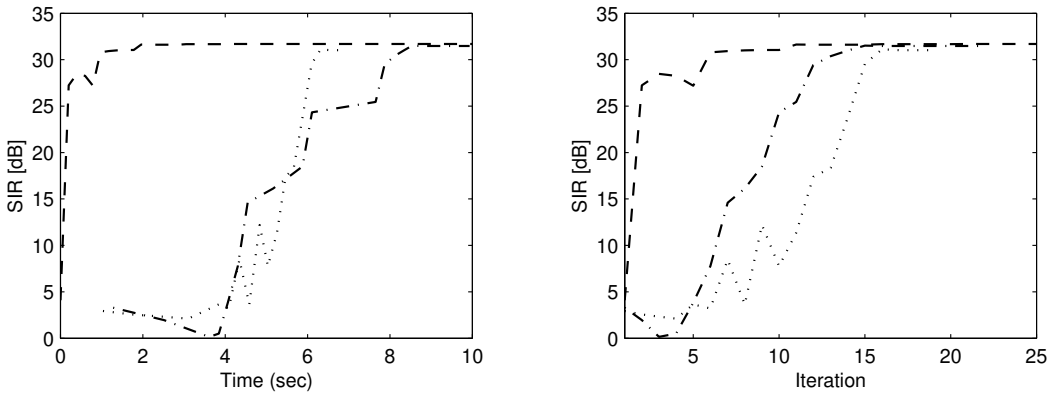

Figure 17: Convergence of the Newton method (dotted), the relative Newton method (dash-dotted), the memory limited fast relative Newton method (dashed) for $M = N = 5$. Fast relative Newton method demonstrated convergence identical to that of the memory limited version.

sec per input sample, thus allowing to process in real time signals with sampling frequency up to 6.27 KHz. For comparison, Joho's algorithm required about 1.86×10^{-5} sec per input sample, which is equivalent to sampling frequency of 53.8 KHz. It can be concluded that although having computational complexity of the same order as gradient-based methods, fast relative Newton method is almost 10 times slower compared to an efficient implementation of the natural gradient deconvolution algorithm. Still, it is fast enough and suitable for real-time processing and gives significantly better restoration quality (improvement by 10 dB to 20 dB) and faster convergence.

8 Conclusions

We have presented a relative optimization framework for quasi-maximum likelihood single channel blind deconvolution and studied relative Newton method as its particular instance. Tri-diagonal structure of the Hessian in a neighbourhood of the solution allowed to derive a fast version of the relative Newton algorithm, with iteration complexity comparable to that of gradient methods. The batch mode relative Newton method was extended to the online case by considering block-wise relative optimization. The latter algorithm is capable of handling time-varying convolution systems. Additionally, in this work we introduced rational deconvolution kernels, which constitute a richer and more flexible family of filters than the traditionally used FIR kernels, and often allow to reduce the optimization problem size.

The use of sequential optimization and smoothing method of multipliers for gradual refinement of the absolute value approximation was proposed. Combined with the relative

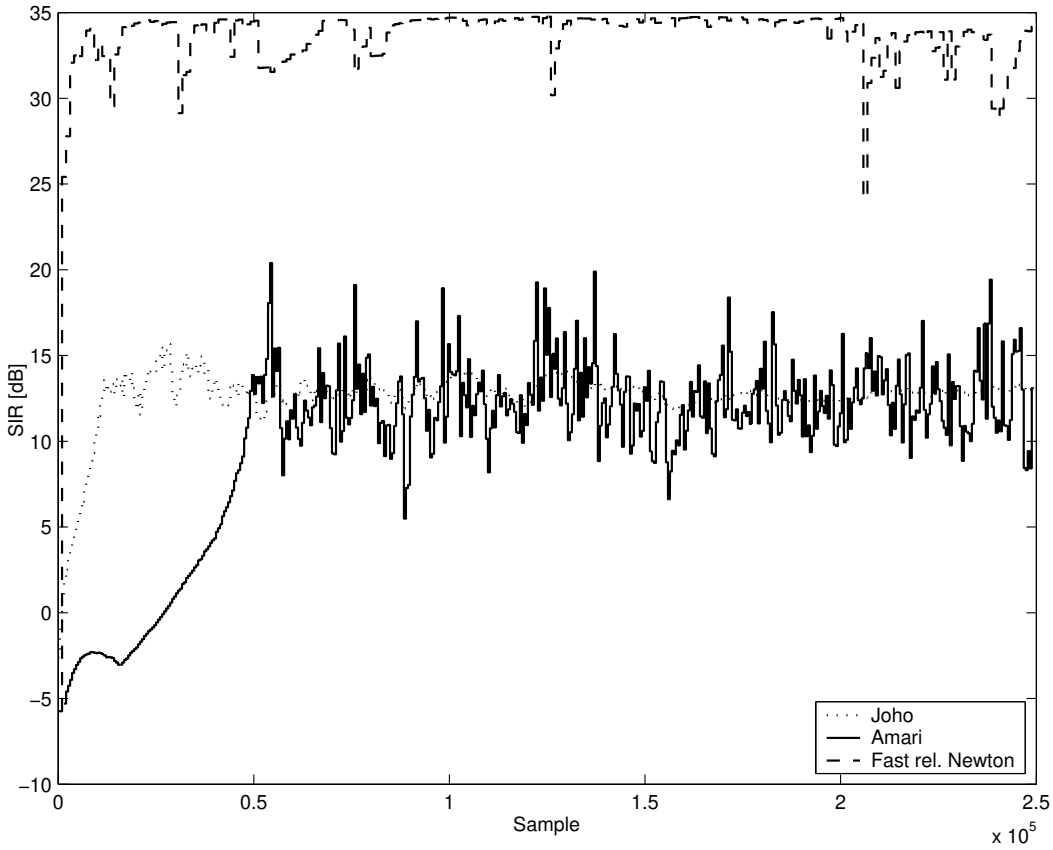

Figure 18: SIR of the limited memory fast block relative Newton method (dashed), and the natural gradient algorithm (solid) and Joho's algorithm (dotted) as function of input signal sample number for the Gauss-Bernoully input signal.

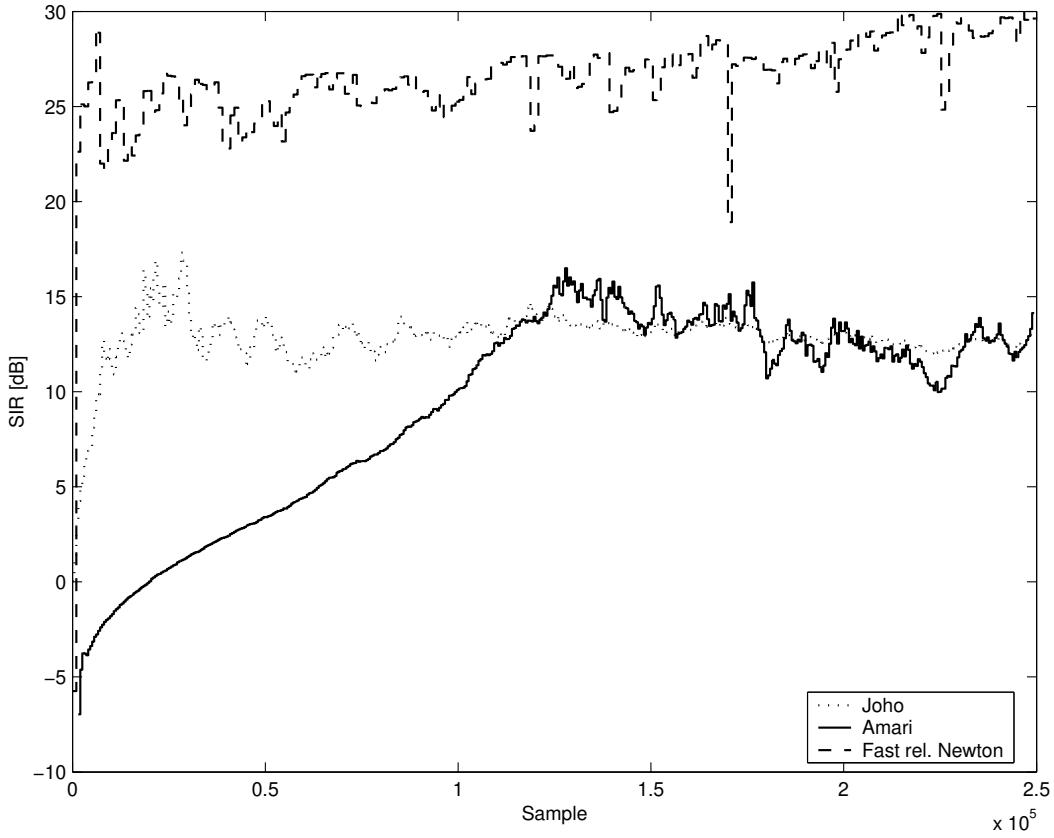

Figure 19: SIR of the limited memory fast block relative Newton method (dashed), and the natural gradient algorithm (solid) and Joho's algorithm (dotted) as function of input signal sample number for the generalized Laplacian signal.

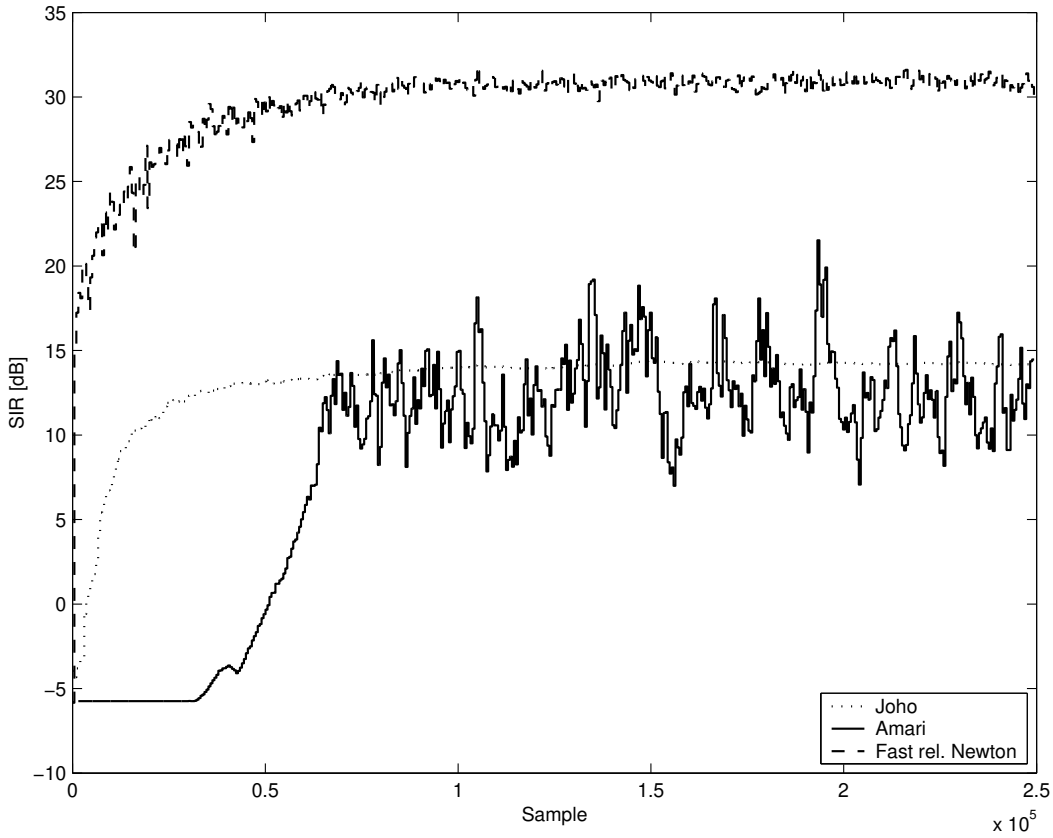

Figure 20: SIR of the limited memory fast block relative Newton method (dashed), and the natural gradient algorithm (solid) and Joho's algorithm (dotted) as function of input signal sample number for the PAM signal.

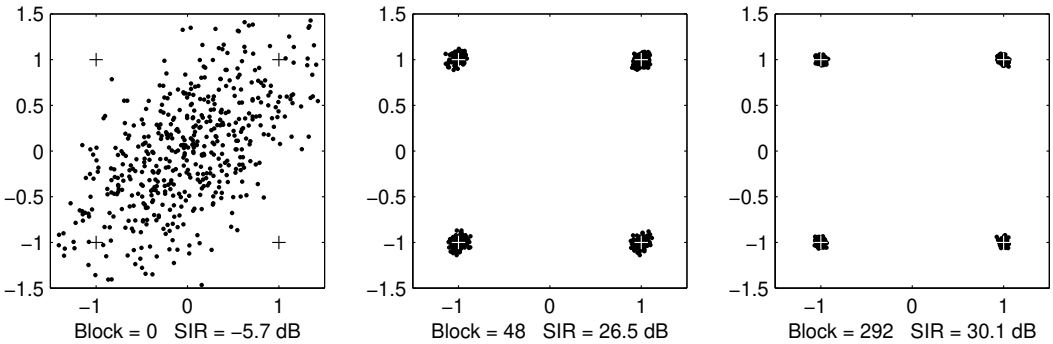

Figure 21: Output constellations y_n vs. y_{n-1} of the limited memory fast block relative Newton algorithm for the 2-level PAM signals. Crosses indicate ideal restoration.

Newton method, gradual refinement of the absolute value approximation resulted in an extremely accurate blind deconvolution algorithm with very fast convergence. Although our discussion was mainly focused on super-Gaussian source signals, for which the absolute value is a good approximation of $-\log p(\cdot)$, by a suitable choice of $\phi(\cdot)$ and minor modifications, sub-Gaussian signals can be also handled.

Simulation results demonstrate the efficiency of the proposed methods in low to medium-noise conditions. Possible applications are in acoustics and communications, especially where high accuracy is required. We are currently working on extending the relative Newton blind deconvolution algorithm to multichannel and two-dimensional cases. The latter seems to be of high importance, since many processes in optics and image processing can be described by a relatively short FIR model, which makes the use of rational deconvolution kernels especially attractive.

Acknowledgement

This research has been supported by the Ollendorff Minerva Center, by the Fund for Promotion of Research at the Technion, by the Israeli Ministry of Science, by the HASSIP Research Network Program HPRN-CT-2002-00285, sponsored by the European Commission. Alexander Bronstein expresses his gratitude to Susy Pitzanti for creating a warm atmosphere and an outstanding working environment in the beautiful island of Sardinia, Italy, where part of this work was written.

Appendix A Proofs and derivations

A.1 Proof of Proposition 1 – Gradient and Hessian of f_1

Gradient of f_1

Let us use the matrix notation to express the DFT coefficients A_k and B_k for $k = 0, ..., N_F - 1$ as elements of the vectors

$$\begin{aligned} A &= \Phi_A \, a \\ B &= \Phi_B \, b, \end{aligned} \tag{51}$$

where Φ_A and Φ_B are $N_F \times M$ and $N_F \times N$ DFT matrices, respectively. Φ_A and Φ_B can be regarded as truncated versions of the $N_F \times N_F$ DFT matrix defined by

$$(\Phi)_{mn} = \frac{1}{\sqrt{N_F}} \exp\left\{-\frac{2\pi i \, mn}{N_F}\right\} \tag{52}$$

and can be implemented efficiently by applying the FFT to the zero-padded versions of a and b.

Let us also expand f_1 into

$$\begin{aligned} f_1 &= \sum_{k=0}^{N_F-1} \log B_k B_k^* - \log A_k A_k^* \\ &= \sum_{k=0}^{N_F-1} \log B_k + (\log B_k)^* - \log A_k - (\log A_k)^* , \end{aligned} \tag{53}$$

and let us denote by $(\Phi_B)_k^T$ the k-th row vector of Φ_B. Then, B_k can be expressed as $B_k = (\Phi_B)_k^T\, b$, and the gradient of the term $\log B_k$ for $k = 0, ..., N_F - 1$ is given by

$$\nabla_b \left(\log B_k\right) = \frac{1}{B_k} \nabla_b\, B_k = \frac{1}{B_k}\, \left(\Phi_B\right)_k . \tag{54}$$

Summation on k yields

$$\begin{aligned} \nabla_b \left(\sum_{k=0}^{N_F-1} \log B_k \right) &= \sum_{k=0}^{N_F-1} \frac{1}{B_k}\, \left(\Phi_B\right)_k \\ &= \begin{bmatrix} | & & | \\ (\Phi_B)_0 & \dots & (\Phi_B)_{N_F-1} \\ | & & | \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{B_0} \\ \vdots \\ \frac{1}{B_{N_F-1}} \end{bmatrix} \\ &= \Phi_B^T B' = \left(\Phi_B^H B'^*\right)^* , \end{aligned} \tag{55}$$

where $B' = \begin{bmatrix} B_0^{-1} & \dots & B_{N_F-1}^{-1} \end{bmatrix}^T$, and Φ_B^H can be implemented efficiently using the inverse FFT. Similarly, gradients of the other three terms of f_1 in (53) can be obtained, yielding

$$\nabla_{ab}\, f_1 = \begin{bmatrix} \nabla_a\, f_1 \\ \nabla_b\, f_1 \end{bmatrix}, \tag{56}$$

where

$$\begin{aligned} \nabla_a\, f_1 &= -\Phi_A^H A'^* - \left(\Phi_A^H A'^*\right)^* \\ \nabla_b\, f_1 &= \Phi_B^H B'^* + \left(\Phi_B^H B'^*\right)^* , \end{aligned} \tag{57}$$

and $A' = \begin{bmatrix} A_0^{-1} & \dots & A_{N_F-1}^{-1} \end{bmatrix}^T$.

Hessian of f_1

Let us first evaluate the Hessian of the first term

$$q = \sum_{k=0}^{N_F-1} \left(\log B_k\right)^* \tag{58}$$

in (53). Previously we have seen that the gradient of q with respect to b was given by

$$g \;=\; \nabla_b\, q = \Phi_B^T B'. \tag{59}$$

The differential of g is given by

$$dg \;=\; \Phi_B^T \begin{bmatrix} dB'_0 \\ \vdots \\ dB'_{N_F-1} \end{bmatrix} = \Phi_B^T \begin{bmatrix} \nabla_b B_0'^T db \\ \vdots \\ \nabla_b B_{N_F-1}'^T db \end{bmatrix} = \Phi_B^T \begin{bmatrix} \nabla_b B_0'^T \\ \vdots \\ \nabla_b B_{N_F-1}'^T \end{bmatrix} db. \tag{60}$$

Substituting

$$\nabla_b B'_k \;=\; \nabla_b \left(\frac{1}{B_k}\right) = -\frac{1}{B_k^2}\,\left(\Phi_B\right)_k, \tag{61}$$

one has

$$\begin{aligned} dg \;&=\; -\Phi_B^T \begin{bmatrix} \frac{1}{B_0^2}\,\left(\Phi_B\right)_0^T \\ \vdots \\ \frac{1}{B_{N_F-1}^2}\,\left(\Phi_B\right)_{N_F-1}^T \end{bmatrix} db = -\Phi_B^T \begin{bmatrix} \frac{1}{B_0^2} & & \\ & \ddots & \\ & & \frac{1}{B_{N_F-1}^2} \end{bmatrix} \Phi_B\, db \\ &=\; \Phi_B^T\, \mathrm{diag}\left\{B''\right\} \Phi_B\, db = \left(\Phi_B^H\, \mathrm{diag}\left\{B''\right\}^H\right)^* \Phi_B\, db, \end{aligned} \tag{62}$$

where $B'' = -\begin{bmatrix} B_0^{-2} & \ldots & B_{N_F-1}^{-2} \end{bmatrix}^T$. The Hessian of q is a linear mapping H_q defined via the differential of the gradient g as

$$dg \;=\; H_q\, db;$$

therefore,

$$H_q \;=\; \left(\Phi_B^H\, \mathrm{diag}\left\{B''\right\}^H\right)^* \Phi_B. \tag{63}$$

Similarly, Hessians of the other three terms of f_1 can be obtained. Finally, we obtain

$$\nabla_{ab}^2\, f_1 \;=\; \begin{bmatrix} H_{aa} & \\ & H_{bb} \end{bmatrix}, \tag{64}$$

where

$$\begin{aligned} H_{aa} \;&=\; -\left(\Phi_A^H\, \mathrm{diag}\left\{A''\right\}^H\right)^* \Phi_A - \Phi_A^H \left(\mathrm{diag}\left\{A''\right\} \Phi_A\right)^* \\ H_{bb} \;&=\; \left(\Phi_B^H\, \mathrm{diag}\left\{B''\right\}^H\right)^* \Phi_B + \Phi_B^H \left(\mathrm{diag}\left\{B''\right\} \Phi_B\right)^* \end{aligned} \tag{65}$$

and $A'' = -\begin{bmatrix} A_0^{-2} & \ldots & A_{N_F-1}^{-2} \end{bmatrix}^T$. The terms H_{aa} and H_{bb} can be computed efficiently using the forward and the inverse FFT. ■

A.2 Proof of Proposition 2 – Derivatives of y_n

Let us use first express y_n in the Z-transform domain:

$$A(z)\,Y(z) \;=\; B(z)\,X(z), \tag{66}$$

where

$$\begin{aligned} A(z) &= \sum_k a_k\,z^{-k} \\ B(z) &= \sum_k b_k\,z^{-k} \end{aligned} \tag{67}$$

Differentiating (66) with respect to a_i yields

$$z^{-i}\,Y(z) + A(z)\,\partial_{a_i}Y(z) \;=\; 0,$$

from where

$$\partial_{a_i}Y(z) \;=\; -\frac{z^{-i}}{A(z)}\,Y(z) = -\frac{z^{-i}B(z)}{A^2(z)}\,X(z). \tag{68}$$

Differentiating again with respect to a_j yields

$$z^{-i}\,\partial_{a_j}Y(z) + z^{-j}\,\partial_{a_i}Y(z) + A(z)\,\partial^2_{a_ia_j}Y(z) \;=\; 0.$$

Substituting (68) yields

$$-2\,\frac{z^{-(i+j)}B(z)}{A^2(z)}\,X(z) + A(z)\,\partial^2_{a_ia_j}Y(z) \;=\; 0,$$

from where

$$\partial^2_{a_ia_j}Y(z) \;=\; 2\,\frac{z^{-(i+j)}B(z)}{A^3(z)}\,X(z). \tag{69}$$

Differentiating (66) with respect to b_i yields

$$A(z)\,\partial_{b_i}Y(z) \;=\; z^{-i}\,X(z),$$

from where

$$\partial_{b_i}Y(z) \;=\; \frac{z^{-i}}{A(z)}\,X(z). \tag{70}$$

Differentiating again with respect to b_j yields

$$\partial^2_{b_ib_j}Y(z) \;=\; 0, \tag{71}$$

whereas differentiating with respect to a_j yields

$$z^{-j}\,\partial_{b_i} Y(z) + A(z)\,\partial^2_{b_i a_j} Y(z) \;=\; 0.$$

Substituting (70) yields

$$\partial^2_{b_i a_j} Y(z) \;=\; -\frac{z^{-j}}{A(z)}\,\frac{\partial Y(z)}{\partial b_i} = -\frac{z^{-(i+j)}}{A^2(z)}\,X(z). \tag{72}$$

Expressing the first- and the second-order derivatives of $Y(z)$ with respect to the coefficients of the restoration filter numerator and denominator a_i, b_j in the time domain, one has

$$\begin{aligned}
y_n &= B(z)A^{-1}(z)\,[x_n] \\
\partial_{a_i} y_n &= -A^{-1}(z)\,[y_{n-i}] \\
\partial_{b_i} y_n &= A^{-1}(z)\,[x_{n-i}] \\
\partial^2_{a_i a_j} y_n &= 2A^{-2}(z)\,[y_{n-i-j}] \\
\partial^2_{a_i b_j} y_n &= -A^{-2}(z)\,[x_{n-i-j}] \\
\partial^2_{b_i b_j} y_n &= 0.
\end{aligned} \tag{73}$$

Due to the shift-invariance property of the derivative operator, derivatives of y_n with respect to different coefficients a_i and b_i can be expressed as a time-shifted version of derivatives of y_n with respect to a_0 and b_0:

$$\begin{aligned}
\partial_{a_i} y_n &= \partial_{a_0} y_{n-i} \\
\partial_{b_i} y_n &= \partial_{b_0} y_{n-i} \\
\partial^2_{a_i a_j} y_n &= \partial^2_{a_0} y_{n-i-j} \\
\partial^2_{a_i b_j} y_n &= \partial^2_{a_0 b_0} y_{n-i-j}.
\end{aligned} \tag{74}$$

■

A.3 Proof of Proposition 3 – Gradient and Hessian of f_2

Gradient of f_2

Let us use the abbreviation ϕ_n, ϕ'_n and ϕ''_n to denote $\phi(y_n)$, $\phi'(y_n)$ and $\phi''(y_n)$, respectively. The gradient of f_2 with respect to a and b is given by

$$\nabla_{ab}\, f_2 \;=\; \begin{bmatrix} \nabla_a\, f_2 \\ \nabla_b\, f_2 \end{bmatrix}, \tag{75}$$

where

$$\nabla_a f_2 = \sum_{n=0}^{T-1} \phi'_n \cdot \nabla_a y_n = \begin{bmatrix} \partial_{a_0} y_0 & \partial_{a_0} y_1 & ... & \partial_{a_0} y_{M-1} & ... & \partial_{a_0} y_{T-1} \\ 0 & \partial_{a_0} y_0 & ... & \partial_{a_0} y_{M-2} & ... & \partial_{a_0} y_{T-2} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & ... & \partial_{a_0} y_0 & ... & \partial_{a_0} y_{T-M-1} \end{bmatrix} \cdot \begin{bmatrix} \phi'_0 \\ \phi'_1 \\ \vdots \\ \phi'_{T-1} \end{bmatrix} \tag{76}$$

and

$$\nabla_b f_2 = \sum_{n=0}^{T-1} \phi'_n \cdot \nabla_b y_n = \begin{bmatrix} \partial_{b_0} y_0 & \partial_{b_0} y_1 & ... & \partial_{b_0} y_{N-1} & ... & \partial_{b_0} y_{T-1} \\ 0 & \partial_{b_0} y_0 & ... & \partial_{b_0} y_{N-2} & ... & \partial_{b_0} y_{T-2} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & ... & \partial_{b_0} y_0 & ... & \partial_{b_0} y_{T-N-1} \end{bmatrix} \cdot \begin{bmatrix} \phi'_0 \\ \phi'_1 \\ \vdots \\ \phi'_{T-1} \end{bmatrix}. \tag{77}$$

The two Toeplitz matrices containing partial derivatives of y_n with respect to a_0 and b_0 can be regarded as anti-causal FIR filters reversed in time applied to the sequence $\phi'_n = \begin{bmatrix} \phi'_0 & \phi'_1 & ... & \phi'_{T-1} \end{bmatrix}^T$, or alternatively, as causal FIR filters reversed in time applied to the reversed sequence $\mathcal{J}\phi'_n = \phi'_{T-1-n}$, where $\mathcal{J}$ is the mirror operator. Let us denote the impulse responses of the filters corresponding to equations (76), (77) as

$$\begin{aligned} h_{\partial A}(n) &= \partial_{a_0} y_n = -A^{-1}(z)\,[y_n] \\ h_{\partial B}(n) &= \partial_{b_0} y_n = A^{-1}(z)\,[x_n]\,, \end{aligned} \tag{78}$$

respectively. The gradients of f_2 with respect to a and b can be calculated efficiently as the cropped responses of $h_{\partial A}(n)$ and $h_{\partial B}(n)$ to $\mathcal{J}\phi_n$,

$$\begin{aligned} (\nabla_a f_2)_n &= \mathcal{J}\left(h_{\partial A} * \mathcal{J}\phi'\right)_n \quad : \; n = 0, ..., M-1 \\ (\nabla_b f_2)_n &= \mathcal{J}\left(h_{\partial B} * \mathcal{J}\phi'\right)_n \quad : \; n = 0, ..., N-1. \end{aligned} \tag{79}$$

Hessian of f_2

The Hessian of f_2 with respect to a and b is given by

$$\nabla^2_{ab} f_2 = \begin{bmatrix} H_{aa} & H_{ab} \\ H_{ab}^T & H_{bb} \end{bmatrix}, \tag{80}$$

where H_{aa}, H_{bb} and H_{ab} are, respectively, an $M \times M$, $N \times N$ and $M \times N$ matrix, whose elements are given by

$$\begin{aligned}
(H_{aa})_{ij} &= \left(H^1_{aa}\right)_{ij} + \left(H^2_{aa}\right)_{ij} \\
&= \sum_{n=0}^{T-1} \phi''_n \, \partial_{a_i} y_n \, \partial_{a_j} y_n + \sum_{n=0}^{T-1} \phi'_n \, \partial^2_{a_i a_j} y_n \\
&= \sum_{n=0}^{T-1} \phi''_n \, \partial_{a_0} y_{n-i} \, \partial_{a_0} y_{n-j} + \sum_{n=0}^{T-1} \phi'_n \, \partial^2_{a_0} y_{n-i-j}, \qquad (81) \\
(H_{bb})_{ij} &= \sum_{n=0}^{T-1} \phi''_n \, \partial_{b_0} y_{n-i} \, \partial_{b_0} y_{n-j} + \sum_{n=0}^{T-1} \phi'_n \, \partial^2_{b_0} y_{n-i-j} \\
&= \sum_{n=0}^{T-1} \phi''_n \, \partial_{b_0} y_{n-i} \, \partial_{b_0} y_{n-j} \qquad (82)
\end{aligned}$$

and

$$\begin{aligned}
(H_{ab})_{ij} &= \left(H^1_{ab}\right)_{ij} + \left(H^2_{ab}\right)_{ij} \\
&= \sum_{n=0}^{T-1} \phi''_n \, \partial_{a_0} y_{n-i} \, \partial_{b_0} y_{n-j} + \sum_{n=0}^{T-1} \phi'_n \, \partial^2_{a_0 b_0} y_{n-i-j}. \qquad (83)
\end{aligned}$$

The first term in (81) is an $M \times M$ matrix given by

$$H^1_{aa} = \begin{bmatrix} \partial_{a_0} y_0 & \partial_{a_0} y_1 & ... & \partial_{a_0} y_{M-1} & ... & \partial_{a_0} y_{T-1} \\ 0 & \partial_{a_0} y_0 & ... & \partial_{a_0} y_{M-2} & ... & \partial_{a_0} y_{T-2} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & ... & \partial_{a_0} y_0 & ... & \partial_{a_0} y_{T-M-1} \end{bmatrix} \cdot \begin{bmatrix} \alpha^0_0 & ... & \alpha^{M-1}_0 \\ \vdots & & \vdots \\ \alpha^0_{T-1} & ... & \alpha^{M-1}_{T-1} \end{bmatrix}, \qquad (84)$$

where $\alpha^j_n = \phi''_n \, \partial_{a_0} y_{n-j}$. H^1_{aa} and can be evaluated efficiently by using $h_{\partial A}$, namely

$$\left(H^1_{aa}\right)_{ij} = \mathcal{J}\left(h_{\partial A} * \mathcal{J}\alpha^j\right)_i \quad : \; i,j = 0,...,M-1. \qquad (85)$$

The matrix H_{bb} in (82) consists of a single term, which has a structure similar to H^1_{aa} and can be computed using $h_{\partial B}$, namely

$$(H_{bb})_{ij} = \mathcal{J}\left(h_{\partial B} * \mathcal{J}\beta^j\right)_i \quad : \; i,j = 0,...,N-1, \qquad (86)$$

where $\beta^j_n = \phi''_n \, \partial_{b_0} y_{n-j}$. Applying the kernel $h_{\partial B}$ to the reversed sequence α_n instead of the reversed sequence β_n, one can compute the first mixed derivatives term H^1_{ab}:

$$\left(H^1_{ab}\right)_{ij} = \mathcal{J}\left(h_{\partial B} * \mathcal{J}\alpha^j\right)_i \quad : \; i = 0,...,M-1; j = 0,...,N-1. \qquad (87)$$

The second term H^2_{aa} in (81) can be expressed as a matrix with equal elements along anti-diagonals

$$H^2_{aa} \;=\; \begin{bmatrix} \xi_0 & \xi_1 & \cdots & \xi_{M-1} \\ \xi_1 & \xi_2 & \cdots & \xi_M \\ \vdots & \vdots & & \vdots \\ \xi_{M-1} & \xi_M & \cdots & \xi_{2M-2} \end{bmatrix}, \tag{88}$$

where

$$\xi_m \;=\; \sum_{n=0}^{T-1} \phi'_n \, \partial^2_{a_0} y_{n-m}, \tag{89}$$

which can be efficiently evaluated defining the causal FIR kernel

$$h_{\partial^2 A}(n) \;=\; \partial^2_{a_0} y_n = 2A^{-2}(z)\,[y_n]\,, \tag{90}$$

and applying it to the reversed sequence ϕ'_n:

$$\xi_m \;=\; \mathcal{J}\left(h_{\partial^2 A} * \mathcal{J}\phi'\right)_m \quad : \; m = 0, ..., 2M-2. \tag{91}$$

The second term H^2_{ab} in (83) despite being generally non-square, has a similar anti-diagonal structure

$$H^2_{ab} \;=\; \begin{bmatrix} \eta_0 & \eta_1 & \cdots & \eta_{N-1} \\ \eta_1 & \eta_2 & \cdots & \eta_N \\ \vdots & \vdots & & \vdots \\ \eta_{M-1} & \eta_M & \cdots & \eta_{M+N-2} \end{bmatrix}, \tag{92}$$

where

$$\eta_m \;=\; \sum_{n=0}^{T-1} \phi'_n \, \partial^2_{a_0 b_0} y_{n-m}, \tag{93}$$

which similarly to ξ_m, can be efficiently evaluated defining the causal FIR kernel

$$h_{\partial^2 AB}(n) \;=\; \partial^2_{a_0 b_0} y_n = -A^{-2}(z)\,[x_n] \tag{94}$$

and applying it to the reversed sequence ϕ'_n, namely:

$$\eta_m \;=\; \mathcal{J}\left(h_{\partial^2 AB} * \mathcal{J}\phi'\right)_m \quad : \; m = 0, ..., M+N-2. \tag{95}$$

■

A.4 Proof of Proposition 5 – Gradient and Hessian of f_S

Gradient of f_S

Similarly to $\partial_{a_i} y_n$ from Proposition 2, $\partial_{b_i} q_n$ is given by

$$\partial_{b_i} q_n \;\; = \;\; -B^{-1}(z)\,[q_{n-i}] = -B^{-2}(z)\,[\delta_{n-i}] = h_{\partial Q}(n-i). \tag{96}$$

Hence,

$$\nabla_b\, f_S = \sum_{n=0}^{N_S-1} \phi'_n \cdot \nabla_b\, q_n = \\ \left[\begin{array}{cccccc} \partial_{b_0} q_0 & \partial_{b_0} q_1 & \ldots & \partial_{b_0} q_{N-1} & \ldots & \partial_{b_0} q_{N_S-1} \\ 0 & \partial_{b_0} q_0 & \ldots & \partial_{b_0} y_{N-2} & \ldots & \partial_{b_0} y_{N_S-2} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & \partial_{b_0} q_0 & \ldots & \partial_{b_0} q_{N_S-N-1} \end{array}\right] \cdot \left[\begin{array}{c} \phi'_0 \\ \phi'_1 \\ \vdots \\ \phi'_{N_S-1} \end{array}\right], \tag{97}$$

where $\phi'_n = \phi'_{\lambda_S}(q_n)$. Similarly to the gradient of f_2, the above matrix product can be computed using an FIR kernel, namely,

$$(\nabla_b\, f_S)_n \;\; = \;\; \mathcal{J}\left(h_{\partial Q} * \mathcal{J}\phi'\right)_n \quad : \; n = 0, ..., N-1. \tag{98}$$

Since $f_S(b)$ is independent on a, $\nabla_a f_S = 0$.

Hessian of f_S

Again, similarly to $\partial^2_{a_i a_j} y_n$ from Proposition 2, $\partial^2_{b_i b_j} q_n$ is given by

$$\partial^2_{b_i b_j} q_n \;\; = \;\; 2B^{-2}(z)\,[q_{n-i-j}] = 2B^{-3}(z)\,[\delta_{n-i-j}] = h_{\partial^2 Q}(n-i-j). \tag{99}$$

The only non-zero block in the Hessian of f_S is H_{bb}, which consists of two terms

$$\begin{aligned} (H_{bb})_{ij} \;\; &= \;\; \left(H^1_{bb}\right)_{ij} + \left(H^2_{bb}\right)_{ij} \\ &= \;\; \sum_{n=0}^{N_S-1} \phi''_n\, \partial_{b_0} q_{n-i}\, \partial_{b_0} q_{n-j} + \sum_{n=0}^{N_S-1} \phi'_n\, \partial^2_{b_0} q_{n-i-j}, \end{aligned} \tag{100}$$

where $\phi''_n = \phi''_{\lambda_S}(q_n)$.

Computation of H_{bb} in (100) is analogous to computation of H_{aa} in (81). The first term H^1_{bb} is an $N \times N$ matrix given by

$$H^1_{bb} \;\; = \;\; \left[\begin{array}{cccccc} \partial_{b_0} q_0 & \partial_{b_0} q_1 & \ldots & \partial_{b_0} q_{N-1} & \ldots & \partial_{b_0} q_{N_S-1} \\ 0 & \partial_{b_0} q_0 & \ldots & \partial_{b_0} q_{N-2} & \ldots & \partial_{b_0} q_{N_S-2} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & \partial_{b_0} q_0 & \ldots & \partial_{b_0} q_{N_S-N-1} \end{array}\right] \cdot \left[\begin{array}{ccc} \gamma^0_0 & \ldots & \gamma^{N-1}_0 \\ & & \\ \vdots & & \vdots \\ & & \\ \gamma^0_{N_S-1} & \ldots & \gamma^{N-1}_{N_S-1} \end{array}\right], \tag{101}$$

where $\gamma_n^j = \phi_n'' \, \partial_{b_0} q_{n-j}$. H_{bb}^1 and can be evaluated efficiently by using $h_{\partial Q}$, namely

$$\left(H_{bb}^1\right)_{ij} \;=\; \mathcal{J}\left(h_{\partial Q} * \mathcal{J}\gamma^j\right)_i \quad : \; i,j = 0,...,N-1. \tag{102}$$

The second term H_{bb}^2 in (100) can be expressed as a matrix with equal elements along anti-diagonals

$$H_{bb}^2 \;=\; \begin{bmatrix} \zeta_0 & \zeta_1 & \cdots & \zeta_{N-1} \\ \zeta_1 & \zeta_2 & \cdots & \zeta_N \\ \vdots & \vdots & & \vdots \\ \zeta_{N-1} & \zeta_N & \cdots & \zeta_{2N-2} \end{bmatrix}, \tag{103}$$

where

$$\zeta_m \;=\; \sum_{n=0}^{N_S-1} \phi_n' \, \partial_{b_0}^2 q_{n-m}, \tag{104}$$

which can be efficiently evaluated applying the causal FIR kernel $h_{\partial^2 Q}$ to the reversed sequence ϕ_n':

$$\zeta_m \;=\; \mathcal{J}\left(h_{\partial^2 Q} * \mathcal{J}\phi'\right)_m \quad : \; m = 0,...,2N-2. \tag{105}$$

■

A.5 Proof of Proposition 6 – Approximate Hessian

Approximate Hessian of f_1

In Section 2.2 we have shown that f_1 was a discrete approximation of the integral

$$f_1 \;\approx\; \frac{N_F}{\pi} \int_{-\pi}^{\pi} \left[\log\left|B(e^{i\theta})\right| - \log\left|A(e^{i\theta})\right|\right] \, d\theta. \tag{106}$$

Here we will use the continuous version of f_1 to approximate the structure of $\nabla^2 f$ for $a = b = \delta_n$. Let us define

$$q \;=\; \int_{-\pi}^{\pi} \log B(e^{i\theta}) \, d\theta = \int_{-\pi}^{\pi} \log\left(b_0 + b_1 \, e^{-i\theta} + ... + b_{N-1} \, e^{-i(N-1)\theta}\right) \, d\theta \tag{107}$$

and find $\partial_{b_m b_n}^2 q$:

$$\begin{aligned} \partial_{b_m b_n}^2 q \;&=\; \int_{-\pi}^{\pi} \partial_{b_m b_n}^2 \log B(e^{i\theta}) \, d\theta \\ &=\; -\int_{-\pi}^{\pi} \frac{\partial_{b_m} B(e^{i\theta}) \, \partial_{b_n} B(e^{i\theta})}{B^2(e^{i\theta})} \, d\theta + \int_{-\pi}^{\pi} \frac{\partial_{b_m b_n} B(e^{i\theta})}{B(e^{i\theta})} \, d\theta. \end{aligned} \tag{108}$$

Substituting $\partial_{b_n} B(e^{i\theta}) = e^{-in\theta}$ and $\partial^2_{b_n b_m} B(e^{i\theta}) = 0$, one has

$$\partial^2_{b_m b_n} q \;\; = \;\; -\int_{-\pi}^{\pi} \frac{e^{-i(m+n)\theta}}{B^2(e^{i\theta})}\, d\theta. \tag{109}$$

Substituting $b_0 = 1, b_1 = b_2 = ... = b_{N-1} = 0$ into (109) we obtain

$$\partial^2_{b_m b_n} q\big|_{b_0=1,\; b_1=b_2=...=b_{N-1}=0} \;\; = \;\; -\int_{-\pi}^{\pi} e^{-i(m+n)\theta}\, d\theta. \tag{110}$$

Using this result, one can find the approximate second-order derivatives of f_1 with respect to b_n for $b_n = \delta_n$:

$$\begin{aligned}
\partial^2_{b_m b_n} f_1 \;\; &\approx \;\; \frac{N_F}{\pi} \int_{-\pi}^{\pi} \partial^2_{b_m b_n} \left[\log \left| B(e^{i\theta}) \right| - \log \left| A(e^{i\theta}) \right| \right] \, d\theta \\
&= \;\; \frac{N_F}{2\pi} \int_{-\pi}^{\pi} \partial^2_{b_m b_n} \log \left| B(e^{i\theta}) \right|^2 \, d\theta \\
&= \;\; \frac{N_F}{2\pi} \int_{-\pi}^{\pi} \partial^2_{b_m b_n} \log B(e^{i\theta})\, d\theta + \frac{N_F}{2\pi} \left[\int_{-\pi}^{\pi} \partial^2_{b_m b_n} \log B(e^{i\theta})\, d\theta \right]^* \\
&= \;\; -\frac{N_F}{2\pi} \int_{-\pi}^{\pi} \left[e^{-i(m+n)\theta} + e^{i(m+n)\theta} \right] \, d\theta \\
&= \;\; -2N_F \, \frac{\sin(m+n)\pi}{(m+n)\pi} = -2N_F \, \delta_{m+n}.
\end{aligned} \tag{111}$$

Similarly, one has for $a_n = \delta_n$,

$$\partial^2_{a_m a_n} f_1 \;\; \approx \;\; 2N_F \, \delta_{m+n}. \tag{112}$$

The Hessian of f_1 at the point $a_n = b_n = \delta_n$ can be therefore approximated by

$$\nabla^2 f_1 \;\; \approx \;\; 2N_F \left[\begin{array}{cc} E_M & \\ & -E_N \end{array} \right], \tag{113}$$

where E_k denotes a $k \times k$ matrix with the first diagonal element equal to one and the rest of elements equal to zero.

Approximate Hessian of f_2

In Section A.3 we have seen that the Hessian of f_2 is composed of three blocks, namely, H_{aa}, H_{ab} and H_{bb}, whose sizes are $M \times M$, $M \times N$ and $N \times N$, respectively. For $a_n = b_n = \delta_n$ and $x \approx s$, one has $y \approx s$. Therefore, the first- and the second-order derivatives of y_n from

Proposition 2 can be approximated by

$$\begin{aligned}
\partial_{a_0} y_n &\approx -s_n \\
\partial_{b_0} y_n &\approx s_n \\
\partial^2_{a_0} y_n &\approx 2s_n \\
\partial^2_{a_0 b_0} y_n &\approx -s_n,
\end{aligned} \tag{114}$$

yielding

$$\begin{aligned}
(H_{aa})_{ij} &\approx \sum_{n=0}^{T-1} \phi''(s_n)\, s_{n-i}\, s_{n-j} + 2\sum_{n=0}^{T-1} \phi'(s_n)\, s_{n-i-j} \\
(H_{bb})_{ij} &\approx \sum_{n=0}^{T-1} \phi''(s_n)\, s_{n-i}\, s_{n-j} \\
(H_{ab})_{ij} &\approx -\sum_{n=0}^{T-1} \phi''(s_n)\, s_{n-i}\, s_{n-j} - \sum_{n=0}^{T-1} \phi'(s_n)\, s_{n-i-j}.
\end{aligned} \tag{115}$$

As the sample size T grows, the sums approach the corresponding expectation values

$$\begin{aligned}
(H_{aa})_{ij} &\approx T \cdot \mathbb{E}\left\{\phi''(s_n)\, s_{n-i}\, s_{n-j} + 2\phi'(s_n)\, s_{n-i-j}\right\} \\
(H_{bb})_{ij} &\approx T \cdot \mathbb{E}\left\{\phi''(s_n)\, s_{n-i}\, s_{n-j}\right\} \\
(H_{ab})_{ij} &\approx -T \cdot \mathbb{E}\left\{\phi''(s_n)\, s_{n-i}\, s_{n-j} + \phi'(s_n)\, s_{n-i-j}\right\}.
\end{aligned} \tag{116}$$

Since s_n are zero-mean i.i.d., the expectation value $\mathbb{E}\left\{\phi''(s_n)\, s_{n-i}\, s_{n-j}\right\}$ is equal to zero for $i \neq j$. Similarly, $\mathbb{E}\left\{\phi'(s_n)\, s_{n-i-j}\right\} = 0$ for $i \neq -j$ and since i and j are non-negative, this expectation value is non-zero only for $i = j = 0$. Therefore, the blocks (H_{aa}) and (H_{bb}) are approximately diagonal, and the block (H_{ab}), although being usually non-square, has also approximately zero entries outside the main diagonal.

Particularly, for small values of λ, when $\phi'(\cdot)$ and $\phi''(\cdot)$ approach $\mathrm{sign}(\cdot)$ and $\frac{1}{2\lambda}\delta(\cdot)$, respectively, the following form is obtained:

$$(H_{aa})_{ij} \approx \begin{cases} 2c_1 & : \quad i = j = 0, \\ c_2 & : \quad i = j \neq 0, \\ 0 & : \quad \text{otherwise.} \end{cases} \qquad (H_{ab})_{ij} \approx \begin{cases} -c_1 & : \quad i = j = 0, \\ -c_2 & : \quad i = j \neq 0, \\ 0 & : \quad \text{otherwise.} \end{cases}$$

$$(H_{bb})_{ij} \approx \begin{cases} c_2 & : \quad i = j \neq 0, \\ 0 & : \quad \text{otherwise.} \end{cases}$$

where $c_1 = T \cdot \mathbb{E}\,|s_n|$ and $c_2 = \frac{T}{2\lambda} \cdot \sigma_s^2\, p_s(0)$ are constants.

Approximate Hessian of f_S

In Section A.4 we have seen that the Hessian of f_S had only one non-zero block of size $N \times N$, corresponding to derivatives with respect to b. This block was given by

$$(H_{bb})_{ij} = \sum_{n=0}^{N_S-1} \phi''_n \, \partial_{b_0} q_{n-i} \, \partial_{b_0} q_{n-j} + \sum_{n=0}^{N_S-1} \phi'_n \, \partial^2_{b_0} q_{n-i-j}, \tag{117}$$

where q_n is obtained by applying $B^{-1}(z)$ to the sequence δ_n. When $b_n = \delta_n$, one has $q_n = \partial_{b_0} q_n = \partial^2_{b_0} q_n = \delta_n$ and consequently

$$\begin{aligned}(H_{bb})_{ij} &= \phi''_{\lambda_S}(\delta_n)\, \delta_{n-i}\, \delta_{n-j} + \phi'_{\lambda_S}(\delta_n)\, \delta_{n-i-j} \\ &= \begin{cases} \phi''_{\lambda_S}(1) + \phi'_{\lambda_S}(1) & : \quad i = j = 0 \\ \phi''_{\lambda_S}(0) + \phi'_{\lambda_S}(0) & : \quad i = j \neq 0 \\ \phi'_{\lambda_S}(0) & : \quad i \neq j \end{cases}\end{aligned} \tag{118}$$

Substituting

$$\begin{aligned} \phi'_{\lambda_S}(0) &= 0 \qquad \phi'_{\lambda_S}(1) = \tfrac{\lambda_S^{-1}}{1+\lambda_S^{-1}} \\ \phi''_{\lambda_S}(0) &= 1 \qquad \phi''_{\lambda_S}(1) = \left(1+\lambda_S^{-1}\right)^{-2}, \end{aligned}$$

one has

$$(H_{bb})_{ij} = \begin{cases} \frac{\lambda_S^{-1}}{1+\lambda_S^{-1}} + \left(1+\lambda_S^{-1}\right)^{-2} & : \quad i = j = 0 \\ 1 & : \quad i = j \neq 0 \\ 0 & : \quad i \neq j. \end{cases} \tag{119}$$

Particularly, for $\lambda_S = 1$,

$$(H_{bb})_{ij} = \begin{cases} 1.25 & : \quad i = j = 0 \\ 1 & : \quad i = j \neq 0 \\ 0 & : \quad i \neq j. \end{cases} \tag{120}$$

The Hessian of f_S for $b = \delta_n$ is therefore given by the following diagonal matrix:

$$\nabla^2 f_S = \begin{bmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & I_N + c_3 \, E_N \end{bmatrix}, \tag{121}$$

where c_3 is a constant (equal to 0.25 in the particular case of $\lambda_S = 1$).

Combining the above result with the approximations of $\nabla^2 f_1$ and $\nabla^2 f_2$, we conclude that the Hessian of $f(a, b; x)$ for $a = b = \delta_n$ and $x \approx s$ can be approximated by a tri-diagonal matrix with non-zero elements on the main diagonal and the diagonals $\pm M$. ■

Appendix B Probability density functions

B.1 Gauss-Bernoully distribution

A random variable s is said to obey the Gauss-Bernoully distribution with sparsity ρ and variance σ^2 if its probability function is given by

$$p(s) \quad = \quad (1-\rho)\delta(s) + \frac{\rho}{\sqrt{2\pi\rho\sigma^2}} \exp\left\{-\frac{s^2}{2\rho\sigma^2}\right\}, \tag{122}$$

where $\rho \in [0,1]$ and $\sigma^2 > 0$. It will be henceforth assumed $\sigma^2 = 1$. Probability density function (PDFs) and cumulative distribution function (CDFs) of the Gauss-Bernoully distribution are depicted in Figure 22.

B.2 Generalized Laplacian distribution

A random variable s is said to obey the generalized Laplacian distribution with parameters α, λ if its probability function is given by

$$p(s) \quad = \quad \frac{1}{2\Gamma(1+\alpha^{-1})\lambda^{\alpha^{-1}}} \exp\left\{-\frac{|s|^\alpha}{\lambda}\right\}, \tag{123}$$

where $\Gamma(z)$ is the Euler Gamma function defined by

$$\Gamma(z) \quad = \quad \int_0^\infty t^{z-1} \exp\{-t\}\, dt, \tag{124}$$

and $\alpha > 0$ and $\lambda > 0$. Distribution (125) can be interpreted as a generalization of the Laplace distribution obtained for $\alpha = 1$, and of the normal distribution obtained for $\alpha = 2$. For $\alpha < 2$, the distribution is super-Gaussian. It will be henceforth assumed $\lambda = 1$ and $\alpha = 0.5$. PDFs and CDFs of the generalized Laplace distribution are depicted in Figure 23.

B.3 Discrete uniform distribution

A random variable s is said to obey the discrete uniform distribution on interval $[-1,1]$ with N levels if its probability function is given by

$$p(s) \quad = \quad \frac{1}{N}\sum_{n=0}^{N-1} \delta\left(\frac{2n}{N-1} - 1 - s\right). \tag{125}$$

In this work, i.i.d. signals having discrete uniform distribution will be termed as *pulse amplitude modulated* or *PAM* signals. PDFs and CDFs of the discrete uniform distribution are depicted in Figure 24.

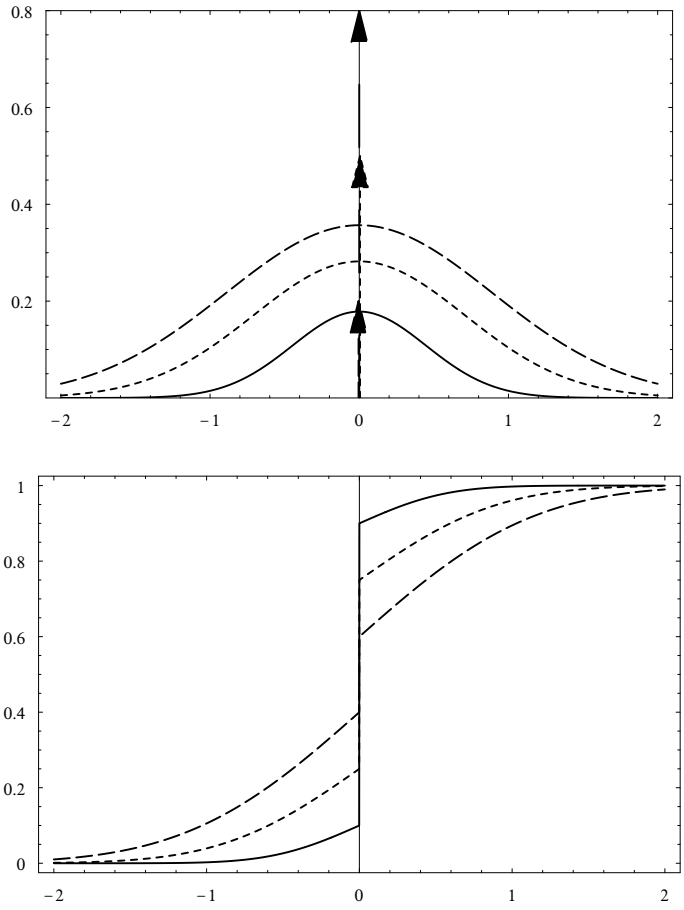

Figure 22: PDF (top) and CDF (bottom) of the Gauss-Bernoully distribution for $\sigma^2 = 1$ and ρ = 0.2 (solid), 0.5 (dotted), and 0.8 (dashed). Vertical arrows denote delta functions.

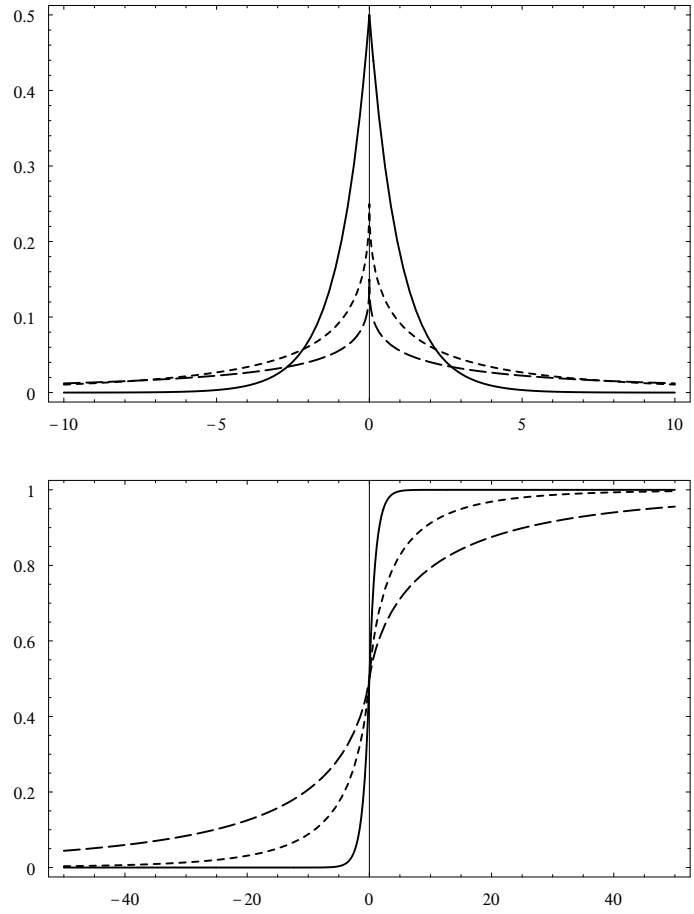

Figure 23: PDF (top) and CDF (bottom) of the generalized Laplace distribution for $\lambda = 1$ and α = 1 (solid), 0.5 (dotted), and 0.4 (dashed).

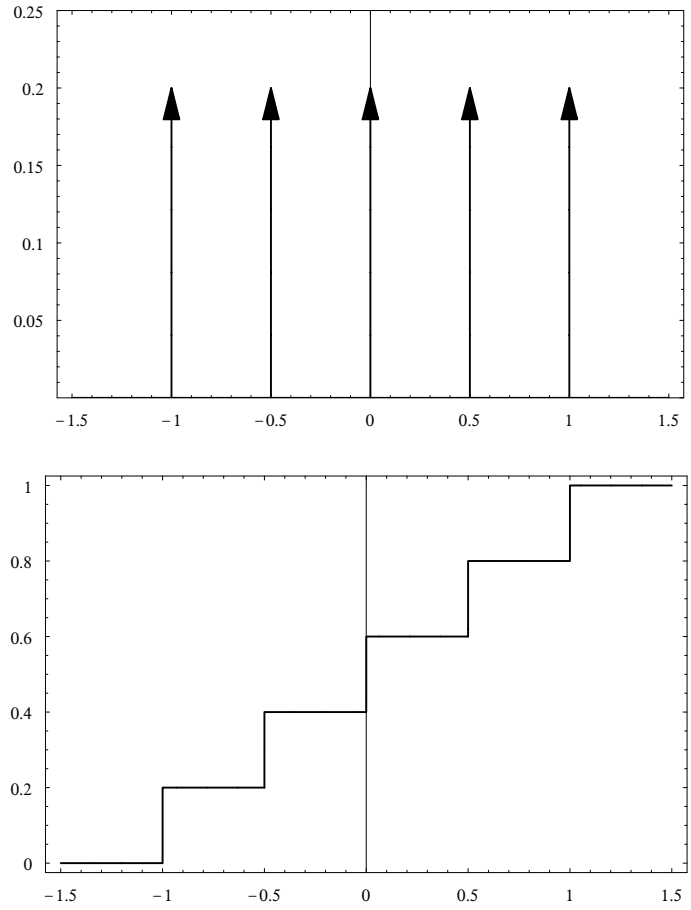

Figure 24: PDF (top) and CDF (bottom) of the discrete uniform distribution for $N = 5$. Vertical arrows denote delta functions.

References

[1] Y. Sato. A method of self-recovering equalization for multilevel amplitude-modulation systems. *IEEE Trans. Computers*, pages 679–682, 1975.

[2] A. Benveniste, M. Goursat, and G. Ruget. Robust identification of a nonminimum phase system: Blind adjustment of a linear equalizer in data communications. *IEEE Trans. Automat. Contr.*, 25(3):385–399, 1980.

[3] D. N. Godard. Self-recovering equalization and carrier tracking in two-dimensional data communication systems. *IEEE Trans. Commun.*, 28(11):1867–1875, 1980.

[4] J. R. Treichler and B. G. Agee. A new approach to the multi-path correction of constant modulus signals. *IEEE Trans. Acoust. Speech Sig. Proc.*, 31(2):331–344, 1983.

[5] S. Bellini. Bussgang techniques for blind equalization. In *Proc. IEEE Global Telecommunication Conf. Rec.*, pages 1634–1640, 1986.

[6] L. Tong, G. Xu, and T. Kailath. A new approach to blind identification and equalization of multipath channel. In *Proc. 25th Asilomar Conf. Signal, Syst., Comput.*, 1991.

[7] G. Xu, H. Liu, L. Tong, and T. Kailath. Least squares approach to blind channel identification. *IEEE Trans. Sig. Proc.*, 43(12):2982–2993, 1995.

[8] E. Moulines, P. Duhamel, J.-F. Cardoso, and S. Mayrargue. Subspace methods for the blind identification of multichannel fir filters. *IEEE Trans. Sig. Proc.*, 43:516–525, 1995.

[9] L. Tong, G. Xu, and T. Kailath. Blind identification and equalization based on secondorder statistics: A time domain approach. *IEEE Trans. Inform. Theory*, 40(2):340–349, 1994.

[10] M.I. Gurelli and C.L. Nikias. EVAM: An eigenvectorbased algorithm for multichannel bind deconvolution of input colored signals. *IEEE Trans. Signal Processing*, 43(1):134–149, 1995.

[11] Y. Hua. Fast maximum likelihood for blind identification of multiple FIR channels. *IEEE Trans. Sig. Proc.*, 44(3):661–672, 1996.

[12] A. Gorokhov, P. Loubaton, and E. Moulines. Second order blind equalization in multiple input multiple output FIR systems: A weighted least squares approach. In *Proc. ICASSP*, volume 5, pages 2415–2418, 1996.

[13] S.-I. Amari, A. Cichocki, and H. H. Yang. Novel online adaptive learning algorithms for blind deconvolution using the natural gradient approach. In *Proc. SYSID*, pages 1057–1062, July 1997.

[14] S.-I. Amari, S. C. Douglas, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. *Advances in Neural Information Processing Systems*, 8:757–763, 1996.

[15] J.-F. Cardoso and B. Laheld. Equivariant adaptive source separation. *IEEE Trans. Sig. Proc.*, 44(12):3017–3030, 1996.

[16] S.C. Douglas. On equivariant adaptation in blind deconvolution. In *Proc. Asilomar Conf. Signals, Syst., Comput.*, November 2002.

[17] S.-I. Amari, S. C. Douglas, A. Cichocki, and H. H. Yang. Multichannel blind deconvolution and equalization using the natural gradient. In *Proc. SPAWC*, pages 101–104, April 1997.

[18] R. H. Lambert. *Multichannel blind deconvolution: FIR Matrix algebra and separation of multipath mixtures*. PhD thesis, University of Southern California, 1996.

[19] M. Joho, H. Mathis, and G. S. Moschytz. An FFT-based algorithm for multichannel blind deconvolution. In *Proc. ISCAS*, volume 3, pages 203–206, May 1999.

[20] M. Joho, H. Mathis, and G. S. Moschytz. On frequency-domain implementations of filtered-gradient blind deconvolution algorithms. In *Proc. Asilomar Conf. Signals, Syst., Comput.*, November 2002.

[21] M. Joho and P. Schniter. Frequency domain realization of a multichannel blind deconvolution algorithm based on the natural gradient. In *Proc. 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, April 2003.

[22] J.K. Tugnait. Blind equalization and channel estimation for multipleinput multipleoutput communications systems. In *Proc. ICASSP*, volume 5, pages 2443–2462, 1996.

[23] S. C. Douglas, A. Cichocki, and S.-I. Amari. Quasi-Newton filtered-error and filtered-regressor algorithms for adaptive equalization and deconvolution. In *Proc. 1st IEEE Workshop on Sig. Proc. App. Wireless Comm.*, 1997.

[24] D. Pham and P. Garrat. Blind separation of a mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Trans. Sig. Proc.*, 45:1712–1725, 1997.

[25] M. Zibulevsky. Sparse source separation with relative Newton method. 2002.

[26] B. Porat. *A Course in Digital Signal Processing*. John Wiley & Sons, Inc., 1997.

[27] E. Moulines, J.-F. Cardoso, and E. Gassiat. Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. 1997.

[28] A.J. Bell and T.J. Sejnowski. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[29] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, 1998.

[30] M. Zibulevsky, B. A. Pearlmutter, P. Bofill, and P. Kisilev. *Independent Components Analysis: Principles and Practice*, chapter Blind source separation by sparse decomposition. Cambridge University Press, 2001.

[31] M. Zibulevsky, P. Kisilev, Y. Y. Zeevi, and B. A. Pearlmutter. *Advances in Neural Information Processing Systems*, volume 12, chapter Blind source separation via multinode sparse representation. MIT Press, 2002.

[32] D. P. Bertsekas. *Nonlinear Programming (2nd edition)*. Athena Scientific, 1999.

[33] L. Mosheyev and M. Zibulevsky. Penalty/barrier multiplier algorithm for semide nite programming. *Optim. Methods Software*, 13(4):235–261, 2000.

[34] L. Grippo and M. Sciandrone. Globally convergent block-coordinate techniques for unconstrained optimization. *Optimization Methods and Software*, 10(4):587–637, 1999.

[35] A. M. Bronstein, M.M. Bronstein, and M. Zibulevsky. Block-coordinate relative Newton method for blind source separation. 2003.

[36] M. Zibulevsky. Smoothing method of multipliers for sum-max problems. 2002.

[37] S. Fiori, A. Uncini, and F. Piazza. Blind deconvolution by modified Bussgang algorithm. In *Proc. ICAS*, 1999.