Series in Signal and Information Processing

Volume 6

Diss. ETH No 13783

A Systematic Approach to Adaptive Algorithms for Multichannel System Identification, Inverse Modeling, and Blind Identification

To Iris, Dina, and Jan
and in memory of Markus

A dissertation submitted to the
Swiss Federal Institute of Technology, Zürich
for the degree of
Doctor of Technical Sciences

Mehr zu hören, als zu reden – solches lehrt uns die Natur:
Sie versah uns mit zwei Ohren, doch mit einer Zunge nur.

Gottfried Keller

presented by

Marcel Joho

dipl. El.-Ing. ETH
born on October 8, 1967
citizen of Bettwil AG

accepted on the recommendation of
Prof. Dr. George S. Moschytz, examiner
Prof. Dr. Scott C. Douglas, co-examiner
Prof. Dr. Hans-Andrea Loeliger, co-examiner

December 2000

Hartung Gorre Konstanz

Acknowledgments

I wish to express my sincere gratitude to my supervisor, Prof. George S. Moschytz, for his confidence in me and my work, and for providing a first-class research environment at the Signal and Information Processing Laboratory (ISI) of the Swiss Federal Institute of Technology (ETH) in Zürich. Working at the ISI has always been a big pleasure for me.

I am very thankful to Prof. H.-A. Loeliger, the current head of the ISI, for accepting to act as the co-referee, and maintaining the wonderful spirit at our lab.

I am very grateful to Prof. Scott C. Douglas for accepting to serve as a competent co-referee for the present doctoral dissertation. His insight and constructive criticism have considerably improved this work.

I am deeply indebted to Werner Hartmann for being my first teacher in Linear Algebra, to Franta Kraus, who taught me in System Identification and Adaptive Control Theory, and to Russ Lambert, who certainly influenced my work by sending me his wonderful thesis.

Many thanks also go to my colleagues at the ISI: Wolfgang Knecht and Pius Estermann, who introduced me to the exciting field of spatial and adaptive filtering, my kind roommate Stefan Oberle, Pascal Vontobel for finding a proof for everything I wanted to prove, Hanspeter Schmid for showing me alternative viewpoints of life, work, and everything else, and also for being my LaTeX hotline. Many thanks also go to Felix Lustenberger, Felix Tarköy, and Markus Hofbauer for many interesting non-technical discussions.

Very special thanks go to my room neighbor and Unix expert Max Dünki, for his exciting discussions about everything, and for providing us with Sun

workstations, which prevented us from never-ending “plug & pray” games! He must have found a global Lyapunov equation for controlling our computer network, since it always showed a stable behavior.

I always had a very close friendship with Andreas Gygi, with whom I enjoyed having very interesting discussions during lunch for many years, and with Sigi Wyrsch, who was drinking away my tasty coffee and hiding his chocolate supply in vain. Very special thanks go to Heinz Mathis. Without our very close research collaboration and his assistance, this work would have never become what it is. I have really appreciated our daily discussions about research, bringing up children, and Jerry Seinfeld.

I wish to thank my parents, for supporting my studies and paving my way. In particular, I wish to express all my love to my wonderful wife Iris and also to our two children Dina and Jan. By seeing them growing up, I have learned many aspects of “supervised” and also “unsupervised learning”.

Marcel Joho

Zürich, December 2000

Abstract

In many situations related to acoustics and data communications we are confronted with multiple signals received from a multipath mixture, e.g., the famous cocktail-party problem. A multipath mixture can be described by a mixing matrix, whose elements are the individual transfer functions between a source and a sensor. The mixing matrix is usually unknown, and so are sometimes also the source signals.

Depending on the application, different parameters are of interest: the mixing matrix for system identification, the inverse mixing matrix for inverse modeling, or the source signals for system equalization. This thesis gives a systematic approach to the aforementioned problems in a multipath mixing environment. To this end, we investigate the multichannel-mixing problem and the single-channel multipath problem separately.

Based on a mean-squared-error (MSE) cost function, several stochastic-gradient update equations, which are related to the least-mean-square (LMS) and the recursive least-squares (RLS) algorithm, are derived for the instantaneous mixing case. Thereby the matrix-inversion lemma has shown to be a very powerful tool to transform an algorithm which estimates the mixing matrix (system identification) into an algorithm which estimates the inverse mixing matrix (inverse modeling).

With the help of circulant matrices, the adaptive algorithms for the multichannel instantaneous mixing case are transformed to cope with the single-channel multipath case. Block processing techniques are used, allowing efficient implementation of the filtering and adaptation in the frequency domain. The Fast Fourier Transform (FFT) plays a crucial role, owing to its close relationship to circulant matrices.

We extend the algorithms to operate as multichannel adaptive filters, using the fact that a multipath mixture is the combination of instantaneous mixing and single-channel multipath convolution.

In addition, we investigate the situation where not only the multipath-mixing system, but also the source signals are unknown. This situation is referred to as blind identification. By exchanging the non-blind error criterion with a blind error criterion, we derive new algorithms for blind identification (blind source separation, single-channel and multichannel blind deconvolution). The same technique provides an alternative derivation of the well-known natural-gradient learning algorithm for blind source separation, revealing new insight.

Throughout the thesis, many simulation examples illustrate the performance behavior of the different adaptive algorithms.

Keywords. Multichannel adaptive signal processing, multichannel adaptive filtering, system identification, inverse modeling, system/channel equalization, blind identification, blind source separation, blind deconvolution, multichannel blind deconvolution, acoustical signal processing, multipath mixture.

Kurzfassung

In der Akustik und in der Datenkommunikation hat man es oft mit echobehafteten und vermischten Signalen zu tun, zum Beispiel mehrere Sprecher in einem halligen Raum oder Mehrwegausbreitung in Mobilfunkkanälen. Ein solches mehrkanaliges Übertragungssystem kann mit einer Mischmatrix beschrieben werden, deren Elemente die Übertragung zwischen den Sendern und den Empfängern beschreiben, zum Beispiel mittels einer Impulsantwort. Diese mehrkanalige Übertragungsmatrix ist normalerweise nicht bekannt. In einigen Anwendungen sind sogar die ausgesendeten Signale (Quellensignale) unbekannt. Abhängig von der Anwendung interessiert man sich für die Schätzung von verschiedenen Parametern: In der Systemidentifikation für die Schätzung der Übertragungsmatrix oder deren Inverse, bei einer Kanalentzerrung für die Schätzung der übertragenen Datensignale.

Die vorliegende Dissertation analysiert die obigen Problemstellungen in einer systematischen Weise. Dazu wird das allgemeine Problem auf zwei unterschiedliche Arten vereinfacht, die zuerst getrennt untersucht werden. Es sind dies eine einfache Signalmischung und ein einfacher Kanal mit Mehrwegausbreitung.

Basierend auf einem quadratischen Fehlerkriterium leiten wir verschiedene stochastische Gradientenmethoden her, um eine unbekannte Mischmatrix zu schätzen. Diese Methoden weisen eine grosse Verwandtschaft mit dem LMS- (*least-mean-square*) und dem RLS- (*recursive-least-squares*) Algorithmus auf. Das Matrix-Inversions-Lemma hat sich dabei als sehr nützliches Hilfsmittel erwiesen, um einen Schätzalgorithmus für die Mischmatrix in einen effizienten Schätzalgorithmus für die inverse Mischmatrix umzuwandeln.

Adaptive Filteralgorithmen für die einkanalige Kanalschätzung und Kana-

legalisation werden hergeleitet. Es wird eine blockweise Verarbeitung der Eingangsdaten verwendet, welche eine effiziente Implementation der Filterung und Adaption im Frequenzbereich erlaubt. Dabei wird die enge Verwandtschaft zwischen der schnellen Fourier Transformation (FFT) und zirkulären Matrizen ausgenutzt.

Die Algorithmen werden für eine mehrkanalige adaptive Filterung erweitert, indem die Methoden für die mehrkanalige Mischung mit denjenigen der einkanaligen Filterung vereinigt werden.

Zusätzlich wird der Fall der blinden Systemidentifikation untersucht, bei der weder die Übertragungsmatrix noch die Quellensignale bekannt sind. Durch Auswechseln des Fehlerkriteriums lassen sich Algorithmen für die mehrkanalige adaptive Filterung in solche umwandeln, die sich für die blinde Quellenseparation und Rückfaltung eignen. Mit dem selben Vorgehen lässt sich der Algorithmus des natürlichen Gradienten, der in der blinden Quellenseparation weit verbreitet ist, auf eine neue Weise herleiten.

Das Adaptionsverhalten der verschiedenen Algorithmen wird mittels Simulationsbeispielen aufgezeigt.

Stichworte. Mehrkanalige adaptive Signalverarbeitung, Systemidentifikation, Kanalentzerrung, blinde Quellenseparation, blinde Kanalentzerrung, Signalverarbeitung von akustischen Signalen, Mehrwegausbreitung.

Contents

Chapter 1

Introduction

1.1 Preface

Multichannel signal processing is a challenging field in data communications, acoustics, geophysics, biomedical applications, data fusion, fault-detecting systems, and many other application areas. Since the early sixties, when the Kalman filter was invented and Widrow *et al*. introduced the least-mean-square (LMS) algorithm, a tremendous growth of applications using adaptive signal processing in various fields was observed. One of the first to apply the LMS algorithm in data communication for channel equalization was Lucky [76].

Later on, blind algorithms, which do not have access to any reference signal, came up in data communications for channel equalization [11, 12, 41, 91]. Blind deconvolution techniques were also used in geophysical applications [26, 42, 110]. In geophysics the term *blind deconvolution* is more common, as the interest mainly lies in obtaining a model of the system, whereas in data communications the term *blind equalization* is more commonly used, as the main interest lies in retrieving the data.

Algorithms for *blind source separation* or *independent component analysis* (ICA) [21] came later. Jutten, Hérault, and Comon [22, 64] were among the first to describe the problem. Early work was also done by Shalvi and Weinstein [95] and Weinstein *et al.* in [112]. Later on, Bell and Sejnowski came up

with the *Infomax* algorithm [7]. The introduction of the *natural gradient* by Amari *et al.* [4] or the *relative gradient* by Cardoso and Laheld [17] provided a new class of algorithms, which have the so-called *equivariant property*, i.e., the convergence rate is independent of the conditioning of the unknown mixing system. The systematic extension of most known blind source separation algorithms to their *multichannel blind deconvolution* counterpart was done by Lambert [69] using FIR-matrix algebra. Douglas and Haykin showed in [33, 34] the structural relationship between blind deconvolution and blind source separation under the circulant mixing condition, which is also a subject of this thesis.

In the last few years, blind algorithms have attracted many researchers in the field of adaptive signal processing, neural networks, and higher-order statistics.

Applications in acoustics Main applications in acoustics are: Man-machine interface [66], acoustic noise canceler, adaptive microphone arrays, echo cancellation in hands-free telephone and hearing aids, multichannel echo cancellation and speaker separation in teleconferencing, active noise control, dereverberation of acoustical signals, beam steering of loudspeaker arrays, head related transfer functions (HRTF), crosstalk cancellation, removal of multipath in sonar systems, and many others.

Applications in data communications Main applications in data communications are: Smart antennas or adaptive beamforming, single- and multichannel equalization, multi-user separation (e.g. CDMA), adaptive line enhancement, etc.

Further reading Textbooks which cover many aspects of single-channel and multichannel adaptive signal processing are [20, 44, 51, 58]. An overview of the field of *blind* or *unsupervised learning* can be found in [3, 16, 52, 53, 72, 83, 90, 106].

1.2 Problem formulation

1.2.1 Description of the unknown system

In the following, we restrict ourselves to linear systems and assume that the unknown multiple-input multiple-output (MIMO) system can be described by a matrix $\mathbf{A}(z)$, whose elements $a_{ij}(z)$ contain the impulse response between the jth input and the ith output in the two-sided z-domain. M_s denotes the number of system inputs and M the number of system outputs. Thus, the $M \times M_\mathrm{s}$ transfer matrix of the system is defined as

$$\mathbf{A}(z) = \sum_{n=-\infty}^{\infty} \mathbf{A}_n z^{-n} = [a_{ij}(z)] \tag{1.1}$$

$$a_{ij}(z) = \sum_{n=-\infty}^{\infty} a_{ij,n} z^{-n} \qquad \begin{matrix} i = 1, \ldots, M \\ j = 1, \ldots, M_\mathrm{s}\,. \end{matrix} \tag{1.2}$$

We assume that each system is stable, i.e. $\sum_{n=-\infty}^{\infty} |a_{ij,n}| < \infty$. The left hand side of (1.1) represents a *polynomial matrix* or a *Laurent-series matrix* (a matrix whose elements are polynomials, power series, or *Laurent series*) and the right hand side is referred to as a *matrix polynomial* or a *matrix Laurent series* (a polynomial or a Laurent series whose coefficients are matrices) [65]. *Polynomial vectors* and *vector polynomials* are defined accordingly. In fact, the formulation in (1.1) describes a non-causal system. However, we will treat a causal system as a special case of a non-causal system, i.e., $\mathbf{A}(z) = \sum_{n=0}^{\infty} \mathbf{A}_n z^{-n}$.

Aside from the non causality and the infinite extent of the impulse responses, the description (1.1) is common in the field of acoustics and communications. In other applications which are related to control theory, usually a state-space model is preferred, especially if a physical model based on differential equations is available [75].

1.2.2 Environment

The environment, which the system $\mathbf{A}(z)$ is embedded in, is shown in Fig. 1.1. The general *convolutive-mixing system* with additive noise is described in the z-domain as

$$\mathbf{x}(z) = \mathbf{A}(z)\mathbf{s}(z) + \mathbf{n}(z)\,. \tag{1.3}$$

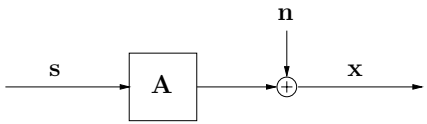

Figure 1.1: Setup of the mixing system with additive noise.

We have M_s source signals, whose time series are represented by their two-sided z-transforms in vector form

$$\mathbf{s}(z) = \sum_{t=-\infty}^{\infty} \mathbf{s}_t z^{-t} = [s_m(z)] \tag{1.4}$$

$$s_m(z) = \sum_{t=-\infty}^{\infty} s_{m,t} z^{-t} \qquad m = 1, \ldots, M_\mathrm{s}\,. \tag{1.5}$$

Likewise, the time series of the M sensor signals are represented as

$$\mathbf{x}(z) = \sum_{t=-\infty}^{\infty} \mathbf{x}_t z^{-t} = [x_m(z)] \tag{1.6}$$

$$x_m(z) = \sum_{t=-\infty}^{\infty} x_{m,t} z^{-t} \qquad m = 1, \ldots, M \tag{1.7}$$

and the time series of the sensor noise as

$$\mathbf{n}(z) = \sum_{t=-\infty}^{\infty} \mathbf{n}_t z^{-t} = [n_m(z)] \tag{1.8}$$

$$n_m(z) = \sum_{t=-\infty}^{\infty} n_{m,t} z^{-t} \qquad m = 1, \ldots, M\,. \tag{1.9}$$

Alternative description Equivalent ways to describe the noisy mixing process are either by a convolutional sum

$$\mathbf{x}_t = (\mathbf{A} * \mathbf{s})_t + \mathbf{n}_t \tag{1.10}$$

$$= \sum_{k=-\infty}^{\infty} \mathbf{A}_{t-k}\, \mathbf{s}_k + \mathbf{n}_t \tag{1.11}$$

or with help of the delay operator q^{-1}, e.g., $q^{-d} x_t = x_{t-d}$ [44, 75],

$$\mathbf{x}_t = \mathbf{A}(q)\, \mathbf{s}_t + \mathbf{n}_t\,. \tag{1.12}$$

Assumptions In all cases, we assume that we have full access to the sensor signals. The sensor noise is always unknown. Furthermore, in a system identification and inverse-modeling setup, we also have access to the source signals. However, in a blind system-identification setup, we only know some statistical properties of the source signals, e.g., non-Gaussianity, but not the source signals themselves, therefore the terminology *blind*. Furthermore, $\mathbf{A}(z)$ is unknown.

Problem formulation Throughout this thesis, we aim at finding an estimate $\hat{\mathbf{A}}(z)$ or $\hat{\mathbf{A}}^{-1}(z)$ of the unknown system $\mathbf{A}(z)$. We have access to the time samples of sensors $\mathbf{x}_t$ and, in the non-blind case, also to the time samples of the sources $\mathbf{s}_t$.

1.2.3 Special cases

Depending on $\mathbf{A}(z)$, we can subdivide the general convolutive-mixing system into several special cases:

1. Instantaneous mixing system

$$\mathbf{A}(z) = \mathbf{A}\,. \tag{1.13}$$

2. Delayed instantaneous mixing system

$$\mathbf{A}(z) = z^{-d}\mathbf{A}\,. \tag{1.14}$$

3. Individually delayed source signals with instantaneous mixing

$$\mathbf{A}(z) = \mathbf{A}\mathbf{D}(z) \tag{1.15}$$

$$\mathbf{D}(z) = \left[z^{-d_{ij}}\right] \tag{1.16}$$

$$\text{or} \qquad \mathbf{D}(z) = \mathrm{diag}\left[z^{-d_1}, \ldots, z^{-d_{M_\mathrm{s}}}\right]\,. \tag{1.17}$$

4. Instantaneous mixing with individually delayed sensor signals

$$\mathbf{A}(z) = \mathbf{D}(z)\mathbf{A}\,. \tag{1.18}$$

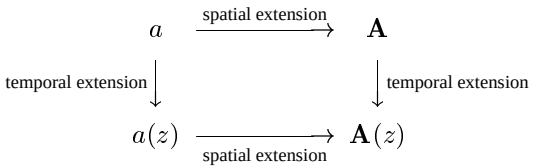

Figure 1.2: Commutative diagram which reveals the relationship between convolution and mixing: a attenuation, $a(z)$ convolution, $\mathbf{A}$ instantaneous mixing, and $\mathbf{A}(z)$ convolutive mixing.

5. Delay-and-sum system[1]

$$\mathbf{A}(z) = \left[z^{-d_{ij}}\right] \mathbf{D} \tag{1.19}$$

with
$$\mathbf{D} = \operatorname{diag}\left[d_1, \ldots, d_{M_\mathrm{s}}\right]. \tag{1.20}$$

6. Single-channel convolution

$$\mathbf{A}(z) = a(z). \tag{1.21}$$

7. Signal attenuation

$$\mathbf{A}(z) = a. \tag{1.22}$$

Fig. 1.2 illustrates the relationship between single-channel and multichannel convolution on the one side, and instantaneous and convolutive mixing on the other side.

Furthermore, depending on the dimension of $\mathbf{A}(z)^{M \times M_\mathrm{s}}$, we distinguish between the following cases for inverse modeling and blind identification:

- *fully determined system* ($M = M_\mathrm{s}$): We have an equal number of sources and sensors and $\det \mathbf{A}(z)$ has no roots on the unit circle. This means that $\mathbf{A}\left(e^{j\omega}\right)$ is of full rank for $-\pi < \omega \leq \pi$.
- *overdetermined system* ($M > M_\mathrm{s}$): More sensors than sources.
- *underdetermined system* ($M < M_\mathrm{s}$): Fewer sensors than sources.

[1] This model is often used in beamforming applications, if the sensors have equal gain and the sources are located in the far field. The jth column of $\mathbf{A}(z)$ is just the *steering vector* for the jth source.

1.3 System identification

In *system identification* we wish to directly find an estimate $\mathbf{H}(z) = \hat{\mathbf{A}}(z)$ of the unknown system

$$\mathbf{H}(z) = \sum_{n=-\infty}^{\infty} \mathbf{H}_n z^{-n} = [h_{ij}(z)] \tag{1.23}$$

$$h_{ij}(z) = \sum_{n=-\infty}^{\infty} h_{ij,n} z^{-n} \qquad \begin{matrix} i = 1, \ldots, M \\ j = 1, \ldots, M_\mathrm{s} \end{matrix} \tag{1.24}$$

such that

$$\hat{\mathbf{x}}(z) = \mathbf{H}(z)\, \mathbf{s}(z) \tag{1.25}$$

becomes an estimate of $\mathbf{x}(z)$. The estimation or prediction error is then

$$\mathbf{e}_\mathrm{x}(z) \triangleq \mathbf{x}(z) - \hat{\mathbf{x}}(z) \tag{1.26}$$

$$= \left[\mathbf{A}(z) - \mathbf{H}(z)\right] \mathbf{s}(z) + \mathbf{n}(z). \tag{1.27}$$

We aim at finding a $\mathbf{H}(z)$ such as to minimize

$$\|\mathbf{e}_\mathrm{x}(z)\|_\mathcal{F}^2 = \|\left[\mathbf{A}(z) - \mathbf{H}(z)\right] \mathbf{s}(z) + \mathbf{n}(z)\|_\mathcal{F}^2 \tag{1.28}$$

$$= \|\left[\mathbf{A}(z) - \mathbf{H}(z)\right] \mathbf{s}(z)\|_\mathcal{F}^2 + \|\mathbf{n}(z)\|_\mathcal{F}^2 \tag{1.29}$$

$$= M_\mathrm{s}\, \sigma_\mathrm{s}^2\, \|\mathbf{A}(z) - \mathbf{H}(z)\|_\mathcal{F}^2 + M\, \sigma_\mathrm{n}^2. \tag{1.30}$$

In these steps, we have assumed that the sensor noise and the source signals are mutually uncorrelated, i.e. $\langle \mathbf{s}(z), \mathbf{n}(z) \rangle_\mathcal{F} = 0$, that all source signals have equal power σ_s^2, and that all noise signals have equal power σ_n^2. The inner product $\langle ., . \rangle_\mathcal{F}$ and the norm $\|.\|_\mathcal{F}$ are defined in Section C.2 and Section C.3, respectively. As seen from (1.30), minimizing $\|\mathbf{e}_\mathrm{x}(z)\|_\mathcal{F}^2$ is equal to minimizing $\|\mathbf{A}(z) - \mathbf{H}(z)\|_\mathcal{F}^2$, as we have no influence on σ_n^2. Therefore, in an iterative or adaptive algorithm, we use the error signal $\mathbf{e}_{\mathrm{x}t}$ to adapt $\mathbf{H}(z)$, as depicted in Fig. 1.3.

1.4 Inverse modeling

In *inverse modeling* we wish to find an estimate of the inverse system, $\mathbf{W}(z) = \hat{\mathbf{A}}^{-1}(z)$, or find an estimate of the pseudoinverse, defined in Section A.8, of

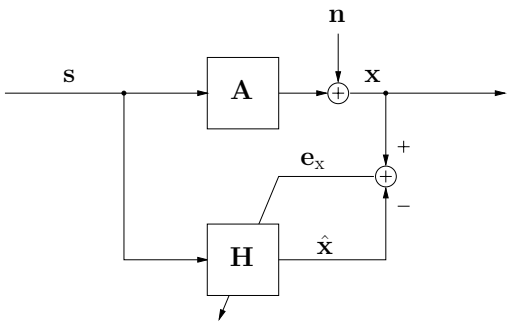

Figure 1.3: System identification. The prediction error $\mathbf{e}_{\mathrm{x}}$ is used for the adaptation.

the unknown system, i.e. $\mathbf{W}(z) = \hat{\mathbf{A}}^{\#}(z)$

$$\mathbf{W}(z) = \sum_{n=-\infty}^{\infty} \mathbf{W}_n z^{-n} = [w_{ij}(z)] \tag{1.31}$$

$$w_{ij}(z) = \sum_{n=-\infty}^{\infty} w_{ij,n} z^{-n} \qquad i,j = 1,\ldots,M \tag{1.32}$$

such that the *global system* matrix

$$\mathbf{G}(z) \triangleq \mathbf{W}(z)\mathbf{A}(z) \tag{1.33}$$

becomes close to the unity matrix $\mathbf{I}$ and

$$\mathbf{u}(z) = \mathbf{W}(z)\mathbf{A}(z)\mathbf{s}(z) = \mathbf{G}(z)\mathbf{s}(z) \tag{1.34}$$

becomes an estimate of $\mathbf{s}(z)$. Thus, we build the *equalization error* as

$$\begin{aligned} \mathbf{e}_{\mathrm{s}}(z) &\triangleq \mathbf{s}(z) - \mathbf{u}(z) && (1.35)\\ &= [\mathbf{I} - \mathbf{W}(z)\mathbf{A}(z)]\,\mathbf{s}(z) - \mathbf{W}(z)\mathbf{n}(z) && (1.36)\\ &= [\mathbf{I} - \mathbf{G}(z)]\,\mathbf{s}(z) - \mathbf{W}(z)\mathbf{n}(z) && (1.37) \end{aligned}$$

which is taken for the adaptation of $\mathbf{W}(z)$, as depicted in Fig. 1.4. The corresponding cost function is

$$\begin{aligned} \|\mathbf{e}_{\mathrm{s}}(z)\|_{\mathcal{F}}^2 &= \|[\mathbf{I} - \mathbf{W}(z)\mathbf{A}(z)]\,\mathbf{s}(z) - \mathbf{W}(z)\mathbf{n}(z)\|_{\mathcal{F}}^2 && (1.38)\\ &= \|[\mathbf{I} - \mathbf{W}(z)\mathbf{A}(z)]\,\mathbf{s}(z)\|_{\mathcal{F}}^2 + \|\mathbf{W}(z)\mathbf{n}(z)\|_{\mathcal{F}}^2 && (1.39)\\ &= M\,\sigma_{\mathrm{s}}^2\,\|\mathbf{I} - \mathbf{W}(z)\mathbf{A}(z)\|_{\mathcal{F}}^2 + M\,\sigma_{\mathrm{n}}^2\,\|\mathbf{W}(z)\|_{\mathcal{F}}^2\,. && (1.40) \end{aligned}$$

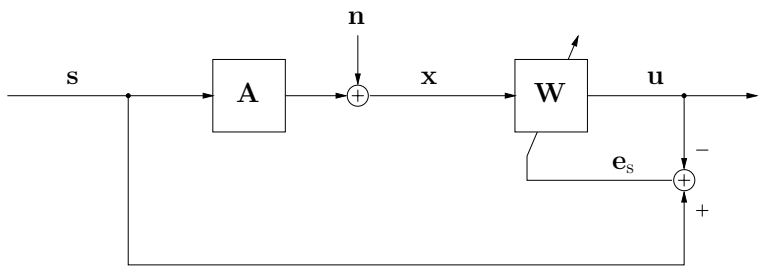

Figure 1.4: Inverse modeling or channel equalization. The equalization error $\mathbf{e}_{\mathrm{s}}$ is used for the adaptation.

Again, we have assumed that the sensor noise and the source signals are mutually uncorrelated, i.e. $\langle \mathbf{s}(z), \mathbf{n}(z) \rangle_{\mathcal{F}} = 0$, all source signals have equal power σ_{s}^2, and all noise signals have equal power σ_{n}^2.

Depending on the application, in fact, we can choose between two different cost functions to minimize. We can either search for the minimum mean-squared error (MMSE) solution $\mathbf{W}^{\text{MMSE}}(z)$, which minimizes (1.40), or search for the so-called zero-forcing (ZF) solution $\mathbf{W}^{\text{ZF}}(z)$, which minimizes only the first term of (1.40), namely $\|\mathbf{I} - \mathbf{W}(z)\mathbf{A}(z)\|_{\mathcal{F}}^2$. In the noiseless case, we have $\mathbf{W}^{\text{MMSE}}(z) = \mathbf{W}^{\text{ZF}}(z)$. In data communications, the MMSE solution is usually preferred, as one is rather interested in estimating the data than obtaining an exact model of the inverse channel, whereas in other applications, e.g. geophysical prospecting, one is more interested in an exact model or inverse model of the channel.

Phase property of $\mathbf{A}(z)$ We introduce the following definitions for an $M \times M$ polynomial matrix $\mathbf{A}(z)$:

- $\mathbf{A}(z)$ is called *minimum phase* if all zeros of $\det \mathbf{A}(z)$ lie inside the unit circle.
- $\mathbf{A}(z)$ is called *maximum phase* if all zeros of $\det \mathbf{A}(z)$ lie outside the unit circle.
- $\mathbf{A}(z)$ is called *mixed phase* if the zeros of $\det \mathbf{A}(z)$ lie on both sides of the unit circle.

The case where the system is either maximum or mixed phase is referred to as *non-minimum phase*. We exclude the case where zeros lie on the unit circle, as such a system is not invertible over the whole frequency range $-\pi < \Omega \leq \pi$.

Inversion of a polynomial matrix A fundamental problem arises if we wish to invert a nonminimum-phase system. Since the elements of $\mathbf{A}(z)$ are polynomials, the elements of $\mathbf{A}^{-1}(z)$ are rational polynomials and therefore have poles, see (A.15). Since we approximate the elements of $\mathbf{A}^{-1}(z)$ with all-zero filters $w_{ij}(z)$, we expand the poles of $[\mathbf{A}^{-1}(z)]_{ij}$ into an infinite-length impulse response.

A pole outside the unit circle can either be expanded such that the resulting impulse response becomes causal but unstable, or non-causal but stable. *Stability can be exchanged with non-causality.* We are interested in a stable inverse, even if we end up with a non-causal system $\mathbf{H}(z)$. As a consequence, if $\mathbf{A}(z)$ in nonminimum phase, we end up with a non-causal $\mathbf{H}(z)$.

In a practical application, where the number of coefficients is limited anyway, we can introduce a delay into the system such that the non-causal part becomes causal again. Hence, we replace (1.35) by

$$\mathbf{e}_{\mathrm{s}}(z) = z^{-d}\,\mathbf{s}(z) - \mathbf{u}(z) \tag{1.41}$$

and in analogy to (1.40), derive the new MSE cost function

$$\|\mathbf{e}_{\mathrm{s}}(z)\|_{\mathcal{F}}^2 = M\,\sigma_{\mathrm{s}}^2\,\left\|\,z^{-d}\,\mathbf{I} - \mathbf{W}(z)\mathbf{A}(z)\right\|_{\mathcal{F}}^2 + M\,\sigma_{\mathrm{n}}^2\,\|\mathbf{W}(z)\|_{\mathcal{F}}^2\,. \tag{1.42}$$

Consequently, the zero-forcing solution then becomes $\mathbf{W}(z) = z^{-d}\,\mathbf{A}^{-1}(z)$ or $\mathbf{W}(z) = z^{-d}\,\mathbf{A}^{\#}(z)$, depending on the dimension of $\mathbf{A}(z)$.

1.5 Blind identification

In the so-called blind identification problem, we have an inverse-modeling problem, except that we have no access to the source signals $\mathbf{s}(z)$. The algorithm is *blind* to the source signals. We distinguish between the following blind problems:

- *Blind source separation* (BSS)
  The unknown system is described by a mixing matrix $\mathbf{A}$ which is an ordinary matrix with scalar elements.

- *Blind deconvolution* (BD)
  The unknown system is described by a single polynomial $a(z)$, representing the z-transformed impulse response $\{a_n\}$.

- *Multichannel blind deconvolution* (MCBD)
  The unknown system is described by a polynomial matrix $\mathbf{A}(z)$. This is the combination of blind source separation and blind deconvolution.

- *Automatic gain control* (AGC)
  AGC belongs to the degenerated case of BSS and BD, where the unknown system is described by a single scalar a. For a complex gain a and a complex source signal, under certain conditions phase corrections up to a multiple of $\pi/2$ can be achieved. We then have automatic gain *and* phase control.

The relationship between the different blind problems is depicted in Fig. 1.5.

Since we have no access to the source signals $\mathbf{s}(z)$, we cannot build the error signal $\mathbf{e}_{\mathrm{s}}$. Hence, we have to estimate either $\mathbf{s}(z)$ or $\mathbf{e}_{\mathrm{s}}$ somehow, or find an alternative error criterion. A common choice for a blind error signal is [9,71]

$$\mathbf{e}_{\mathrm{b}}(z) = \mathbf{u}(z) - \mathbf{y}(z) \tag{1.43}$$

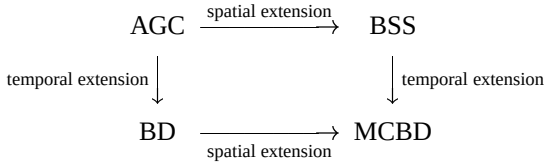

Figure 1.5: Commutative diagram to reveal the relationship between the different blind problems.

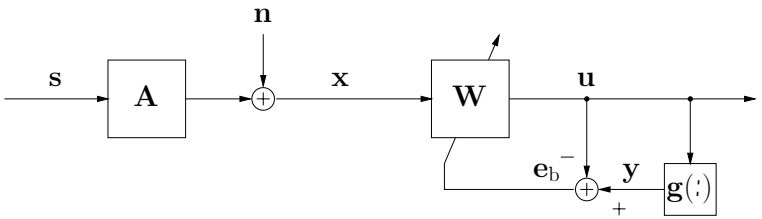

Figure 1.6: Blind identification. The blind error $\mathbf{e}_\mathrm{b}$ is used for the adaptation.

with

$$\mathbf{y}(z) = \sum_{t=-\infty}^{\infty} \mathbf{y}_t z^{-t} = [\, y_m(z) \,] \tag{1.44}$$

$$y_m(z) = \sum_{t=-\infty}^{\infty} y_{m,t} z^{-t} \qquad m = 1, \ldots, M_\mathrm{s} \tag{1.45}$$

$$y_{m,t} = g_m(u_{m,t})\,. \tag{1.46}$$

$y_{m,t}$ is a nonlinear function of the output signal $u_{m,t}$. The choice of the memoryless nonlinearity $g_m(.)$, also known as the *Bussgang nonlinearity*, depends on $p_{S_m}(s_m)$, the pdf of the unknown source signal s_m. If the pdf $p_S(s)$ of a source signal is known, the *score function* [16], defined as

$$g(u) = -\frac{\partial}{\partial u} \ln p_S(u) = -\frac{\frac{\partial}{\partial u} p_S(u)}{p_S(u)} \tag{1.47}$$

is usually the preferred choice, justified from maximum-likelihood estimation theory. However, the exact choice of the nonlinearity is not very crucial for the performance of most blind algorithms. In fact, the knowledge whether the pdf of a source signal is super-Gaussian (more peaky than a Gaussian pdf) or sub-Gaussian (flatter than a Gaussian pdf) is usually sufficient for the choice of the nonlinearity.

1.6 Semi-blind identification

In a semi-blind setup we refer to the situation where some parts of the source signals are known. We distinguish between spatial and temporal semi-blind problems:

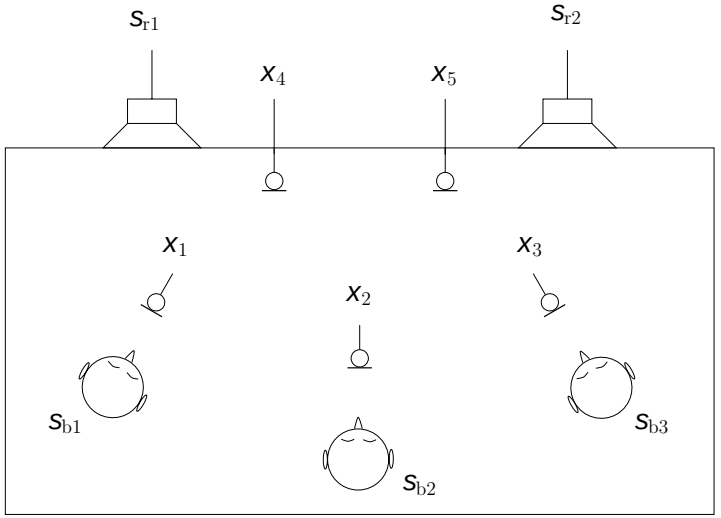

Figure 1.7: Teleconferencing setup. The five sensor signals x_1 to x_5 capture a mixture of several audio signals stemming from three speakers (unknown source signals $s_{\mathrm{b}1}$ to $s_{\mathrm{b}3}$), and two loudspeaker signals (accessible source signals $s_{\mathrm{r}1}$ and $s_{\mathrm{r}2}$). The objective is to retrieve the unknown source signals $s_{\mathrm{b}m}$. If a blind-only algorithm is used, all five sensor signals are required, as a blind algorithm does not distinguish between known and unknown source signals. A semi-blind algorithm which makes use of the known source signals requires only three sensors.

- Some of the source signals are known. This situation appears e.g. in a teleconferencing setup, see Fig. 1.7. We have the situation, where some of the source signals are accessible, and therefore do not have to be separated from the mixture. The known source signals can either be directly incorporated in the update equation as *virtual sensors* [61], or, in an echo-canceler preprocessing step, be subtracted from the sensor signals [59]. Moreover, every known source signal reduces the number of required sensors for the separation by one. Other algorithms which cope with this situation are given in [92–94].

- In communications, training sequences are usually embedded in the data stream to allow a training-based equalization of the channel. During the actual data transmission, the adaptive channel equalizer operates in a

fully blind or decision-directed mode, depending on the current quality of the equalization [45, 108, 111].

1.7 Overview

In this thesis we derive several algorithms for single-channel and multichannel adaptive filtering. First we analyze the non-blind case, where we derive update rules for the system-identification and for the inverse-modeling problem. Afterwards, based upon the concepts of the non-blind case, we modify the algorithms to be applicable to a blind environment. The thesis is structured as follows:

chapter 1 In this chapter the general problem description is introduced in a mathematical context. Furthermore, some special cases of the general problem are given.

chapter 2 In this chapter we deal with the instantaneous mixing problem. Based on a quadratic error criterion, several gradient and Quasi-Newton-type algorithms are derived for multichannel identification and multichannel equalization, which are related to LMS and RLS algorithms.

chapter 3 In this chapter the basic tools from Linear Algebra which build the basis for deriving efficient blind and non-blind adaptive algorithms are introduced. The Fourier matrix, circulant matrices, and block circulant matrices play a major role in transforming the filtering and the update of the coefficients into the frequency domain. Moreover, the isomorphic mapping between convolution, multiplication of polynomials, and multiplication of circulant matrices are shown, which build the key concept for the extension of the instantaneous mixing case to the convolutive mixing case.

chapter 4 Here we consider the single-channel system-identification and inverse-modeling problem, where the unknown system consists of a single filter. With the help of the mathematical tools described in Chapter 3, we transform the update rules from Chapter 2 to work with the convolutive-mixing case. Furthermore, efficient implementations of the algorithms in the frequency domain are given.

chapter 5 In this chapter we focus on algorithms for multichannel system identification and inverse modeling. We combine the concepts of the multichannel instantaneous-mixing problem from Chapter 2 with the concepts of the single-channel filtering problem from Chapter 4, and derive algorithms for the multichannel convolutive-mixing problem.

chapter 6 Based on the methods for the described non-blind problems and by exchanging the non-blind error criterion by a blind error criterion, we can easily obtain algorithms which are suitable for blind source separation (BSS), blind deconvolution (BD), and multichannel blind deconvolution (MCBD).

appendix 1 Summary of some useful mathematical tools.

appendix 2 Properties of the trace operation.

appendix 3 Extension of the Frobenius norm to polynomial matrices.

appendix 4 The definitions of the generalized remainder and the polynomial projection operators are given, together with many of their properties.

appendix 5 Summary of update equations for system identification, inverse modeling, and blind identification.

appendix 6 MATLAB implementation of a single-channel blind deconvolution algorithm in the frequency domain.

Chapter 2

System identification and inverse modeling of an instantaneous mixing system

In this chapter we derive several adaptive algorithms for system identification and inverse modeling (inverse-system identification) of an instantaneous mixing system. An instantaneous mixing system can be seen as a special case of multichannel system identification or multichannel inverse modeling where the unknown system has no dynamics (memoryless system), and can therefore be described by an ordinary matrix, whose elements are scalars. The purpose is hereby to gain a first insight into the behavior of a general multichannel algorithm and its properties. Furthermore, as will be seen in Chapter 6, the analysis and understanding of *non-blind* algorithms helps in the development of *blind* algorithms and in the improvement of their convergence rate.

In this chapter we derive two Wiener solutions based on different error criteria. From the Wiener solutions, we develop several LMS and RLS update equations, which are summarized in Table E.3 and E.7 in Appendix E. The matrix-inversion lemma, which is given in Appendix A.3, plays a key role in the transformation of an algorithm which is applicable for system identification into one for inverse modeling, and vice versa. As a result, we will see that many known blind algorithms also have non-blind counterparts.

2.1 Unknown mixing system

Since we consider an instantaneous (memoryless) mixing system, we can describe the unknown system by a mixing matrix with scalar elements, also shown in Fig. 1.1

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t + \mathbf{n}_t\,. \tag{2.1}$$

The mixing matrix $\mathbf{A}$ and the noise vector $\mathbf{n}_t$ are unknown, the signal vectors $\mathbf{s}_t$ and $\mathbf{x}_t$ are known for $t \geq 0$. The task is now to find an estimate of $\mathbf{A}$, i.e. $\mathbf{H} = \hat{\mathbf{A}}$, or an estimate of $\mathbf{A}^{-1}$, i.e. $\mathbf{W} = \hat{\mathbf{A}}^{-1}$, based on the knowledge of the system input $\mathbf{s}_t$ and system output $\mathbf{x}_t$. The first task is referred to as *system identification* and the second one as *inverse modeling* or *inverse-system identification*. As we will see in this chapter, the algorithms for these tasks can have quite a different performance behavior, although they actually pursue the same objective.

2.2 System identification

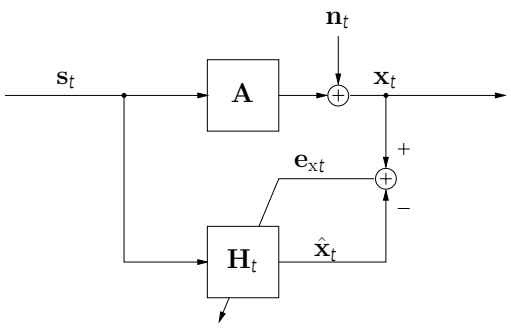

Figure 2.1: System identification. The error signal $\mathbf{e}_{\mathrm{x}}$ is used for the adaptation.

In *system identification* we try to find a matrix $\mathbf{H}$ which is "close" to the true system $\mathbf{A}$, see Fig. 2.1. "Closeness" can be measured in different ways. For example it could be defined by $\|\mathbf{A} - \mathbf{H}\|_F$. However, by assumption there is no access to the true matrix $\mathbf{A}$, therefore closeness is usually defined in system identification as a function of the prediction or estimation error

$$\mathbf{e}_{\mathrm{x}t} \triangleq \mathbf{x}_t - \hat{\mathbf{x}}_t \tag{2.2}$$

where

$$\hat{\mathbf{x}}_t = \mathbf{H}_t\mathbf{s}_t \tag{2.3}$$

is the estimation of the output of the unknown mixing system. $\mathbf{H}$ can be either estimated by a batch algorithm, or by an online learning algorithm. A batch algorithm uses at discrete time t all the past input and output time samples for the new estimate $\mathbf{H}_{t+1}$, whereas an online algorithm uses at time t only $\mathbf{s}_t$, $\mathbf{x}_t$, and the current estimate $\mathbf{H}_t$ to evaluate $\mathbf{H}_{t+1}$.

2.2.1 Wiener solution $\mathbf{H}^{\text{MMSE-x}}$

We now wish to find the Wiener or minimum mean-squared error (MMSE) solution $\mathbf{H}^{\text{MMSE-x}}$, which minimizes the cost function

$$J_{\text{MSE-x}} \triangleq E\left\{\|\mathbf{e}_{\mathrm{x}}\|_2^2\right\} = \mathrm{tr}\left\{\mathbf{R}_{\mathbf{e_x e_x}}\right\} \tag{2.4}$$

where

$$\mathbf{R}_{\mathbf{e_x e_x}} \triangleq E\left\{\mathbf{e}_{\mathrm{x}}\mathbf{e}_{\mathrm{x}}^H\right\} = E\left\{(\mathbf{x}-\hat{\mathbf{x}})\,(\mathbf{x}-\hat{\mathbf{x}})^H\right\} \tag{2.5}$$

$$= \mathbf{R}_{\mathbf{xx}} - \mathbf{H}\,\mathbf{R}_{\mathbf{sx}} - \mathbf{R}_{\mathbf{xs}}\mathbf{H}^H + \mathbf{H}\,\mathbf{R}_{\mathbf{ss}}\mathbf{H}^H \tag{2.6}$$

is the error covariance matrix, $\mathbf{R}_{\mathbf{ss}} \triangleq E\left\{\mathbf{s}\mathbf{s}^H\right\}$, $\mathbf{R}_{\mathbf{sx}} \triangleq E\left\{\mathbf{s}\mathbf{x}^H\right\}$, $\mathbf{R}_{\mathbf{xs}} \triangleq E\left\{\mathbf{x}\mathbf{s}^H\right\}$, and $\mathbf{R}_{\mathbf{xx}} \triangleq E\left\{\mathbf{x}\mathbf{x}^H\right\}$. Towards this end, we build the gradient of the cost function with respect to the matrix $\mathbf{H}$

$$\nabla_{\mathbf{H}} J_{\text{MSE-x}} = \nabla_{\mathbf{H}}\,\mathrm{tr}\left\{\mathbf{R}_{\mathbf{e_x e_x}}\right\} = -2\mathbf{R}_{\mathbf{xs}} + 2\mathbf{H}\,\mathbf{R}_{\mathbf{ss}} \tag{2.7}$$

where we used $\nabla_{\mathbf{H}} = 2\frac{\partial}{\partial \mathbf{H}^*}$, see [51] Eq. (B.18). By setting $\nabla_{\mathbf{H}} J_{\text{MSE-x}}$ equal to $\mathbf{0}$ and solving for $\mathbf{H}$, we finally obtain the MMSE or Wiener solution

$$\mathbf{H}^{\text{MMSE-x}} = \mathbf{R}_{\mathbf{xs}}\mathbf{R}_{\mathbf{ss}}^{-1}\,. \tag{2.8}$$

Alternatively, we can also derive the Wiener solution by using the *orthogonality principle* [1] which says that the estimation error has to be orthogonal to

[1] Two random complex variables X and Y with $E\{X\} = E\{Y\} = 0$ are said to be *orthogonal* if $E\{XY^*\} = 0$.

the input data

$$E\left\{\mathbf{e}_{\mathrm{x}}\mathbf{s}^H\right\} = E\left\{\mathbf{x}\mathbf{s}^H\right\} - \mathbf{H}E\left\{\mathbf{s}\mathbf{s}^H\right\} \tag{2.9}$$
$$= \mathbf{R}_{\mathbf{xs}} - \mathbf{H}\,\mathbf{R}_{\mathbf{ss}}\,. \tag{2.10}$$

By setting (2.10) equal to $\mathbf{0}$ we obtain the *Wiener Hopf equation* (WHE)

$$\mathbf{H}\,\mathbf{R}_{\mathbf{ss}} = \mathbf{R}_{\mathbf{xs}}\,. \tag{2.11}$$

Solving for $\mathbf{H}$ also gives $\mathbf{H}^{\text{MMSE-x}}$ given in (2.8).

2.2.2 Batch learning

Since we usually do not know the true covariance matrices $\mathbf{R}_{\mathbf{xs}}$ and $\mathbf{R}_{\mathbf{ss}}$, we can replace them by their estimates $\hat{\mathbf{R}}_{\mathbf{xs}}$ and $\hat{\mathbf{R}}_{\mathbf{ss}}$, respectively. We can then estimate the Wiener solution in (2.8) by

$$\mathbf{H} = \hat{\mathbf{R}}_{\mathbf{xs}}\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\,. \tag{2.12}$$

In a batch processing, we use all available measurements of the system to estimate the correlation matrices

$$\hat{\mathbf{R}}_{\mathbf{xs}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_t\mathbf{s}_t^H \tag{2.13}$$

$$\hat{\mathbf{R}}_{\mathbf{ss}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{s}_t\mathbf{s}_t^H\,. \tag{2.14}$$

The number of samples T needs to be large enough such that $\hat{\mathbf{R}}_{\mathbf{ss}}$ has full rank and is therefore invertible.

2.3 LMS-x

For the derivation of a stochastic learning algorithm we first start with the method of *steepest descent* where we iteratively update $\mathbf{H}$ by following the

negative gradient of the performance function

$$\mathbf{H}_{t+1} = \mathbf{H}_t - \frac{\mu}{2}\nabla_{\mathbf{H}}J_{\text{MSE-x}} \tag{2.15}$$
$$= \mathbf{H}_t + \mu\left(\mathbf{R}_{\mathbf{xs}} - \mathbf{H}_t\,\mathbf{R}_{\mathbf{ss}}\right) \tag{2.16}$$
$$= \mathbf{H}_t\left(\mathbf{I} - \mu\,\mathbf{R}_{\mathbf{ss}}\right) + \mu\,\mathbf{R}_{\mathbf{xs}} \tag{2.17}$$
$$= \mathbf{H}_t + \mu\left(\mathbf{R}_{\mathbf{xs}} - \mathbf{R}_{\hat{\mathbf{x}}\mathbf{s}}\right) \tag{2.18}$$

where μ is the step size of the algorithm, which is properly chosen such that convergence is guaranteed, i.e.,

$$0 < \mu < \frac{2}{\|\mathbf{R}_{\mathbf{ss}}\|}\,. \tag{2.19}$$

2.3.1 Updating H

LMS1-Hx Since the true correlation matrices $\mathbf{R}_{\mathbf{xs}}$, $\mathbf{R}_{\mathbf{ss}}$, and $\mathbf{R}_{\hat{\mathbf{x}}\mathbf{s}}$ are usually unknown, we have to estimate them. The simplest way to do this is to replace them by their *instantaneous estimates*, e.g. $\hat{\mathbf{R}}_{\mathbf{xs}} = \mathbf{x}\mathbf{s}^H$. By doing so, we derive a *stochastic gradient algorithm*

$$\mathbf{H}_{t+1} = \mathbf{H}_t + \mu\left(\mathbf{x} - \hat{\mathbf{x}}\right)\mathbf{s}^H \tag{2.20}$$

which we refer to as LMS1-Hx. We call it LMS, because the algorithm is derived analogously to the *least mean square algorithm* used in adaptive filtering [100]. The index H denotes the variable which is adapted and x denotes that the underlying cost function which is to be minimized is $J_{\text{MSE-x}}$.

LMS2-Hx In case we know that $\mathbf{R}_{\mathbf{ss}} = \mathbf{I}$, which is often assumed in a blind setup, we obtain from (2.16)

$$\mathbf{H}_{t+1} = \mathbf{H}_t + \mu\left(\mathbf{x}\mathbf{s}^H - \mathbf{H}_t\right) \tag{2.21}$$
$$= (1-\mu)\,\mathbf{H}_t + \mu\mathbf{x}\mathbf{s}^H \tag{2.22}$$

which we refer to as LMS2-Hx. $\mathbf{R}_{\mathbf{xs}}$ was replaced by its instantaneous estimate.

2.3.2 Updating $\mathbf{H}^{-1}$

Either algorithm, LMS1-Hx and LMS2-Hx, can be modified such as to update the inverse of $\mathbf{H}$, i.e. $\mathbf{W} \triangleq \mathbf{H}^{-1}$, rather than $\mathbf{H}$ itself. By doing so, we obtain two new algorithms which can be used for inverse modeling.

LMS1a-Wx Starting with (2.20) and applying the matrix-inversion lemma with $\mathbf{A}' = \mathbf{H}_t$, $\mathbf{B}' = \mu\,(\mathbf{x} - \hat{\mathbf{x}})$, $\mathbf{C}' = 1$, and $\mathbf{D}' = \mathbf{s}^H$, we derive

$$\begin{aligned}\mathbf{W}_{t+1} = [\mathbf{H}_{t+1}]^{-1} &= \left[\mathbf{H}_t + \mu\,(\mathbf{x} - \hat{\mathbf{x}})\,\mathbf{s}^H\right]^{-1} \qquad (2.23)\\ &= \mathbf{W}_t - \mu\mathbf{W}_t\,(\mathbf{x} - \hat{\mathbf{x}})\left[1 + \mu\mathbf{s}^H\mathbf{W}_t\,(\mathbf{x} - \hat{\mathbf{x}})\right]^{-1}\mathbf{s}^H\mathbf{W}_t\\ &= \mathbf{W}_t + \mu\,(\mathbf{s} - \mathbf{u})\left[1 - \mu\mathbf{s}^H\,(\mathbf{s} - \mathbf{u})\right]^{-1}\mathbf{s}^H\mathbf{W}_t\,. \qquad (2.24)\end{aligned}$$

Recall that $\mathbf{H}_t = \mathbf{W}_t^{-1}$, $\mathbf{W}_t\mathbf{x} = \mathbf{u}$ and $\mathbf{W}_t\hat{\mathbf{x}} = \mathbf{H}_t^{-1}\hat{\mathbf{x}} = \mathbf{s}$. We refer to this algorithm as LMS1a-Wx, which is, in fact, exactly the same algorithm as LMS1-Hx, except that $\mathbf{H}^{-1}$ is updated instead of $\mathbf{H}$.

LMS1b-Wx Again, we start with (2.23) and applying the matrix-inversion lemma with $\mathbf{A}' = \mathbf{H}_t$, $\mathbf{B}' = \mu\,(\mathbf{x} - \hat{\mathbf{x}})\,\mathbf{s}^H$, $\mathbf{C}' = \mathbf{I}$, and $\mathbf{D}' = \mathbf{I}$, we derive

$$\begin{aligned}\mathbf{W}_{t+1} &= \mathbf{W}_t - \mu\mathbf{W}_t\,(\mathbf{x} - \hat{\mathbf{x}})\,\mathbf{s}^H\left[\mathbf{I} + \mu\mathbf{W}_t\,(\mathbf{x} - \hat{\mathbf{x}})\,\mathbf{s}^H\right]^{-1}\mathbf{W}_t\\ &= \mathbf{W}_t + \mu\,(\mathbf{s} - \mathbf{u})\,\mathbf{s}^H\left[\mathbf{I} - \mu\,(\mathbf{s} - \mathbf{u})\,\mathbf{s}^H\right]^{-1}\mathbf{W}_t\,. \qquad (2.25)\end{aligned}$$

We refer to this algorithm as LMS1b-Wx, which is exactly the same algorithm as LMS1-Hx, except that $\mathbf{H}^{-1}$ is updated instead of $\mathbf{H}$. Furthermore, this algorithm is identical to the LMS1a-Wx. However, as we will see, there is a difference in the convergence behavior, if the algorithm is updated in a block-wise manner, as described in Section 2.9.

LMS2a-Wx In a similar way we can start with the inverse of (2.22)

$$\mathbf{W}_{t+1} = [\mathbf{H}_{t+1}]^{-1} = \left[(1-\mu)\,\mathbf{H}_t + \mu\mathbf{x}\mathbf{s}^H\right]^{-1} \qquad (2.26)$$

and apply the matrix-inversion lemma with $\mathbf{A}' = (1-\mu)\,\mathbf{H}_t$, $\mathbf{B}' = \mu\mathbf{x}$, $\mathbf{C}' = 1$, and $\mathbf{D}' = \mathbf{s}^H$, and after some calculations we finally obtain

$$\mathbf{W}_{t+1} = \frac{1}{1-\mu}\left(\mathbf{I} - \frac{\mu}{1-\mu+\mu\mathbf{s}^H\mathbf{u}}\mathbf{u}\mathbf{s}^H\right)\mathbf{W}_t \qquad (2.27)$$

which we will refer to as LMS2a-Wx [59]. Both algorithms, LMS1a-Wx and LMS2a-Wx belong to the class of *serial update algorithms*, as they can be written as a matrix product $\mathbf{W}_{t+1} = \Delta\mathbf{W}_t\mathbf{W}_t = \left(\prod_{\tau=1}^{t}\Delta\mathbf{W}_\tau\right)\mathbf{W}_0$ [17]. The update $\Delta\mathbf{W}_t$ is applied multiplicatively in the update equation and converges towards the unity matrix. Note that the signal vector $\mathbf{x}$ does not explicitly appear in these two update equations.

LMS2b-Wx In a similar way we can start with the inverse of (2.26) and apply the matrix-inversion lemma with $\mathbf{A}' = (1-\mu)\,\mathbf{H}_t$, $\mathbf{B}' = \mu\mathbf{x}\mathbf{s}^H$, $\mathbf{C}' = \mathbf{I}$, and $\mathbf{D}' = \mathbf{I}$, and after some calculations we finally obtain

$$\begin{aligned}\mathbf{W}_{t+1} &= \frac{1}{1-\mu}\left(\mathbf{I} - \frac{\mu}{1-\mu}\mathbf{u}\mathbf{s}^H\left[\mathbf{I} + \frac{\mu}{1-\mu}\mathbf{u}\mathbf{s}^H\right]^{-1}\right)\mathbf{W}_t\\ &= \frac{1}{1-\mu}\left(\mathbf{I} - \mu\mathbf{u}\mathbf{s}^H\left[(1-\mu)\mathbf{I} + \mu\mathbf{u}\mathbf{s}^H\right]^{-1}\right)\mathbf{W}_t\\ &= \left[(1-\mu)\mathbf{I} + \mu\mathbf{u}\mathbf{s}^H\right]^{-1}\mathbf{W}_t \qquad (2.28)\end{aligned}$$

which we will refer to as LMS2b-Wx.

2.4 RLS-x

An alternative to the batch algorithm (2.12), where the correlation matrices $\mathbf{R}_{\mathbf{xs}}$ and $\mathbf{R}_{\mathbf{ss}}$ are estimated in (2.13) and (2.14) with the help of all available measurement data, respectively, is to estimate the correlation matrices recursively

$$\hat{\mathbf{R}}_{\mathbf{xs}_t} = \hat{\mathbf{R}}_{\mathbf{xs}_{t-1}} + \mathbf{x}_t\mathbf{s}_t^H \qquad (2.29)$$

$$\hat{\mathbf{R}}_{\mathbf{ss}_t} = \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}} + \mathbf{s}_t\mathbf{s}_t^H\,. \qquad (2.30)$$

We then obtain an online learning algorithm

$$\mathbf{H}_{t+1} = \hat{\mathbf{R}}_{\mathbf{xs}_t}\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1} \qquad (2.31)$$

which we will refer to as RLS1-x, as it is actually a recursive least squares (RLS) algorithm which minimizes the cost function

$$\tilde{J}_{\text{MSE-x}} = \sum_{\tau=0}^{t}\|\mathbf{x}_\tau - \mathbf{H}_t\,\mathbf{s}_\tau\|_2^2 \qquad (2.32)$$

at discrete time t.

Note, $\hat{\mathbf{R}}_{\mathbf{xs}_t}$ from (2.29) and $\hat{\mathbf{R}}_{\mathbf{ss}_t}$ from (2.30) are not consistent estimates of $\mathbf{R}_{\mathbf{xs}}$and $\mathbf{R}_{\mathbf{ss}}$, respectively, as their norms grow with time t. However, $\mathbf{H}_{t+1}$ from (2.31) is a consistent estimate of $\mathbf{H}^{\text{MMSE-x}}$, as $\hat{\mathbf{R}}_{\mathbf{xs}_t}$ and $\hat{\mathbf{R}}_{\mathbf{ss}_t}$ grow with the same rate.

2.4.1 RLS1-Hx

The matrix inversion in (2.31), which is carried out at every time instant t, is a very demanding task. As we will now see, again with the help of the matrix-inversion lemma, we can circumvent this matrix inversion. To this end, we recursively update $\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1}$ instead of $\hat{\mathbf{R}}_{\mathbf{ss}_t}$. By inverting both sides of (2.30) and using the matrix-inversion lemma with $\mathbf{A}' = \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}$, $\mathbf{B}' = \mathbf{s}_t$, $\mathbf{C}' = 1$, and $\mathbf{D}' = \mathbf{s}_t^H$ we obtain

$$\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1} = \left[\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}} + \mathbf{s}_t\mathbf{s}_t^H\right]^{-1} \tag{2.33}$$

$$= \hat{\mathbf{R}}_{\mathbf{ss}}^{-1} - \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\left[1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\right]^{-1}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1} \tag{2.34}$$

$$= \hat{\mathbf{R}}_{\mathbf{ss}}^{-1} - \tilde{\mu}\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1} \tag{2.35}$$

$$\tilde{\mu} = \frac{1}{1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}} \tag{2.36}$$

where $\tilde{\mu}$ can be seen as a step size of the update. Next we use (2.34) to modify (2.31) and find an efficient update for $\mathbf{H}_{t+1}$

$$\begin{aligned}
\mathbf{H}_{t+1} &= \hat{\mathbf{R}}_{\mathbf{xs}_t}\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1} \qquad (2.37)\\
&= \left(\hat{\mathbf{R}}_{\mathbf{xs}_{t-1}} + \mathbf{x}_t\mathbf{s}_t^H\right)\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1}\\
&= \left(\hat{\mathbf{R}}_{\mathbf{xs}} + \mathbf{x}\mathbf{s}^H\right)\left(\hat{\mathbf{R}}_{\mathbf{ss}}^{-1} - \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\left[1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\right]^{-1}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\right)\\
&= \mathbf{H}_t - \mathbf{H}_t\mathbf{s}\left[1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\right]^{-1}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1} + \mathbf{x}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\\
&\quad - \mathbf{x}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\left[1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\right]^{-1}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\\
&= \mathbf{H}_t - \mathbf{H}_t\mathbf{s}\left[1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\right]^{-1}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\\
&\quad + \mathbf{x}\left[1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\right]\left[1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\right]^{-1}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\\
&\quad - \mathbf{x}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\left[1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}\right]^{-1}\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\\
&= \mathbf{H}_t + \frac{1}{1 + \mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}}\left(\mathbf{x} - \mathbf{H}_t\mathbf{s}\right)\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1} \qquad (2.38)\\
&= \mathbf{H}_t + \tilde{\mu}\left(\mathbf{x} - \hat{\mathbf{x}}\right)\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1} \qquad (2.39)
\end{aligned}$$

where $\mathbf{e}_{\text{x}} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \mathbf{H}_t\mathbf{s}$ is the current estimation-error vector. In the above derivation we sometimes omitted the time-sample index for $\mathbf{s}_t$, $\mathbf{x}_t$, $\hat{\mathbf{R}}_{\mathbf{xs}_{t-1}}$, and $\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}$. We further used the fact that $\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{s}$ is a scalar.

2.4.2 RLS1-Wx

In case we want to update $\mathbf{W}_{t+1}$ instead of $\mathbf{H}_{t+1}$, we can invert both sides of (2.31) and use $\mathbf{W}_{t+1} \triangleq \mathbf{H}_{t+1}^{-1}$

$$\mathbf{W}_{t+1} = \hat{\mathbf{R}}_{\mathbf{ss}_t}\hat{\mathbf{R}}_{\mathbf{xs}_t}^{-1}. \tag{2.40}$$

However, just as in (2.31), we again need a matrix inversion for every update. To avoid this, we start by inverting both sides of (2.39)

$$\mathbf{W}_{t+1} = \left[\mathbf{W}_t^{-1} + \tilde{\mu}\left(\mathbf{x} - \mathbf{W}_t^{-1}\mathbf{s}\right)\mathbf{s}^H\hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\right]^{-1}. \tag{2.41}$$

Applying the matrix-inversion lemma with $\mathbf{A}' = \mathbf{W}_t^{-1}$, $\mathbf{B}' = \tilde{\mu}\left(\mathbf{x} - \mathbf{W}_t^{-1}\mathbf{s}\right)$, $\mathbf{C}' = 1$, and $\mathbf{D}' = \mathbf{s}^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}$ gives

$$\begin{aligned}\mathbf{W}_{t+1} &= \mathbf{W}_t - \tilde{\mu}\mathbf{W}_t\left(\mathbf{x} - \mathbf{W}_t^{-1}\mathbf{s}\right) \\ &\quad \cdot \left[1 + \tilde{\mu}\mathbf{s}^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{W}_t\left(\mathbf{x} - \mathbf{W}_t^{-1}\mathbf{s}\right)\right]^{-1} \mathbf{s}^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{W}_t \\ &= \mathbf{W}_t + \tilde{\mu}\left(\mathbf{s} - \mathbf{u}\right)\left[1 - \tilde{\mu}\mathbf{s}^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\left(\mathbf{s} - \mathbf{u}\right)\right]^{-1} \mathbf{s}^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{W}_t \\ &= \mathbf{W}_t + \mu\left(\mathbf{s} - \mathbf{u}\right)\mathbf{s}^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{W}_t \end{aligned} \tag{2.42}$$

with

$$\mu = \frac{\tilde{\mu}}{1 - \tilde{\mu}\mathbf{s}^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\left(\mathbf{s} - \mathbf{u}\right)} = \frac{1}{1 + \mathbf{s}^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1}\mathbf{u}} \tag{2.43}$$

where $\tilde{\mu}$ is defined in (2.36). In the above derivation we used $\mathbf{u} = \mathbf{W}_t\mathbf{x}$. Note that $\mathbf{e}_{\mathrm{s}} \triangleq \mathbf{s} - \mathbf{u}$ is also an estimation error, however, in general not the same as $\mathbf{e}_{\mathrm{x}}$.

2.4.3 RLS1-Hx with exponential forgetting

To track time-varying systems, the cost function defined in (2.32) is extended by an exponential weighting of the past measurements

$$\tilde{J}_{\text{MSE-x}} = \sum_{\tau=0}^{t} \lambda^{t-\tau} \left\|\mathbf{x}_\tau - \mathbf{H}_t\,\mathbf{s}_\tau\right\|_2^2 . \tag{2.44}$$

This causes exponential forgetting in the recursive estimates of the correlation matrices

$$\hat{\mathbf{R}}_{\mathbf{xs}_t} = \lambda\hat{\mathbf{R}}_{\mathbf{xs}_{t-1}} + (1-\lambda)\mathbf{x}_t\mathbf{s}_t^H \tag{2.45}$$

$$\hat{\mathbf{R}}_{\mathbf{ss}_t} = \lambda\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}} + (1-\lambda)\mathbf{s}_t\mathbf{s}_t^H \tag{2.46}$$

where λ denotes a forgetting factor with $1 > \lambda > 0$. Doing similar calculations as in Section 2.4.1 yields the same update equation for $\mathbf{H}_{t+1}$ but with a different step size $\tilde{\mu}$

$$\tilde{\mu}_t = \frac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\mathbf{s}_t} \tag{2.47}$$

$$\mathbf{H}_{t+1} = \mathbf{H}_t + \tilde{\mu}_t\left(\mathbf{x}_t - \hat{\mathbf{x}}_t\right)\mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \tag{2.48}$$

$$\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} - \tilde{\mu}_t\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\mathbf{s}_t\mathbf{s}_t^H\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\right) \tag{2.49}$$

where (2.47), (2.48), and (2.49) are referred to as RLS1-Hx.

2.4.4 RLS1-Wx with exponential forgetting

Using exponential forgetting and doing similar calculations as in Section 2.4.2 yields the same update equation for $\mathbf{W}_{t+1}$ but with different step sizes μ and $\tilde{\mu}$

$$\mu_t = \frac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\mathbf{u}_t} \tag{2.50}$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t\left(\mathbf{s}_t - \mathbf{u}_t\right)\mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\mathbf{W}_t \tag{2.51}$$

$$\tilde{\mu}_t = \frac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\mathbf{s}_t} \tag{2.52}$$

$$\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} - \tilde{\mu}_t\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\mathbf{s}_t\mathbf{s}_t^H\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\right) \tag{2.53}$$

where (2.50), (2.51), (2.52), and (2.53) are referred to as RLS1-Wx.

As we will see in Chapter 6, the blind version of the RLS1-Wx will be a key algorithm for the BSS problem.

2.5 Inverse modeling

In *inverse modeling* we try to find a matrix $\mathbf{W} = \hat{\mathbf{A}}^{-1}$ which is close to the inverse of the true system $\mathbf{A}^{-1}$, see also Fig. 2.2. Closeness can again be measured in different ways. For example it could be defined by $\|\mathbf{A}^{-1} - \mathbf{W}\|_F$ or $\|\mathbf{I} - \mathbf{G}\|_F$ where

$$\mathbf{G}_t = \mathbf{W}_t\mathbf{A} \tag{2.54}$$

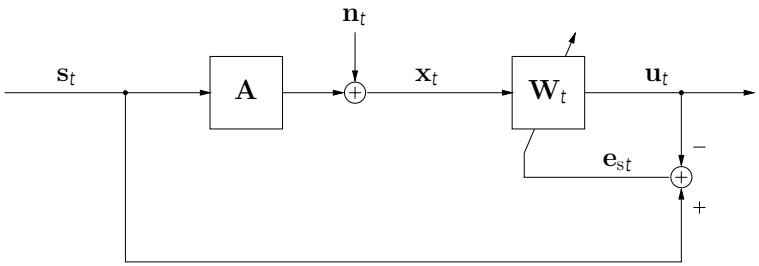

Figure 2.2: Inverse modeling or inverse system identification. The error signal $\mathbf{e}_{\mathrm{s}}$ is used for the adaptation.

is the global response. However, by assumption there is no access to the true matrix $\mathbf{A}$, therefore closeness is usually defined in system identification as a function of the prediction or estimation error

$$\mathbf{e}_{\mathrm{s}t} \triangleq \mathbf{s}_t - \hat{\mathbf{s}}_t = \mathbf{s}_t - \mathbf{u}_t \tag{2.55}$$

where

$$\mathbf{u}_t = \hat{\mathbf{s}}_t = \mathbf{W}_t\mathbf{x}_t = \mathbf{G}_t\mathbf{s}_t + \mathbf{W}_t\mathbf{n}_t \tag{2.56}$$

is the estimation of the source signals. $\mathbf{W}$ can be either estimated by a batch algorithm, or by an online learning algorithm. A batch algorithm uses at discrete time t all the past input and output time samples for the new estimate $\mathbf{W}_{t+1}$, whereas an online algorithm uses at time t only $\mathbf{s}_t$, $\mathbf{x}_t$, and the current estimate $\mathbf{W}_t$ to evaluate $\mathbf{W}_{t+1}$.

2.5.1 Wiener solution $\mathbf{W}^{\text{MMSE-s}}$

We now wish to find the Wiener or minimum mean-squared error (MMSE) solution $\mathbf{W}^{\text{MMSE-s}}$ which minimizes the cost function

$$J_{\text{MSE-s}} \triangleq E\left\{\|\mathbf{e}_{\mathrm{s}}\|_2^2\right\} = \mathrm{tr}\left\{\mathbf{R}_{\mathbf{e_s e_s}}\right\} \tag{2.57}$$

where

$$\mathbf{R}_{\mathbf{e_s e_s}} \triangleq E\left\{\mathbf{e}_{\mathrm{s}}{\mathbf{e}_{\mathrm{s}}}^H\right\} = E\left\{(\mathbf{s}-\mathbf{u})(\mathbf{s}-\mathbf{u})^H\right\} \tag{2.58}$$

$$= \mathbf{R}_{\mathbf{ss}} - \mathbf{W}\mathbf{R}_{\mathbf{xs}} - \mathbf{R}_{\mathbf{sx}}\mathbf{W}^H + \mathbf{W}\mathbf{R}_{\mathbf{xx}}\mathbf{W}^H \tag{2.59}$$

is the error covariance matrix. Towards this end, we build the gradient of the cost function with respect to the matrix $\mathbf{W}$

$$\nabla_{\mathbf{W}} J_{\text{MSE-s}} = \nabla_{\mathbf{W}}\,\mathrm{tr}\left\{\mathbf{R}_{\mathbf{e_s e_s}}\right\} = -2\mathbf{R}_{\mathbf{sx}} + 2\mathbf{W}\mathbf{R}_{\mathbf{xx}} \tag{2.60}$$

where we used $\nabla_{\mathbf{W}} = 2\frac{\partial}{\partial \mathbf{W}^*}$, see [51] Eq. (B.18). By setting $\nabla_{\mathbf{W}} J_{\text{MSE-s}}$ equal to $\mathbf{0}$ and solving for $\mathbf{W}$, we finally obtain the MMSE or Wiener solution

$$\mathbf{W}^{\text{MMSE-s}} = \mathbf{R}_{\mathbf{sx}}\mathbf{R}_{\mathbf{xx}}^{-1}. \tag{2.61}$$

Alternatively, we can also derive the Wiener solution by using the orthogonality principle which says that the estimation error has to be orthogonal to the input data

$$E\left\{\mathbf{e}_{\mathrm{s}}\mathbf{x}^H\right\} = E\left\{\mathbf{s}\mathbf{x}^H\right\} - \mathbf{W}E\left\{\mathbf{x}\mathbf{x}^H\right\} \tag{2.62}$$

$$= \mathbf{R}_{\mathbf{sx}} - \mathbf{W}\mathbf{R}_{\mathbf{xx}}. \tag{2.63}$$

By setting (2.63) equal to $\mathbf{0}$ we obtain the *Wiener Hopf equation* (WHE)

$$\mathbf{W}\mathbf{R}_{\mathbf{xx}} = \mathbf{R}_{\mathbf{sx}}. \tag{2.64}$$

Solving for $\mathbf{W}$ also gives $\mathbf{W}^{\text{MMSE-s}}$ given in (2.61).

Note, $\mathbf{H}^{\text{MMSE-x}}$ and $\mathbf{W}^{\text{MMSE-s}}$ are both MMSE solutions, however, they are both optimal for two different cost functions (2.4) and (2.57), respectively. Therefore $\mathbf{H}^{\text{MMSE-s}} \triangleq \left[\mathbf{W}^{\text{MMSE-s}}\right]^{-1} = \mathbf{H}^{\text{MMSE-x}}$ does not hold in general. By using $\mathbf{R}_{\mathbf{xs}} = \mathbf{A}\mathbf{R}_{\mathbf{ss}}$, we see from (2.8) that $\mathbf{H}^{\text{MMSE-x}} = \mathbf{A}$. Thus $\mathbf{H}^{\text{MMSE-x}}$ is a bias-free estimate of $\mathbf{A}$. However, the same is not true for $\mathbf{W}^{\text{MMSE-s}}$ as we will see in the following. Starting with (2.61) and using $\mathbf{R}_{\mathbf{xx}} = \mathbf{A}\mathbf{R}_{\mathbf{ss}}\mathbf{A}^H + \mathbf{R}_{\mathbf{nn}}$ and $\mathbf{R}_{\mathbf{sn}} = \mathbf{R}_{\mathbf{ns}} = \mathbf{0}$ we obtain

$$\mathbf{H}^{\text{MMSE-s}} \triangleq \left[\mathbf{W}^{\text{MMSE-s}}\right]^{-1} = \left(\mathbf{A}\mathbf{R}_{\mathbf{ss}}\mathbf{A}^H + \mathbf{R}_{\mathbf{nn}}\right)\left(\mathbf{R}_{\mathbf{ss}}\mathbf{A}^H\right)^{-1} \tag{2.65}$$

$$= \left(\mathbf{A}\mathbf{R}_{\mathbf{ss}}\mathbf{A}^H + \mathbf{R}_{\mathbf{nn}}\right)\mathbf{A}^{-H}\mathbf{R}_{\mathbf{ss}}^{-1} \tag{2.66}$$

$$= \mathbf{A} + \mathbf{R}_{\mathbf{nn}}\mathbf{A}^{-H}\mathbf{R}_{\mathbf{ss}}^{-1} \tag{2.67}$$

and for the special case where $\mathbf{R}_{\mathbf{ss}} = \sigma_{\mathrm{s}}^2\,\mathbf{I}$ and $\mathbf{R}_{\mathbf{nn}} = \sigma_{\mathrm{n}}^2\,\mathbf{I}$ we have

$$\mathbf{H}^{\text{MMSE-s}} = \mathbf{A} + \frac{\sigma_{\mathrm{n}}^2}{\sigma_{\mathrm{s}}^2}\mathbf{A}^{-H}. \tag{2.68}$$

We see that $\mathbf{H}^{\text{MMSE-s}}$ is a biased estimate of $\mathbf{A}$ if sensor noise is present. Hence, $\mathbf{H}^{\text{MMSE-s}} = \mathbf{H}^{\text{MMSE-x}}$ is valid only in the noise-free case. As a consequence, the choice of the cost function is critical and has an impact on the estimate.

2.5.2 Batch learning

Since we usually do not know the true covariance matrices $\mathbf{R_{sx}}$ and $\mathbf{R_{xx}}$, we can replace them by estimates $\hat{\mathbf{R}}_{\mathbf{sx}}$ and $\hat{\mathbf{R}}_{\mathbf{xx}}$, respectively. We can then estimate the Wiener solution in (2.61) by

$$\mathbf{W} = \hat{\mathbf{R}}_{\mathbf{sx}} \hat{\mathbf{R}}_{\mathbf{xx}}^{-1} . \tag{2.69}$$

In a batch processing, we use all available measurement data of the system to estimate the correlation matrices

$$\hat{\mathbf{R}}_{\mathbf{sx}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{s}_t \mathbf{x}_t^H \tag{2.70}$$

$$\hat{\mathbf{R}}_{\mathbf{xx}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t \mathbf{x}_t^H . \tag{2.71}$$

The number of samples T needs to be large enough such that $\hat{\mathbf{R}}_{\mathbf{xx}}$ has full rank and is therefore invertible.

2.6 LMS-s

Analogously to $J_{\text{MSE-x}}$ in Section 2.3, we can formulate a steepest-descend algorithm which minimizes $J_{\text{MSE-s}}$ as

$$\begin{aligned} \mathbf{W}_{t+1} &= \mathbf{W}_t - \frac{\mu}{2} \nabla_{\mathbf{W}} J_{\text{MSE-s}} && (2.72) \\ &= \mathbf{W}_t + \mu \left(\mathbf{R_{sx}} - \mathbf{W}_t \mathbf{R_{xx}} \right) \\ &= \mathbf{W}_t + \mu \left(\mathbf{R_{sx}} - \mathbf{R_{ux}} \right) . && (2.73) \end{aligned}$$

The step size μ controls the stability and the rate of convergence of the algorithm.

2.6.1 Updating $\mathbf{W}$

LMS3-Ws By replacing $\mathbf{R_{sx}}$ and $\mathbf{R_{ux}}$ with their corresponding instantaneous estimates, $\mathbf{s}_t \mathbf{x}_t^H$ and $\mathbf{u}_t \mathbf{x}_t^H$, respectively, we obtain

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu \left(\mathbf{s} - \mathbf{u} \right) \mathbf{x}^H \tag{2.74}$$

which we refer to as LMS3-Ws. Note that Eq. (2.74) has great similarity with the form of (2.20), the LMS1-Hx algorithm. But it is the error criterion which makes the difference.

LMS4-Ws Alternatively, we can reformulate (2.73) as

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu \left(\mathbf{R_{ss}} \mathbf{A}^H - \mathbf{R_{ux}} \right) .$$

In the special case were the source signals are uncorrelated and of unity power, i.e. $\mathbf{R_{ss}} = \mathbf{I}$, and by replacing $\mathbf{R_{ux}}$ by its instantaneous estimate $\mathbf{u}\mathbf{x}^H$, we obtain

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu \left(\mathbf{A}^H - \mathbf{u}\mathbf{x}^H \right) . \tag{2.75}$$

This update form is not practicable as it requires knowledge of $\mathbf{A}$, the unknown system which we actually wish to identify. However, we know that near convergence $\mathbf{W}_t^{-1}$ is a fair approximation of $\mathbf{A}$ and we therefore can modify (2.75) as

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu \left(\mathbf{W}_t^{-H} - \mathbf{u}\mathbf{x}^H \right) \tag{2.76}$$

which we refer to as LMS4-Ws. The behavior of this algorithm might be difficult to predict, especially far away from convergence. Nevertheless, as we will see in Chapter 6, this algorithm has great similarity to its blind counterpart, namely the infomax algorithm proposed by Bell and Sejnowski [7]. Furthermore, the update (2.76) can be used for second-order blind decorrelation of instantaneous signal mixtures [31]. In the noiseless case, $\mathbf{W}_{t \to \infty} = \mathbf{Q}\mathbf{A}^{-1}$, where $\mathbf{Q}$ is a unitary matrix.

2.6.2 Updating $\mathbf{W}^{-1}$

Since $\mathbf{H} \triangleq \mathbf{W}^{-1}$, we can also transform an update equation for inverse modeling into an update algorithm for system identification.

LMS3a-Hs First we start with (2.74) and invert both sides

$$\begin{aligned} \mathbf{H}_{t+1} &= \left[\mathbf{W}_{t+1} \right]^{-1} = \left[\mathbf{W}_t + \mu \left(\mathbf{s} - \mathbf{u} \right) \mathbf{x}^H \right]^{-1} && (2.77) \\ &= \mathbf{H}_t - \mu \mathbf{H}_t \left(\mathbf{s} - \mathbf{u} \right) \left[1 + \mu \mathbf{x}^H \mathbf{H}_t \left(\mathbf{s} - \mathbf{u} \right) \right]^{-1} \mathbf{x}^H \mathbf{H}_t \\ &= \mathbf{H}_t + \mu \left(\mathbf{x} - \hat{\mathbf{x}} \right) \left[1 - \mu \mathbf{x}^H \left(\mathbf{x} - \hat{\mathbf{x}} \right) \right]^{-1} \mathbf{x}^H \mathbf{H}_t . && (2.78) \end{aligned}$$

Here we used the matrix-inversion lemma with $\mathbf{A}' = \mathbf{W}_t$, $\mathbf{B}' = \mu\,(\mathbf{s}-\mathbf{u})$, $\mathbf{C}' = 1$, and $\mathbf{D}' = \mathbf{x}^H$. We refer to (2.78) as LMS3a-Hs.

LMS3b-Hs We can start again with (2.77) but use the matrix-inversion lemma with $\mathbf{A}' = \mathbf{W}_t$, $\mathbf{B}' = \mu\,(\mathbf{s}-\mathbf{u})\,\mathbf{x}^H$, $\mathbf{C}' = \mathbf{I}$, and $\mathbf{D}' = \mathbf{I}$

$$\begin{aligned}\mathbf{H}_{t+1} &= \mathbf{H}_t - \mu\mathbf{H}_t\,(\mathbf{s}-\mathbf{u})\,\mathbf{x}^H\left[\mathbf{I}+\mu\mathbf{H}_t\,(\mathbf{s}-\mathbf{u})\,\mathbf{x}^H\right]^{-1}\mathbf{H}_t \\ &= \mathbf{H}_t + \mu\,(\mathbf{x}-\hat{\mathbf{x}})\,\mathbf{x}^H\left[\mathbf{I}-\mu\,(\mathbf{x}-\hat{\mathbf{x}})\,\mathbf{x}^H\right]^{-1}\mathbf{H}_t\,. \qquad (2.79)\end{aligned}$$

We refer to (2.79) as LMS3b-Hs.

LMS4b-Hs Starting with (2.75) we can apply the matrix-inversion lemma $\mathbf{A}' = \mathbf{W}_t$, $\mathbf{B}' = \mu\left(\mathbf{A}^H - \mathbf{u}\mathbf{x}^H\right)$, $\mathbf{C}' = \mathbf{I}$, and $\mathbf{D}' = \mathbf{I}$.

$$\begin{aligned}\mathbf{H}_{t+1} &= [\mathbf{W}_{t+1}]^{-1} = \left[\mathbf{W}_t + \mu\left(\mathbf{A}^H - \mathbf{u}\mathbf{x}^H\right)\right]^{-1} \qquad (2.80)\\ &= \mathbf{H}_t - \mu\mathbf{H}_t\left(\mathbf{A}^H - \mathbf{u}\mathbf{x}^H\right)\left[\mathbf{I}+\mu\mathbf{H}_t\left(\mathbf{A}^H - \mathbf{u}\mathbf{x}^H\right)\right]^{-1}\mathbf{H}_t \\ &= \mathbf{H}_t + \mu\left(\mathbf{x}\mathbf{x}^H - \mathbf{H}_t\mathbf{A}^H\right)\left[\mathbf{I}-\mu\left(\mathbf{x}\mathbf{x}^H - \mathbf{H}_t\mathbf{A}^H\right)\right]^{-1}\mathbf{H}_t \\ &= \mathbf{H}_t + \mu\left(\mathbf{x}\mathbf{x}^H - \mathbf{R}_{\hat{\mathbf{x}}\mathbf{x}}\right)\left[\mathbf{I}-\mu\left(\mathbf{x}\mathbf{x}^H - \mathbf{R}_{\hat{\mathbf{x}}\mathbf{x}}\right)\right]^{-1}\mathbf{H}_t \\ &= \mathbf{H}_t + \mu\,(\mathbf{x}-\hat{\mathbf{x}})\,\mathbf{x}^H\left[\mathbf{I}-\mu\,(\mathbf{x}-\hat{\mathbf{x}})\,\mathbf{x}^H\right]^{-1}\mathbf{H}_t \qquad (2.81)\end{aligned}$$

which is the same as LMS3b-Hs defined in (2.79). In this derivation we made the assumption that $\mathbf{R}_{\mathbf{ss}} = \mathbf{I}$, so that $\mathbf{H}_t\mathbf{A} = \mathbf{H}_t\mathbf{R}_{\mathbf{ss}}\mathbf{A} = \mathbf{R}_{\hat{\mathbf{x}}\mathbf{x}}$. Furthermore, we replaced $\mathbf{R}_{\hat{\mathbf{x}}\mathbf{x}}$ by its instantaneous estimate $\hat{\mathbf{x}}\mathbf{x}^H$. With these assumptions, LMS4b-Hs is mathemathically not exactly the same as LMS4-Ws anymore.

LMS4a-Hs We now start with (2.81) and apply the matrix-inversion lemma to the matrix inverse $\left[\mathbf{I}-\mu\,(\mathbf{x}-\hat{\mathbf{x}})\,\mathbf{x}^H\right]^{-1}$ with $\mathbf{A}' = \mathbf{I}$, $\mathbf{B}' = -\mu\left(\mathbf{x}-\mathbf{x}^H\right)$, $\mathbf{C}' = 1$, and $\mathbf{D}' = \mathbf{x}^H$. After some calculations we obtain

$$\mathbf{H}_{t+1} = \mathbf{H}_t + \frac{\mu}{1-\mu\mathbf{x}^H\,(\mathbf{x}-\hat{\mathbf{x}})}\,(\mathbf{x}-\hat{\mathbf{x}})\,\mathbf{x}^H\mathbf{H}_t \qquad (2.82)$$

which is the same as LMS3a-Hs defined in (2.78).

2.7 RLS-s

An alternative to the batch algorithm (2.69), where the correlation matrices $\mathbf{R}_{\mathbf{sx}}$ and $\mathbf{R}_{\mathbf{xx}}$ are estimated in (2.70) and (2.71) with the help of all available measurement data, respectively, is to estimate the correlation matrices recursively

$$\hat{\mathbf{R}}_{\mathbf{sx}_t} = \hat{\mathbf{R}}_{\mathbf{sx}_{t-1}} + \mathbf{s}_t\mathbf{x}_t^H \qquad (2.83)$$

$$\hat{\mathbf{R}}_{\mathbf{xx}_t} = \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}} + \mathbf{x}_t\mathbf{x}_t^H\,. \qquad (2.84)$$

We then obtain an online learning algorithm

$$\mathbf{W}_{t+1} = \hat{\mathbf{R}}_{\mathbf{sx}_t}\hat{\mathbf{R}}_{\mathbf{xx}_t}^{-1} \qquad (2.85)$$

which we will refer to as RLS2-s, as it is actually a recursive least squares (RLS) algorithm which minimizes the cost function

$$\tilde{J}_{\text{MSE-s}} = \sum_{\tau=0}^{t} \|\mathbf{s}_\tau - \mathbf{W}_t\,\mathbf{x}_\tau\|_2^2 \qquad (2.86)$$

at discrete time t.

2.7.1 RLS2-Ws

In (2.85) we have again a matrix inversion which has to be carried out at every time sample t. However, just as for the RLS1-Hx, by using the matrix-inversion lemma we can circumvent the cumbersome matrix inversion. To this end, we recursively update $\hat{\mathbf{R}}_{\mathbf{xx}_t}^{-1}$ instead of $\hat{\mathbf{R}}_{\mathbf{xx}_t}$. By inverting both sides of (2.84) and using the matrix-inversion lemma with $\mathbf{A}' = \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}$, $\mathbf{B}' = \mathbf{x}_t$, $\mathbf{C}' = 1$, and $\mathbf{D}' = \mathbf{x}_t^H$ we obtain

$$\begin{aligned}\hat{\mathbf{R}}_{\mathbf{xx}_t}^{-1} &= \left[\hat{\mathbf{R}}_{\mathbf{xx}_{t-1}} + \mathbf{x}_t\mathbf{x}_t^H\right]^{-1} \qquad (2.87)\\ &= \hat{\mathbf{R}}_{\mathbf{xx}}^{-1} - \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}\mathbf{x}\left[1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}\mathbf{x}\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{xx}}^{-1} \qquad (2.88)\\ &= \hat{\mathbf{R}}_{\mathbf{xx}}^{-1} - \tilde{\mu}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}\mathbf{x}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{xx}}^{-1} \qquad (2.89)\\ \tilde{\mu} &= \frac{1}{1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}\mathbf{x}} \qquad (2.90)\end{aligned}$$

where $\tilde{\mu}$ can be seen as a step size of the update. Next we use (2.88) to modify (2.85) and find an efficient update for $\mathbf{W}_{t+1}$

$$\begin{aligned}
\mathbf{W}_{t+1} &= \hat{\mathbf{R}}_{\mathbf{s}\mathbf{x}_t}\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_t}^{-1} \qquad (2.91)\\
&= \left(\hat{\mathbf{R}}_{\mathbf{s}\mathbf{x}_{t-1}} + \mathbf{s}_t\mathbf{x}_t^H\right)\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_t}^{-1}\\
&= \left(\hat{\mathbf{R}}_{\mathbf{s}\mathbf{x}} + \mathbf{s}\mathbf{x}^H\right)\left(\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1} - \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\left[1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\right)\\
&= \mathbf{W}_t - \mathbf{W}_t\mathbf{x}\left[1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1} + \mathbf{s}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\\
&\quad - \mathbf{s}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\left[1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\\
&= \mathbf{W}_t - \mathbf{W}_t\mathbf{x}\left[1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\\
&\quad + \mathbf{s}\left[1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\right]\left[1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\\
&\quad - \mathbf{s}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\left[1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\\
&= \mathbf{W}_t + \frac{1}{1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}}\left(\mathbf{s}-\mathbf{W}_t\mathbf{x}\right)\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1} \qquad (2.92)\\
&= \mathbf{W}_t + \tilde{\mu}\left(\mathbf{s}-\mathbf{u}\right)\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1} \qquad (2.93)
\end{aligned}$$

where $\mathbf{e}_{\mathrm{s}} = \mathbf{s} - \mathbf{u} = \mathbf{s} - \mathbf{W}_t\mathbf{x}$ is the current estimation-error vector. In the above derivation we sometimes omitted the time-sample index for $\mathbf{s}_t$, $\mathbf{x}_t$, $\mathbf{u}_t$, $\hat{\mathbf{R}}_{\mathbf{s}\mathbf{x}_{t-1}}$, and $\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}$. We further used the fact that $\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{x}$ is a scalar.

2.7.2 RLS2-Hs

In case we want to update $\mathbf{H}_{t+1}$ instead of $\mathbf{W}_{t+1}$, we can invert both sides of (2.85) and use $\mathbf{H}_{t+1} \triangleq \mathbf{W}_{t+1}^{-1}$

$$\mathbf{H}_{t+1} = \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_t}\hat{\mathbf{R}}_{\mathbf{s}\mathbf{x}_t}^{-1}. \qquad (2.94)$$

However, just as in (2.85), we again need a matrix inversion for every update. To avoid this, we start by inverting both sides of (2.93)

$$\mathbf{H}_{t+1} = \left[\mathbf{H}_t^{-1} + \tilde{\mu}\left(\mathbf{s}-\mathbf{u}\right)\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\right]^{-1}. \qquad (2.95)$$

Applying the matrix-inversion lemma with $\mathbf{A}' = \mathbf{H}_t^{-1}$, $\mathbf{B}' = \tilde{\mu}\left(\mathbf{s}-\mathbf{u}\right)$, $\mathbf{C}' = 1$, and $\mathbf{D}' = \mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}$ gives

$$\begin{aligned}
\mathbf{H}_{t+1} &= \mathbf{H}_t - \tilde{\mu}\mathbf{H}_t\left(\mathbf{s}-\mathbf{u}\right)\left[1+\tilde{\mu}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{H}_t\left(\mathbf{s}-\mathbf{u}\right)\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{H}_t\\
&= \mathbf{H}_t + \tilde{\mu}\left(\mathbf{x}-\hat{\mathbf{x}}\right)\left[1-\tilde{\mu}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\left(\mathbf{x}-\hat{\mathbf{x}}\right)\right]^{-1}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{H}_t\\
&= \mathbf{H}_t + \mu\left(\mathbf{x}-\hat{\mathbf{x}}\right)\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{H}_t \qquad (2.96)
\end{aligned}$$

with

$$\mu = \frac{\tilde{\mu}}{1-\tilde{\mu}\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\left(\mathbf{x}-\hat{\mathbf{x}}\right)} = \frac{1}{1+\mathbf{x}^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}\hat{\mathbf{x}}} \qquad (2.97)$$

where $\tilde{\mu}$ is defined in (2.90). In the above derivation we used $\hat{\mathbf{x}} = \mathbf{H}_t\mathbf{s}$. Note that $\mathbf{e}_{\mathrm{x}} \triangleq \mathbf{x} - \hat{\mathbf{x}}$ is also an estimation error, however, in general not the same as $\mathbf{e}_{\mathrm{s}}$.

2.7.3 RLS2-Ws with exponential forgetting

To track time-varying systems, the cost function defined in (2.86) is extended by an exponential weighting of the past measurements

$$\tilde{J}_{\text{MSE-s}} = \sum_{\tau=0}^{t}\lambda^{t-\tau}\left\|\mathbf{s}_\tau - \mathbf{W}_t\,\mathbf{x}_\tau\right\|_2^2. \qquad (2.98)$$

This causes exponential forgetting in the recursive estimates of the correlation matrices

$$\hat{\mathbf{R}}_{\mathbf{s}\mathbf{x}_t} = \lambda\hat{\mathbf{R}}_{\mathbf{s}\mathbf{x}_{t-1}} + (1-\lambda)\mathbf{s}_t\mathbf{x}_t^H \qquad (2.99)$$

$$\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_t} = \lambda\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}} + (1-\lambda)\mathbf{x}_t\mathbf{x}_t^H \qquad (2.100)$$

where λ denotes a forgetting factor with $1 > \lambda > 0$. Doing similar calculations as in Section 2.7.1 yields the same update equation for $\mathbf{W}_{t+1}$ but with a different step size $\tilde{\mu}$

$$\tilde{\mu}_t = \frac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{x}_t^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1}\mathbf{x}_t} \qquad (2.101)$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \tilde{\mu}_t\left(\mathbf{s}_t - \mathbf{u}_t\right)\mathbf{x}_t^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1} \qquad (2.102)$$

$$\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1} - \tilde{\mu}_t\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1}\mathbf{x}_t\mathbf{x}_t^H\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1}\right) \qquad (2.103)$$

where (2.101), (2.102), and (2.103) are referred to as RLS2-Ws.

2.7.4 RLS2-Hs with exponential forgetting

Using exponential forgetting and doing similar calculations as in Section 2.7.2 yields the same update equation for $\mathbf{H}_{t+1}$ but with different step sizes μ and $\tilde{\mu}$

$$\mu_t = \frac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1} \hat{\mathbf{x}}_t} \tag{2.104}$$

$$\mathbf{H}_{t+1} = \mathbf{H}_t + \mu_t\,(\mathbf{x}_t - \hat{\mathbf{x}}_t)\,\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1} \mathbf{H}_t \tag{2.105}$$

$$\tilde{\mu}_t = \frac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1} \mathbf{x}_t} \tag{2.106}$$

$$\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1} - \tilde{\mu}_t \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1} \mathbf{x}_t \mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}_{t-1}}^{-1}\right) \tag{2.107}$$

where (2.104), (2.105), (2.106), and (2.107) are referred to as RLS2-Hs.

2.8 MMSE—minimum mean-squared error

cost function	$\mathbf{H} = \hat{\mathbf{A}}$		$\mathbf{W} = \hat{\mathbf{A}}^{-1}$
$J_{\text{MSE-x}}$	$\mathbf{H}^{\text{MMSE-x}} = \mathbf{R}_{\mathbf{xs}}\mathbf{R}_{\mathbf{ss}}^{-1}$	$\longrightarrow$	$\mathbf{W}^{\text{MMSE-x}} \triangleq \mathbf{H}^{\text{MMSE-x}\,-1}$
$J_{\text{MSE-s}}$	$\mathbf{H}^{\text{MMSE-s}} \triangleq \mathbf{W}^{\text{MMSE-s}\,-1}$	$\longleftarrow$	$\mathbf{W}^{\text{MMSE-s}} = \mathbf{R}_{\mathbf{sx}}\mathbf{R}_{\mathbf{xx}}^{-1}$

Table 2.1: Relationship of Wiener solutions.

We now wish to evaluate the performances $J_{\text{MSE-x}}$ and $J_{\text{MSE-s}}$ as defined in (2.4) and (2.57), respectively, which we achieve with the Wiener solutions $\mathbf{H}^{\text{MMSE-x}}$ and $\mathbf{W}^{\text{MMSE-s}}$. To this end, we insert $\mathbf{H}^{\text{MMSE-x}}$, $\mathbf{W}^{\text{MMSE-x}}$, $\mathbf{W}^{\text{MMSE-s}}$, and $\mathbf{H}^{\text{MMSE-s}}$ (see Table 2.1) into $\mathbf{R}_{\mathbf{e_x}\mathbf{e_x}}$ defined in (2.6) and $\mathbf{R}_{\mathbf{e_s}\mathbf{e_s}}$ defined in (2.59). Since the mixing system is time invariant, we have $\mathbf{R}_{\mathbf{sx}} = \mathbf{R}_{\mathbf{ss}}\mathbf{A}^H$, $\mathbf{R}_{\mathbf{xs}} = \mathbf{A}\mathbf{R}_{\mathbf{ss}}$, and $\mathbf{R}_{\mathbf{xx}} = \mathbf{A}\mathbf{R}_{\mathbf{ss}}\mathbf{A}^H + \mathbf{R}_{\mathbf{nn}}$. Furthermore we use $\mathbf{R}_{\mathbf{ss}} = \sigma_{\text{s}}^2\mathbf{I}$ and $\mathbf{R}_{\mathbf{nn}} = \sigma_{\text{n}}^2\mathbf{I}$ for the evaluation of $J_{\text{MSE-x}}$ and $J_{\text{MSE-s}}$.

By doing so, we obtain after longer calculations (with the help of the matrix-inversion lemma)

$$\mathbf{R}_{\mathbf{e_x}\mathbf{e_x}}\left(\mathbf{H}^{\text{MMSE-x}}\right) = \mathbf{R}_{\mathbf{nn}} \tag{2.108}$$

$$J_{\text{MSE-x}}\left(\mathbf{H}^{\text{MMSE-x}}\right) = M\sigma_{\text{n}}^2 \tag{2.109}$$

$$\mathbf{R}_{\mathbf{e_s}\mathbf{e_s}}\left(\mathbf{W}^{\text{MMSE-x}}\right) = \mathbf{A}^{-1}\mathbf{R}_{\mathbf{nn}}\mathbf{A}^{-H} \tag{2.110}$$

$$J_{\text{MSE-s}}\left(\mathbf{W}^{\text{MMSE-x}}\right) = \sigma_{\text{n}}^2\|\mathbf{A}^{-1}\|_F^2 \tag{2.111}$$

$$\mathbf{R}_{\mathbf{e_x}\mathbf{e_x}}\left(\mathbf{H}^{\text{MMSE-s}}\right) = \mathbf{R}_{\mathbf{nn}}\left(\mathbf{I} + \left[\mathbf{A}\mathbf{R}_{\mathbf{ss}}\mathbf{A}^H\right]^{-1}\mathbf{R}_{\mathbf{nn}}\right) \tag{2.112}$$

$$J_{\text{MSE-x}}\left(\mathbf{H}^{\text{MMSE-s}}\right) = \sigma_{\text{n}}^2\left(M + \frac{\sigma_{\text{n}}^2}{\sigma_{\text{s}}^2}\|\mathbf{A}^{-1}\|_F^2\right) \tag{2.113}$$

$$\mathbf{R}_{\mathbf{e_s}\mathbf{e_s}}\left(\mathbf{W}^{\text{MMSE-s}}\right) = \left[\mathbf{R}_{\mathbf{ss}}^{-1} + \mathbf{A}^H\mathbf{R}_{\mathbf{nn}}^{-1}\mathbf{A}\right]^{-1} \tag{2.114}$$

$$J_{\text{MSE-s}}\left(\mathbf{W}^{\text{MMSE-s}}\right) = \sigma_{\text{n}}^2\,\text{tr}\left(\left[\mathbf{A}^H\mathbf{A} + \frac{\sigma_{\text{n}}^2}{\sigma_{\text{s}}^2}\mathbf{I}\right]^{-1}\right). \tag{2.115}$$

Note, the values in (2.109) and (2.115) also represent the lowest achievable values for $J_{\text{MSE-x}}$ and $J_{\text{MSE-s}}$, respectively, namely the *minimum mean-squared error*. The special case where $\mathbf{A}$ is unitary results in

$$J_{\text{MSE-x}}\left(\mathbf{H}^{\text{MMSE-x}}\right) = M\sigma_{\text{n}}^2 \tag{2.116}$$

$$J_{\text{MSE-s}}\left(\mathbf{W}^{\text{MMSE-x}}\right) = M\sigma_{\text{n}}^2 \tag{2.117}$$

$$J_{\text{MSE-x}}\left(\mathbf{H}^{\text{MMSE-s}}\right) = M\sigma_{\text{n}}^2\left(1 + \frac{\sigma_{\text{n}}^2}{\sigma_{\text{s}}^2}\right) \tag{2.118}$$

$$J_{\text{MSE-s}}\left(\mathbf{W}^{\text{MMSE-s}}\right) = M\sigma_{\text{n}}^2\left(1 + \frac{\sigma_{\text{n}}^2}{\sigma_{\text{s}}^2}\right)^{-1}. \tag{2.119}$$

2.9 Block-wise update

Instead of updating $\mathbf{H}$ or $\mathbf{W}$ at every time sample, we could do so after every block of L samples. The update equations are thereby modified such that *inner and outer products are replaced by their average over a whole block of L samples*. For illustration we give two examples: LMS1-Hx in (2.20) becomes

$$\mathbf{H}_{t+L} = \mathbf{H}_t + \mu\frac{1}{L}\sum_{l=0}^{L-1}(\mathbf{x}_{t+l} - \hat{\mathbf{x}}_{t+l})\,\mathbf{s}_{t+l}^H \tag{2.120}$$

and LMS1a-Wx defined in (2.24) becomes

$$\mathbf{W}_{t+L} = \mathbf{W}_t + \frac{\mu}{L - \mu \sum_{l=0}^{L-1} \mathbf{s}_{t+l}^H (\mathbf{s}_{t+l} - \mathbf{u}_{t+l})} \sum_{l=0}^{L-1} (\mathbf{s}_{t+l} - \mathbf{u}_{t+l})\, \mathbf{s}_{t+l}^H \mathbf{W}_t \,. \tag{2.121}$$

The block-wise update equations of the other LMS-type algorithms are derived similarly. Note, usually a block-wise update is performed in a different way, namely

$$\mathbf{W}_{t+L} = \mathbf{W}_t + \frac{\mu}{L} \sum_{l=0}^{L-1} \Delta \mathbf{W}_{t+l} \,. \tag{2.122}$$

2.10 Simulation results

2.10.1 Performance of LMS algorithms

First we want to examine the performance of the LMS-type algorithms. We use the mixing model with additive noise as defined in (2.1). The sensor noise is mutually independent, $\mathbf{R}_{\mathbf{nn}} = \sigma_{\mathrm{n}}^2 \mathbf{I}$, and $\sigma_{\mathrm{n}} = 0.01$ (-40 dB). The $M \times M$ dimensional mixing matrix $\mathbf{A}$ has real-valued entries with $M = 10$. The source signals are white Gaussian noise sequences, mutually independent, and of unity power, $\mathbf{R}_{\mathbf{ss}} = \mathbf{I}$.

We analyze the behavior with two different mixing matrices $\mathbf{A}$, one is unitary and one is ill-conditioned. By definition the unitary mixing matrix has $\chi(\mathbf{A}) \triangleq \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^{-1}\|_2 = \sigma_1/\sigma_M = 1$, whereas the ill-conditioned $\mathbf{A}$ has logarithmically distributed singular values and condition number $\chi(\mathbf{A}) = 10$. The largest singular value for both mixing matrices is $\|\mathbf{A}\|_2 \triangleq \sigma_1 = 1$. Initially we set $\mathbf{H}_0 = \mathbf{W}_0 = \mathbf{I}$. All plots are averages over 30 independent runs. The simulation setup is illustrated in Fig. 2.3.

Sample-wise update $L = 1$

In the first simulation we analyze the convergence rate of the LMS algorithms based on a sample-wise update ($L = 1$) of $\mathbf{H}$ and $\mathbf{W}$. The learning curves are

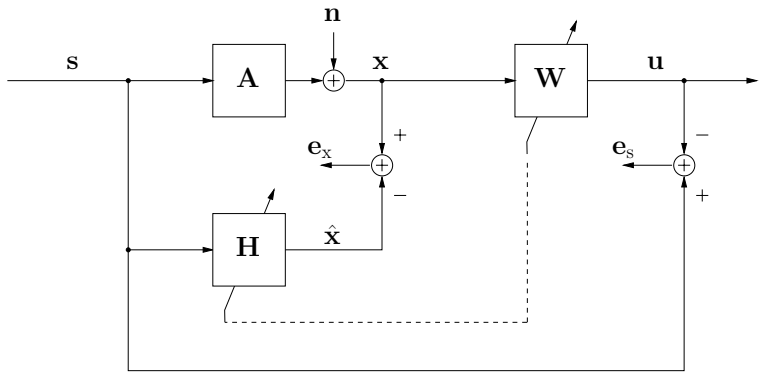

Figure 2.3: System identification and inverse modeling (inverse-system identification). Either of the error signals, $\mathbf{e}_{\mathrm{x}}$ or $\mathbf{e}_{\mathrm{s}}$, can be used for the adaptation. However, the performance behavior can strongly depend on the choice of the error criterion. During the adaptation we have constrained $\mathbf{W} = \mathbf{H}^{-1}$, and vice versa.

averages over 30 runs and shown in Fig. 2.4 and Fig. 2.5. The corresponding step sizes μ are listed in Table 2.2. We have the following remarks:

- For $L = 1$ we can subdivide the algorithms into three classes, where the algorithms within one class are algebraically equivalent

  (a) LMS1-Hx, LMS1a-Wx, and LMS1b-Wx,

  (b) LMS2-Hx, LMS2a-Wx, and LMS2b-Wx,

  (c) LMS3a-Hs, LMS3b-Hs, LMS3-Ws, LMS4a-Hs, and LMS4b-Hs.

  Thus, only one learning curve is shown for each class.

- For the unitary mixing matrix the algorithms from class (a) and (c) are dual in the sense that the behavior of $J_{\text{MSE-x}}$ and $J_{\text{MSE-s}}$ for class (a) is similar to the behavior of $J_{\text{MSE-s}}$ and $J_{\text{MSE-x}}$ for class (c), respectively.

- For the algorithms of class (b) there are two learning curves depicted, which stem from using different step sizes μ. This is to demonstrate that the algorithm is stable, but has a very high steady-state error level.

Plot	Algorithm	$\chi(\mathbf{A})=1$ μ	$\chi(\mathbf{A})=10$ μ	Comments
		Fig. 2.4	Fig. 2.5	
(a)	LMS1-Hx	0.08	0.08	
(b)	LMS2-Hx	0.08 / 0.01	0.08 / 0.01	very high misadjustment
(c)	LMS3a-Hs LMS4a-Hs	0.08	0.2	
(c)	LMS3b-Hs LMS4b-Hs	0.08	0.2	fast if $\mathbf{A}$ is unitary
		Fig. 2.4	Fig. 2.5	
(a)	LMS1a-Wx	0.08	0.08	
(a)	LMS1b-Wx	0.08	0.08	
(b)	LMS2a-Wx	0.08 / 0.01	0.08 / 0.01	very high misadjustment
(b)	LMS2b-Wx	0.08 / 0.01	0.08 / 0.01	very high misadjustment
(c)	LMS3-Ws	0.08	0.2	
	LMS4-Ws	-	-	no convergence, unstable

Table 2.2: Step sizes μ for the simulations of Fig. 2.4 and Fig. 2.5 with block length $L=1$. The step sizes are chosen such as to achieve the fastest initial convergence rate without becoming unstable. The condition numbers of the unknown mixing matrices are $\chi(\mathbf{A})=1$ and $\chi(\mathbf{A})=10$. The update equations of the algorithms are given in Table E.3 and E.7 in Appendix E.

- The convergence rate of the algorithms in class (a) are robust against the eigenvalue spread of $\mathbf{A}$ and $\mathbf{R}_{\mathbf{xx}}$. It rather depends on the eigenvalue spread of $\mathbf{R}_{\mathbf{ss}}$, which is equal to unity in this case.
- LMS4-Ws does not converge for $\mathbf{W}_0=\mathbf{I}$ to $\mathbf{A}^{-1}$. Even using $\mathbf{W}_0=\mathbf{A}^{-1}$ caused the algorithm to drift away from this point, although slowly. However, $\mathbf{W}_t\mathbf{A}$ was adapted to a time-varying unitary matrix $\mathbf{Q}_t$.
- The final steady-state error level of $\frac{1}{M}J_{\text{MSE-x}}$ is equal to the sensor noise level (-40 dB), irrespectively of $\chi(\mathbf{A})$. This is not true for $\frac{1}{M}J_{\text{MSE-s}}$. There the steady-state error level depends strongly on $\chi(\mathbf{A})$, as for a large $\chi(\mathbf{A})$ the sensor noise appears strongly amplified at the output. See also the theoretical analysis in Section 2.8.
- The algorithms of class (a) show from the beginning a monotonic decay of $J_{\text{MSE-x}}$, in contrast to $J_{\text{MSE-s}}$, which has an initial peak. For the algorithms of class (c) it is just the other way round.

Plot	Algorithm	$\chi(\mathbf{A})=1$ μ	$\chi(\mathbf{A})=10$ μ	Comments
		Fig. 2.6	Fig. 2.7	
(a)	LMS1-Hx	1.0	0.8	
(b)	LMS2-Hx	0.5	0.5	very high misadjustment
(c)	LMS3a-Hs LMS4a-Hs	0.03	0.1	
(d)	LMS3b-Hs LMS4b-Hs	0.8	1.2	fast if $\mathbf{A}$ is unitary
		Fig. 2.8	Fig. 2.9	
(a)	LMS1a-Wx	0.03	0.03	
(b)	LMS1b-Wx	1.0	1.0	
(c)	LMS2a-Wx	0.03	0.03	very high misadjustment
(d)	LMS2b-Wx	0.3	0.3	very high misadjustment
(e)	LMS3-Ws	0.9	0.9	
	LMS4-Ws	-	-	no convergence, unstable

Table 2.3: Step sizes μ for the simulations of Fig. 2.6, Fig. 2.7, Fig. 2.8, and Fig. 2.9 with block length $L=30$. The step sizes are chosen such as to achieve the fastest initial convergence rate without becoming unstable. The condition numbers of the unknown mixing matrices are $\chi(\mathbf{A})=1$ and $\chi(\mathbf{A})=10$. The update equations of the algorithms are given in Table E.3 and E.7 in Appendix E.

- The step-size normalization in LMS1a-Wx, LMS3a-Hs, and LMS4a-Hs has a stabilizing effect on the adaptation, if μ is chosen such that the algorithm has a fast initial convergence rate. However, for small step sizes μ the step-size normalization can also be neglected. We then derive

$$\mathbf{W}_{t+1}=\mathbf{W}_t+\mu\,(\mathbf{s}-\mathbf{u})\,\mathbf{s}^H\mathbf{W}_t \tag{2.123}$$

which we refer to as LMS1c-Wx and

$$\mathbf{H}_{t+1}=\mathbf{H}_t+\mu\,(\mathbf{x}-\hat{\mathbf{x}})\,\mathbf{x}^H\mathbf{H}_t \tag{2.124}$$

which we refer to as LMS3c-Hs and LMS4c-Hs.

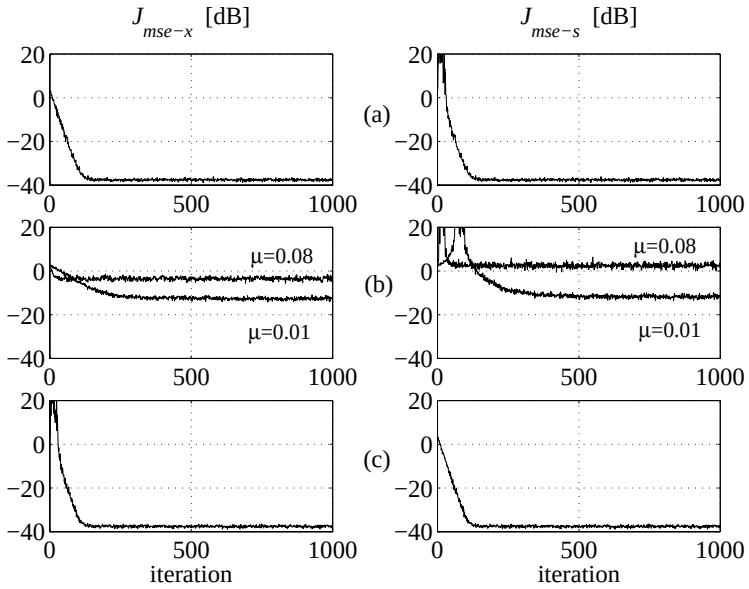

Figure 2.4: **Unitary mixing matrix**, $\chi(\mathbf{A}) = 1$, $L = 1$.
Performance $\frac{1}{M} J_{\text{MSE-x}}$ (left column) and $\frac{1}{M} J_{\text{MSE-s}}$ (right column) of the following algorithms:
(a) LMS1-Hx, LMS1a-Wx, LMS1b-Wx,
(b) LMS2-Hx, LMS2a-Wx, LMS2b-Wx,
(c) LMS3a-Hs, LMS3b-Hs, LMS3-Ws, LMS4a-Hs, LMS4b-Hs.
The update is done sample by sample. The curves are averages over 30 runs. The step sizes μ are given in Table 2.2.

Block-wise update $L = 30$

In the second simulation we compare the convergence rate of the LMS algorithms based on a block-wise update with block length $L = 30$. The learning curves are averages over 30 runs and shown in Figs. 2.6–2.9. The corresponding step sizes μ are listed in Table 2.3 We make the following observations:

- The three classes obtained with $L = 1$ do no longer hold for a block-wise update of $\mathbf{H}$ and $\mathbf{W}$ with $L > 1$.
- The algorithms which are derived from minimizing $J_{\text{MSE-x}}$ are robust

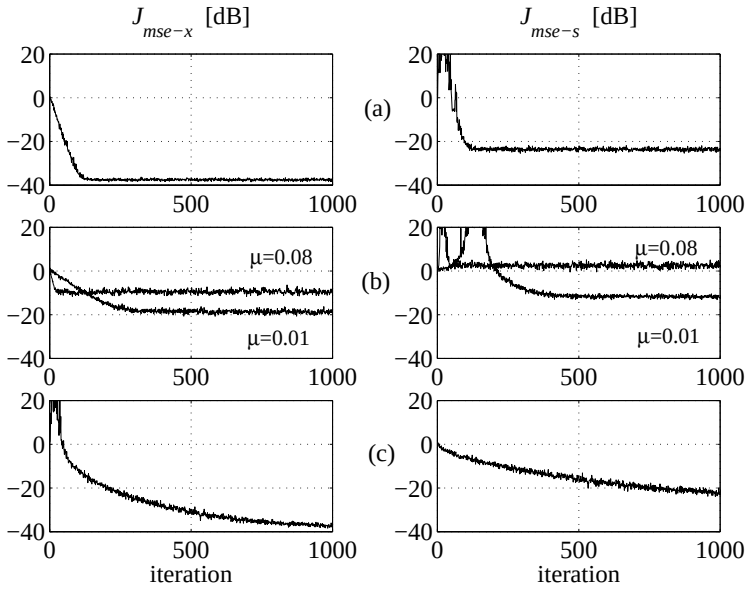

Figure 2.5: **Ill-conditioned mixing matrix**, $\chi(\mathbf{A}) = 10$, $L = 1$.
Performance $\frac{1}{M} J_{\text{MSE-x}}$ (left column) and $\frac{1}{M} J_{\text{MSE-s}}$ (right column) of the following algorithms:
(a) LMS1-Hx, LMS1a-Wx, LMS1b-Wx,
(b) LMS2-Hx, LMS2a-Wx, LMS2b-Wx,
(c) LMS3a-Hs, LMS3b-Hs, LMS3-Ws, LMS4a-Hs, LMS4b-Hs.
The update is done sample by sample. The curves are averages over 30 runs. The step sizes μ are given in Table 2.2.

against the eigenvalue spread of $\mathbf{A}$ and $\mathbf{R}_{\mathbf{xx}}$. Their convergence depends on $\chi(\mathbf{R}_{\mathbf{ss}})$, which is equal to one in this case. This observation is not true for the algorithms which aim at minimizing $J_{\text{MSE-s}}$.

- The final value for $J_{\text{MSE-s}}$ is higher for the ill-conditioned case. See also the theoretical analysis in Section 2.8.
- For $L = 1$, the two algorithms LMS1a-Wx and LMS1b-Wx which both are algebraically equivalent to LMS1-Hx, show quite a different convergence behavior for $L = 30$. The same is true for LMS2a-Wx and LMS2b-Wx, LMS3a-Hs and LMS3b-Hs, and LMS4a-Hs and LMS4b-Hs, respectively.

- LMS4-Ws does not converge. This is not very surprising, as the same phenomenon was also seen for $L=1$.

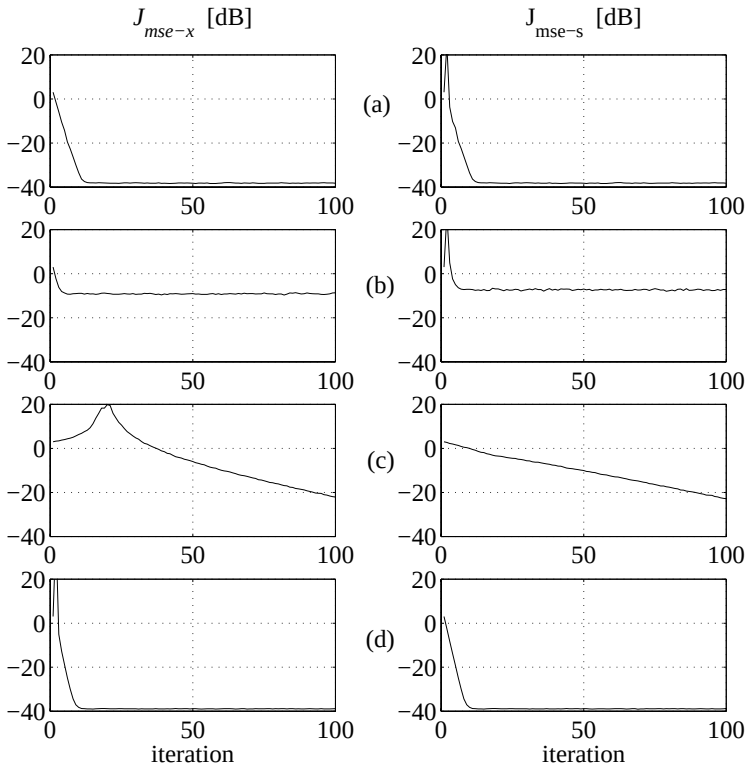

Figure 2.6: **Unitary mixing matrix**, $\chi(\mathbf{A})=1$, $L=30$.
Performance $\frac{1}{M}J_{\text{MSE-x}}$ (left column) and $\frac{1}{M}J_{\text{MSE-s}}$ (right column) of the following algorithms:
(a) LMS1-Hx,
(b) LMS2-Hx,
(c) LMS3a-Hs, LMS4a-Hs,
(d) LMS3b-Hs, LMS4b-Hs.
The update is done block wise. The curves are averages over 30 runs. The step sizes μ are given in Table 2.3.

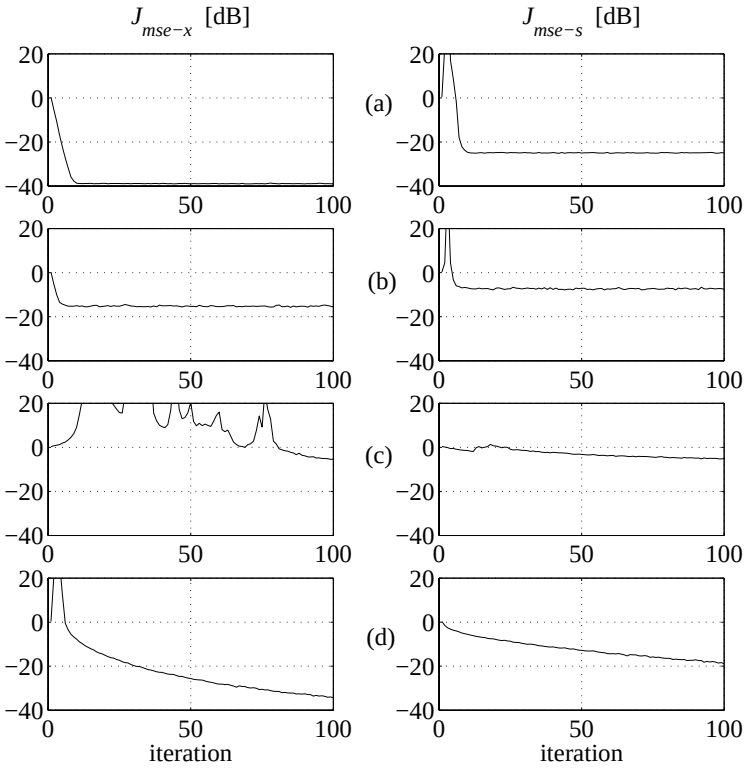

Figure 2.7: **Ill-conditioned mixing matrix**, $\chi(\mathbf{A})=10$, $L=30$.
Performance $\frac{1}{M}J_{\text{MSE-x}}$ (left column) and $\frac{1}{M}J_{\text{MSE-s}}$ (right column) of the following algorithms:
(a) LMS1-Hx,
(b) LMS2-Hx,
(c) LMS3a-Hs, LMS4a-Hs,
(d) LMS3b-Hs, LMS4b-Hs.
The update is done block wise. The curves are averages over 30 runs. The step sizes μ are given in Table 2.3.

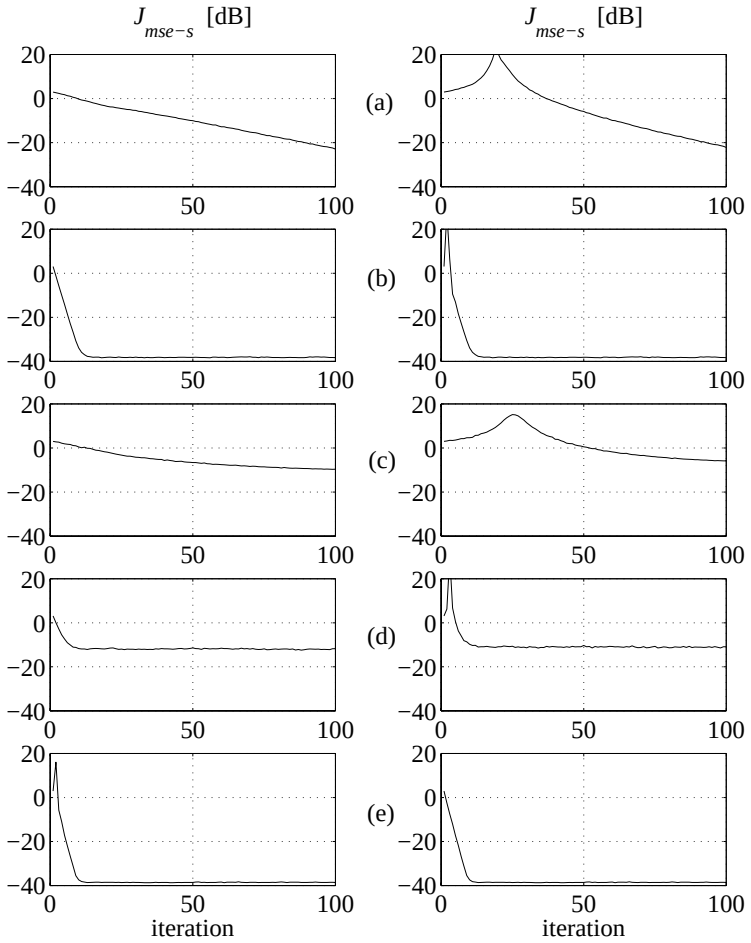

Figure 2.8: **Unitary mixing matrix**, $\chi(\mathbf{A}) = 1$, $L = 30$.
Performance $\frac{1}{M} J_{\text{MSE-x}}$ (left column) and $\frac{1}{M} J_{\text{MSE-s}}$ (right column) of the following algorithms:
(a) LMS1a-Wx,
(b) LMS1b-Wx,
(c) LMS2a-Wx,
(d) LMS2b-Wx,
(e) LMS3-Ws.
The update is done block wise. The curves are averages over 30 runs. The step sizes μ are given in Table 2.3.

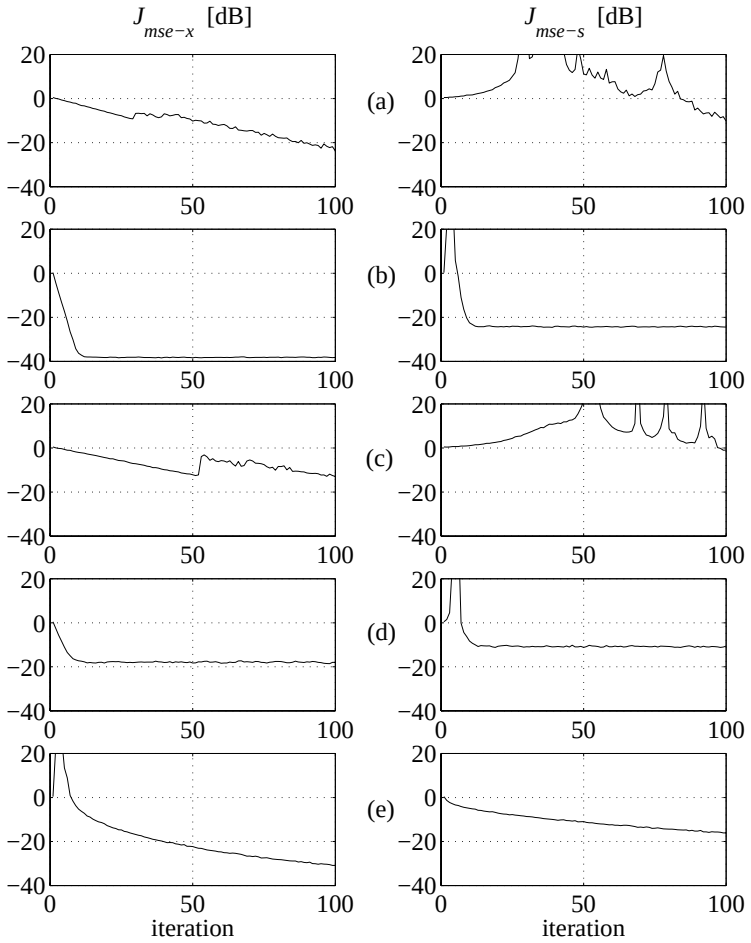

Figure 2.9: **Ill-conditioned mixing matrix**, $\chi(\mathbf{A}) = 10$, $L = 30$.
Performance $\frac{1}{M} J_{\text{MSE-x}}$ (left column) and $\frac{1}{M} J_{\text{MSE-s}}$ (right column) of the following algorithms:
(a) LMS1a-Wx,
(b) LMS1b-Wx,
(c) LMS2a-Wx,
(d) LMS2b-Wx,
(e) LMS3-Ws.
The update is done block wise. The curves are averages over 30 runs. The step sizes μ are given in Table 2.3.

2.10.2 Performance of RLS algorithms

Now we want to investigate the performance of the RLS-type algorithms. We use again the mixing model with additive noise as defined in (2.1). The sensor noise is mutually independent, $\mathbf{R}_{\mathbf{nn}} = \sigma_{\mathrm{n}}^2\mathbf{I}$, and $\sigma_{\mathrm{n}} = 0.01$ (-40 dB). The $M \times M$ dimensional mixing matrix $\mathbf{A}$ has real-valued entries with $M = 10$. The source signals are white Gaussian noise sequences, mutually independent, and with unity power, $\mathbf{R}_{\mathbf{ss}} = \mathbf{I}$.

We use three different mixing matrices with $\chi(\mathbf{A}) = 1, 10, 100$, respectively, logarithmically distributed singular values, where applicable, and $\sigma_1 \triangleq \|\mathbf{A}\|_2 = 1$. The forgetting factor is $\lambda = 0.97$, the block size $L = 1$. The initial conditions for the adaptation are $\mathbf{H}_0 = \mathbf{W}_0 = \mathbf{I}$, and $\hat{\mathbf{R}}_{\mathbf{ss}_0} = \hat{\mathbf{R}}_{\mathbf{xx}_0} = 0.001 \cdot \mathbf{I}$. The learning curves are averages over 30 runs and shown in Fig. 2.10 and Fig. 2.11. We make the following observations:

- RLS1-Hx and RLS1-Wx have the same performance, as they are algebraically equal. The same is true for RLS2-Hs and RLS2-Ws. Differences in the convergence behavior are due to numerical inaccuracies, especially for the case of large eigenvalue spread of the mixing matrix.
- The lower bound for $J_{\text{MSE-s}}$, given in (2.115), is for this simulation setup -40 dB, -26.08 dB, and -10.2 dB for $\chi(\mathbf{A}) = 1, 10$, and 100, respectively. As seen from the performance curves, these values are almost achieved by RLS1-x and RLS2-s. Note that regardless of the algorithm, $J_{\text{MSE-s}}$ cannot become arbitrary close to the sensor noise level for a non-unitary mixing system.
- Additional simulations have shown that reducing the sensor noise from -40 dB to -60 dB causes a further decrease of the final $J_{\text{MSE-x}}$ and $J_{\text{MSE-s}}$ by approximately another 20 dB. This coincides with the theoretical analysis from Section 2.8 for a small σ_{n}.

2.11 Summary

In this chapter we have developed several LMS- and RLS-type algorithms for system identification and inverse modeling of an instantaneous mixing system. We thereby have used two different error criteria, $J_{\text{MSE-x}}$, which measures the output estimation or output prediction error $\mathbf{x} - \hat{\mathbf{x}}$ of the unknown system, and $J_{\text{MSE-s}}$, which actually measures the input estimation error $\mathbf{s} - \hat{\mathbf{s}}$ of the unknown system. Independent from the cost function, we can either update $\mathbf{H}$ or $\mathbf{W} \triangleq \mathbf{H}^{-1}$, see Table 2.4. Thereby the matrix-inversion lemma has been shown to be a very powerful tool to transform an algorithm for system identification into a corresponding algorithm for inverse modeling. The relationship between the derived algorithms is shown Table E.1 in Appendix E. The derived update equations are summarized in Table E.3 and E.7.

From the simulation results we have also seen which algorithms are robust against an eigenvalue spread of the unknown mixing matrix $\mathbf{A}$. In Chapter 5 we extend these algorithms to make them applicable for general multichannel system identification and inverse modeling, where the elements of the mixing matrix are filter polynomials. Furthermore, in Chapter 6 we will transform these *non-blind* algorithms into *blind* algorithms by exchanging the non-blind error criteria with a blind error criteria. Since we know the performance ranking of the non-blind algorithms, we expect that their blind counterparts will exhibit a similar performance ranking within the group of blind algorithms.

cost function	$\mathbf{H} = \hat{\mathbf{A}}$		$\mathbf{W} = \hat{\mathbf{A}}^{-1}$
$J_{\text{MSE-x}}$	$\mathbf{H}_t$	$\longrightarrow$	$\mathbf{W}_t = \mathbf{H}_t^{-1}$
$J_{\text{MSE-s}}$	$\mathbf{H}_t = \mathbf{W}_t^{-1}$	$\longleftarrow$	$\mathbf{W}_t$

Table 2.4: Transforming an update equation for system identification into one for inverse modeling by using the matrix-inversion lemma, and vice versa.

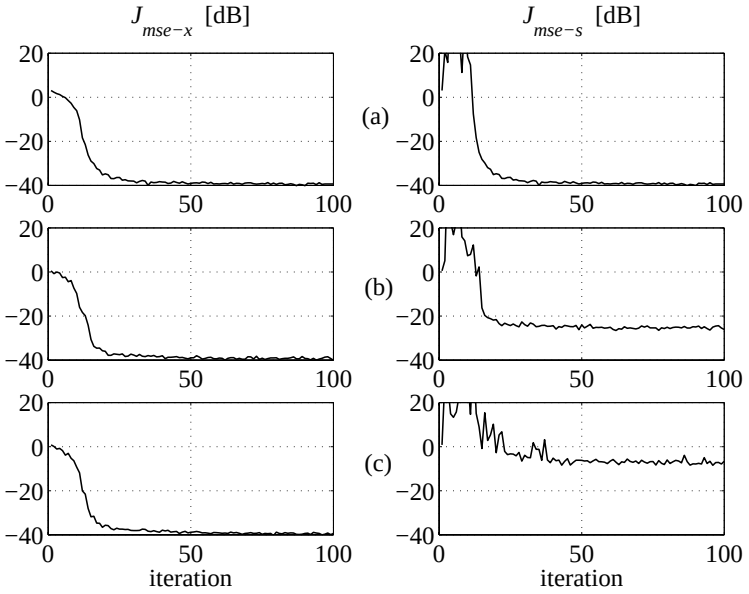

Figure 2.10: **RLS1-x**
Performance $\frac{1}{M} J_{\text{MSE-x}}$ (left column) and $\frac{1}{M} J_{\text{MSE-s}}$ (right column) of the algorithms RLS1-Hx and RLS1-Wx with forgetting factor $\lambda = 0.97$. The mixing matrix $\mathbf{A}$ has logarithmically distributed singular values and condition number:
(a) $\chi(\mathbf{A}) = 1$,
(b) $\chi(\mathbf{A}) = 10$,
(c) $\chi(\mathbf{A}) = 100$.
The curves are averages over 30 runs.

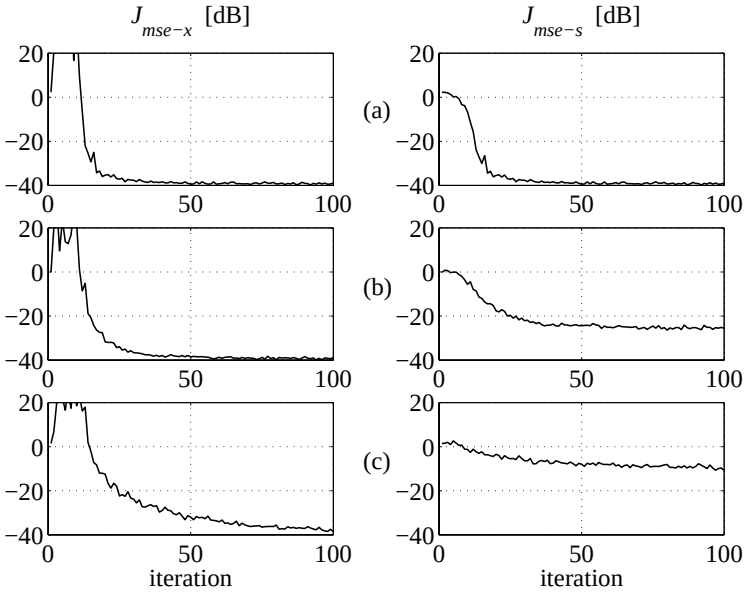

Figure 2.11: **RLS2-s**
Performance $\frac{1}{M} J_{\text{MSE-x}}$ (left column) and $\frac{1}{M} J_{\text{MSE-s}}$ (right column) of the algorithms RLS2-Hs and RLS2-Ws with forgetting factor $\lambda = 0.97$. The mixing matrix $\mathbf{A}$ has logarithmically distributed singular values and condition number:
(a) $\chi(\mathbf{A}) = 1$,
(b) $\chi(\mathbf{A}) = 10$,
(c) $\chi(\mathbf{A}) = 100$.
The curves are averages over 30 runs.

Chapter 3

Circulant matrices

In this chapter we describe the mathematical tools required to extend the adaptive algorithms for the instantaneous mixing case to also work for the convolutive mixing case.

From a Linear Algebra point of view we will focus on the *DFT matrix*, *circulant* and *block-circulant matrices*. Circulant matrices have very attractive properties, e.g., the commutative law under multiplication. Besides, the DFT matrix plays a major role in the analysis of circulant matrices.

We also define many operations which are related to signal processing and describe fast implementations thereof, e.g., convolution, correlation, time reversal of a time series, deconvolution, and others. These are described first for the single-channel case followed by a treatment of the multichannel case.

We describe these operations with polynomials, e.g. the two-sided z-transform, and also in the context of circulant matrices. In the subsequent chapters, we will use the polynomial representation of time series and FIR filters. For the implementation, however, it is more convenient to work with circulant or diagonal matrices, for reasons of computational efficiency.

For a thorough analysis of circulant matrices and their properties, the reader is referred to [25, 48, 49].

We will use the following notations:

- $\bar{\mathbf{A}}$ denotes a diagonal matrix,
- $\overline{\mathbf{A}}$ denotes a block diagonal matrix,
- $\tilde{\mathbf{A}}$ denotes a circulant matrix,
- $\widetilde{\mathbf{A}}$ denotes a block circulant matrix,
- $\bar{\mathbf{a}}$ denotes $\mathrm{diag}\left(\bar{\mathbf{A}}\right)$.
- $\tilde{\mathbf{a}}$ denotes the first column vector of $\tilde{\mathbf{A}}$.

3.1 Special matrices

3.1.1 DFT matrix

The normalized $C \times C$ *DFT* or *Fourier matrix* $\mathbf{F}$ is defined as:

$$[\mathbf{F}_C]_{mn} = \frac{1}{\sqrt{C}} e^{-j\frac{2\pi}{C}mn} \qquad (m, n = 0 \ldots C-1) \tag{3.1}$$

with C being the size of the DFT or FFT. $\mathbf{F}$ is symmetric and unitary, and has the following properties:

$$\mathbf{F}^{-1} = \mathbf{F}^{*} = \mathbf{F}^{H} \tag{3.2}$$
$$\mathbf{F}^{T} = \mathbf{F} \tag{3.3}$$
$$\mathbf{F}^{4} = \mathbf{I} \tag{3.4}$$
$$\mathbf{F}^{2} = \mathbf{F}^{-2} = \mathbf{J}_{\mathrm{c}} \tag{3.5}$$

where the *circulant-time-reversal matrix* or *circulant exchange matrix* $\mathbf{J}_{\mathrm{c}}$ is defined as

$$\mathbf{J}_{\mathrm{c}} \triangleq \begin{bmatrix} 1 & & & 0 \\ & & \iddots & 1 \\ & \iddots & \iddots & \\ 0 & 1 & & \end{bmatrix}. \tag{3.6}$$

with $\mathbf{J}_{\mathrm{c}}^2 = \mathbf{I}$ (*involutary*). The *Fast Fourier Transform* (FFT) is a fast method to compute the matrix-vector product $\bar{\mathbf{x}} = \mathbf{F}\tilde{\mathbf{x}}$ or the similarity transform $\bar{\mathbf{X}} = \mathbf{F}\tilde{\mathbf{X}}\mathbf{F}^{-1}$, i.e. $\bar{\mathbf{x}} = \mathrm{FFT}\left(\tilde{\mathbf{x}}\right)$.

Furthermore, we define the following $MC \times MC$ diagonal block matrix

$$\mathbf{T}_M \triangleq \mathbf{I}_M \otimes \mathbf{F}_C = \begin{bmatrix} \mathbf{F}_C & & \\ & \ddots & \\ & & \mathbf{F}_C \end{bmatrix} \tag{3.7}$$

$$\begin{aligned} \mathbf{T}_M^{-1} &= \left(\mathbf{I}_M \otimes \mathbf{F}_C\right)^{-1} \\ &= \mathbf{I}_M \otimes \mathbf{F}_C^{-1} \end{aligned} \tag{3.8}$$

where $\otimes$ denotes the Kronecker product. Note that $\mathbf{T}_1 = \mathbf{F}_C$. Properties of Kronecker products are described in [15, 46, 101].

3.1.2 Exchange matrix

The *exchange matrix* $\mathbf{J}$ is defined as

$$\mathbf{J} \triangleq \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix} \tag{3.9}$$

and has the property $\mathbf{J}^2 = \mathbf{I}$ (*involutary*), i.e., $\mathbf{J}$ is its own inverse $\mathbf{J}^{-1} = \mathbf{J}$. The product $\mathbf{J}\mathbf{x} = \left(x_M, \ldots, x_1\right)^T$ changes the ordering of the elements. If $\mathbf{x}$ contains the elements of a two-sided time-series, then

$$\mathbf{J}\left(x_{-T_{\mathrm{x}}}, \ldots, x_0, \ldots, x_{T_{\mathrm{x}}}\right)^T = \left(x_{T_{\mathrm{x}}}, \ldots, x_0, \ldots, x_{-T_{\mathrm{x}}}\right)^T \tag{3.10}$$

performs a linear time reversal of the sequence.

Pre- and postmultiplication of a square matrix $\mathbf{A}$ by $\mathbf{J}$ flips the row and column ordering

$$\mathbf{J}\mathbf{A}\mathbf{J} = \begin{bmatrix} a_{M,M} & \cdots & a_{M,1} \\ \vdots & & \vdots \\ a_{1,M} & \cdots & a_{1,1} \end{bmatrix}. \tag{3.11}$$

3.1.3 Projection matrix

A *projection matrix* $\mathbf{P}$ has the following properties:

Property 3.1.1 Projection matrices are idempotent: $\mathbf{P}^2 = \mathbf{P}$.

Property 3.1.2 The eigenvalues of a projection matrix are either zero or one. Consequently the rank of a projection matrix $\mathbf{P}$ is $\mathrm{tr}\{\mathbf{P}\}$.

We introduce the following projection matrix which we will use in the succeeding sections

$$\tilde{\mathbf{P}}_{-N_1,N_2} = \begin{bmatrix} \mathbf{I}_{N_2+1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{N_1} \end{bmatrix}^{C\times C} . \tag{3.12}$$

A special case is when $N = N_1 = N_2$

$$\tilde{\mathbf{P}}_{-N,N} = \begin{bmatrix} \mathbf{I}_{N+1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{N} \end{bmatrix}^{C\times C} . \tag{3.13}$$

The projection matrix $\tilde{\mathbf{P}}$ is used mainly to force certain filter coefficients to become zero, e.g.

$$\begin{aligned} &\tilde{\mathbf{P}}_{-N_\mathrm{h},N_\mathrm{h}}\,(h_0,\ldots,h_{N_\mathrm{h}},h_{N_\mathrm{h}+1},\ldots,h_{-N_\mathrm{h}-1},h_{-N_\mathrm{h}},\ldots,h_{-1})^T \\ &\quad = (h_0,\ldots,h_{N_\mathrm{h}},0,\ldots,0,h_{-N_\mathrm{h}},\ldots,h_{-1})^T \qquad (3.14) \\ &\tilde{\mathbf{P}}_{0,N_\mathrm{h}}\,(h_0,\ldots,h_{N_\mathrm{h}},h_{N_\mathrm{h}+1},\ldots,h_{-N_\mathrm{h}-1},h_{-N_\mathrm{h}},\ldots,h_{-1})^T \\ &\quad = (h_0,\ldots,h_{N_\mathrm{h}},0,\ldots,0)^T\ . \qquad (3.15) \end{aligned}$$

3.1.4 Diagonal matrix

Given a vector $\bar{\mathbf{h}} = (h_0,\ldots,h_{C-1})^T$, the corresponding diagonal matrix $\bar{\mathbf{H}}$ is defined as

$$\bar{\mathbf{H}} = \mathrm{diag}\left[\bar{\mathbf{h}}\right] \triangleq \begin{bmatrix} h_0 & & \\ & \ddots & \\ & & h_{C-1} \end{bmatrix} . \tag{3.16}$$

The inverse operation

$$\bar{\mathbf{h}} = \mathrm{diag}\left(\bar{\mathbf{H}}\right) \tag{3.17}$$

is defined as extracting the diagonal of a matrix into a vector.

3.1.5 Circulant matrix

Given a vector $\mathbf{h} = (h_0,\ldots,h_{C-1})^T$, the corresponding circulant matrix $\tilde{\mathbf{H}}$ is defined as

$$\tilde{\mathbf{H}} = \mathcal{C}(\mathbf{h}) \triangleq \begin{bmatrix} h_0 & h_{C-1} & \ldots & h_2 & h_1 \\ h_1 & h_0 & h_{C-1} & & h_2 \\ h_2 & h_1 & h_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & h_{C-1} \\ h_{C-1} & \ldots & h_2 & h_1 & h_0 \end{bmatrix} . \tag{3.18}$$

The first column of $\tilde{\mathbf{H}}$ determined by $\mathbf{h}$ and the succeeding columns are cyclicly down-shifted versions of $\mathbf{h}$. The inverse operation is defined as

$$\mathbf{h} = \mathcal{C}^{-1}(\tilde{\mathbf{H}}) \triangleq \tilde{\mathbf{H}}\,\mathbf{e}_1 \tag{3.19}$$

and returns the first column of the circulant matrix $\tilde{\mathbf{H}}$. Here $\mathbf{e}_1$ denotes the first unit vector.

Circulant matrices have the following properties:

Property 3.1.3 Circulant matrices are Toeplitz.

Property 3.1.4 If $\tilde{\mathbf{A}}$ is a circulant matrix, then $\tilde{\mathbf{A}}^*$, $\tilde{\mathbf{A}}^T$, and $\tilde{\mathbf{A}}^H$ are also circulant matrices.

Property 3.1.5 If $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are circulant matrices, then $\tilde{\mathbf{A}}+\tilde{\mathbf{B}}$ is also a circulant matrix.

Property 3.1.6 If $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are circulant matrices, then $\tilde{\mathbf{A}}\tilde{\mathbf{B}}$ is also a circulant matrix.

Property 3.1.7 If $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are circulant matrices, then $\tilde{\mathbf{A}}\tilde{\mathbf{B}} = \tilde{\mathbf{B}}\tilde{\mathbf{A}}$ (commutative law).

Property 3.1.8 If $\tilde{\mathbf{A}}$ is a circulant matrix, then $\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T = \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} = (\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^T$ is symmetric (and Toeplitz).

Property 3.1.9 If $\tilde{\mathbf{A}}$ is a circulant matrix, then $\tilde{\mathbf{A}}\tilde{\mathbf{A}}^* = \tilde{\mathbf{A}}^*\tilde{\mathbf{A}} = (\tilde{\mathbf{A}}\tilde{\mathbf{A}}^*)^*$ is real.

Property 3.1.10 If $\tilde{\mathbf{A}}$ is a circulant matrix, then $\tilde{\mathbf{A}}\tilde{\mathbf{A}}^H = \tilde{\mathbf{A}}^H\tilde{\mathbf{A}} = (\tilde{\mathbf{A}}\tilde{\mathbf{A}}^H)^H$ is *self-adjoint* or *Hermitian*: $\mathrm{re}(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^H)$ is symmetric and $\mathrm{im}(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^H)$ is skew symmetric.

Property 3.1.11 $\mathbf{J}\tilde{\mathbf{A}}\mathbf{J} = \tilde{\mathbf{A}}^T$.

Property 3.1.12 $\mathbf{J}\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T\mathbf{J} = \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} = \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T$.

Property 3.1.13 If $\tilde{\mathbf{A}}$ is a circulant matrix, then the similarity transform

$$\tilde{\mathbf{A}} = \mathbf{F}^{-1}\bar{\mathbf{A}}\mathbf{F} \tag{3.20}$$

is just the *eigenvalue decomposition* of $\tilde{\mathbf{A}}$. $\bar{\mathbf{A}}$ is a diagonal matrix which contains the *eigenvalues* of $\tilde{\mathbf{A}}$, and the columns/rows of $\mathbf{F}$ contain the corresponding *eigenvectors*. As a consequence, the similarity transform

$$\mathbf{F}\tilde{\mathbf{A}}\mathbf{F}^{-1} = \bar{\mathbf{A}} \tag{3.21}$$

diagonalizes *any* circulant matrix $\tilde{\mathbf{A}}$. This is probably the most important property of circulant matrices, and many of the other properties can easily be proven

by using (3.20) or (3.21). Some other useful relations are

$$\mathbf{F}\tilde{\mathbf{A}}\mathbf{F}^{-1} = \bar{\mathbf{A}} \tag{3.22}$$

$$\mathbf{F}\tilde{\mathbf{A}}^H\mathbf{F}^{-1} = \bar{\mathbf{A}}^* \tag{3.23}$$

$$\mathbf{F}\tilde{\mathbf{A}}^T\mathbf{F}^{-1} = \mathbf{F}^2\bar{\mathbf{A}}\,\mathbf{F}^{-2} = \mathbf{J}_\mathrm{c}\bar{\mathbf{A}}\mathbf{J}_\mathrm{c} \tag{3.24}$$

$$\mathbf{F}\tilde{\mathbf{A}}^*\mathbf{F}^{-1} = \mathbf{F}^2\bar{\mathbf{A}}^*\mathbf{F}^{-2} = \mathbf{J}_\mathrm{c}\bar{\mathbf{A}}^*\mathbf{J}_\mathrm{c}\,. \tag{3.25}$$

Property 3.1.14 The eigenvalues of a circulant matrix $\tilde{\mathbf{A}} = \mathcal{C}(\mathbf{a})$ are the DFT coefficients of $\mathbf{a}$, i.e., $\bar{\mathbf{a}} = \mathrm{diag}\left(\bar{\mathbf{A}}\right) = \mathbf{F}\,\mathbf{a}$.

Property 3.1.15 The inverse of a circulant matrix of full rank is also a circulant matrix. From (3.20) we immediately obtain $\tilde{\mathbf{A}}^{-1} = \mathbf{F}^{-1}\bar{\mathbf{A}}^{-1}\mathbf{F}$.

Property 3.1.16 The pseudoinverse of a circulant matrix is also a circulant matrix, $\tilde{\mathbf{A}}^\# = \mathbf{F}^{-1}\bar{\mathbf{A}}^\#\mathbf{F}$.

For a thorough analysis of the properties and the corresponding proofs see [25].

3.1.6 Circulant permutation matrix

A special class of circulant matrices are circulant permutation matrices which are defined as $\mathcal{C}(\mathbf{e}_i)^{C\times C}$ for $1 \le i \le C$, and $\mathbf{e}_i$ is the ith unity vector, e.g. $\mathcal{C}(\mathbf{e}_1) = \mathbf{I}$. We define the following $C \times C$ matrix $\tilde{\mathbf{J}}_C$ as

$$\tilde{\mathbf{J}}^{C\times C} \triangleq \mathcal{C}(\mathbf{e}_C) = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 1 & & & 0 \end{bmatrix}. \tag{3.26}$$

Besides the properties of circulant and permutation matrices, matrix $\tilde{\mathbf{J}}$ has the following properties:

$$\tilde{\mathbf{J}}^C = \tilde{\mathbf{J}}^{-C} = \mathbf{I} \tag{3.27}$$

$$\tilde{\mathbf{J}}^k = \tilde{\mathbf{J}}^{<k>_C} \tag{3.28}$$

$$\tilde{\mathbf{J}}^{-1} = \tilde{\mathbf{J}}^T \tag{3.29}$$

$$\tilde{\mathbf{J}}^{-k} = \tilde{\mathbf{J}}^{C-k} \tag{3.30}$$

$$\tilde{\mathbf{J}}^{-k} = \mathbf{J}\tilde{\mathbf{J}}^k\mathbf{J} \tag{3.31}$$

$$\tilde{\mathbf{J}} = \mathbf{J}\mathbf{J}_c = \mathbf{J}\mathbf{F}^2 . \tag{3.32}$$

Pre-multiplication of a vector with $\tilde{\mathbf{J}}$ shifts the vector elements in a cyclic manner. Let $\mathbf{h} = (h_0, \ldots, h_{C-1})^T$. Then we have

$$\tilde{\mathbf{J}}\,\mathbf{h} = (h_1, \ldots, h_{C-1}, h_0)^T \tag{3.33}$$

$$\tilde{\mathbf{J}}^k\,\mathbf{h} = (h_k, \ldots, h_{C-1}, h_0, \ldots, h_{k-1})^T \qquad (0 \le k \le C-1) \tag{3.34}$$

$$\tilde{\mathbf{J}}^k\,\mathbf{h} = (h_{<k>_C}, \ldots, h_{C-1}, h_0, \ldots, h_{<k-1>_C})^T . \tag{3.35}$$

With the help of $\tilde{\mathbf{J}}$, we can easily rearrange a block-partitioned matrix as shown in the following example. We may write a $C \times C$ matrix as

$$\tilde{\mathbf{J}}^m \begin{bmatrix} \mathbf{A}^{m\times n} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \tilde{\mathbf{J}}^{-n} = \begin{bmatrix} \mathbf{D} & \mathbf{C} \\ \mathbf{B} & \mathbf{A}^{m\times n} \end{bmatrix}^{C\times C} . \tag{3.36}$$

There are also some properties related to circulant matrices. Pre- and post-multiplication of a circulant matrix with $\tilde{\mathbf{J}}$ also shifts the matrix elements in a cyclic manner. Let $\tilde{\mathbf{H}} = \mathcal{C}(\mathbf{h})$. Then we have

$$\tilde{\mathbf{J}}^k\,\tilde{\mathbf{H}} = \tilde{\mathbf{H}}\,\tilde{\mathbf{J}}^k \qquad = \mathcal{C}(\tilde{\mathbf{J}}^k\,\mathbf{h}) \tag{3.37}$$

$$\tilde{\mathbf{J}}^m\,\tilde{\mathbf{H}}\,\tilde{\mathbf{J}}^n = \tilde{\mathbf{J}}^{m+n}\,\tilde{\mathbf{H}} = \tilde{\mathbf{H}}\,\tilde{\mathbf{J}}^{m+n} = \mathcal{C}(\tilde{\mathbf{J}}^{m+n}\,\mathbf{h}) \tag{3.38}$$

$$\tilde{\mathbf{J}}^m\,\tilde{\mathbf{H}}\,\tilde{\mathbf{J}}^{-n} = \tilde{\mathbf{J}}^{m-n}\,\tilde{\mathbf{H}} = \tilde{\mathbf{H}}\,\tilde{\mathbf{J}}^{m-n} = \mathcal{C}(\tilde{\mathbf{J}}^{m-n}\,\mathbf{h}) . \tag{3.39}$$

The usefulness of $\tilde{\mathbf{J}}$ lies mainly in the rearrangement of the vector or matrix elements of operations with circulant matrices, as seen in the following: Let $\tilde{\mathbf{U}} \triangleq \mathcal{C}(\mathbf{u})$, $\tilde{\mathbf{W}} \triangleq \mathcal{C}(\mathbf{w})$, $\tilde{\mathbf{X}} \triangleq \mathcal{C}(\mathbf{x})$, $\tilde{\mathbf{U}}' \triangleq \mathcal{C}(\mathbf{u}')$, $\tilde{\mathbf{W}}' \triangleq \mathcal{C}(\mathbf{w}')$, and $\tilde{\mathbf{X}}' \triangleq \mathcal{C}(\mathbf{x}')$. We have the following relation

$$\tilde{\mathbf{U}} = \tilde{\mathbf{W}}\,\tilde{\mathbf{X}} \quad \Longleftrightarrow \quad \tilde{\mathbf{J}}^m\tilde{\mathbf{U}} = \tilde{\mathbf{J}}^m\,\tilde{\mathbf{W}}\,\tilde{\mathbf{J}}^n\,\tilde{\mathbf{J}}^{-n}\,\tilde{\mathbf{X}} \quad \Longleftrightarrow \quad \tilde{\mathbf{U}}' = \tilde{\mathbf{W}}'\,\tilde{\mathbf{X}}' . \tag{3.40}$$

By choosing $\mathbf{u}' = \tilde{\mathbf{J}}^m\,\mathbf{u}$, $\mathbf{x}' = \tilde{\mathbf{J}}^{m+n}\,\mathbf{w}$, and $\mathbf{x}' = \tilde{\mathbf{J}}^{-n}\,\mathbf{x}$, we have $\tilde{\mathbf{U}}' = \tilde{\mathbf{J}}^m\,\tilde{\mathbf{U}}$, $\tilde{\mathbf{W}}' = \tilde{\mathbf{J}}^{m+n}\,\tilde{\mathbf{W}}$, and $\tilde{\mathbf{X}}' = \tilde{\mathbf{J}}^{-n}\,\tilde{\mathbf{X}}$. Note, $\mathbf{u}$ and $\mathbf{u}'$, $\mathbf{w}$ and $\mathbf{w}'$, and $\mathbf{u}$ and $\mathbf{u}'$ have exactly the same elements, respectively, but with a different ordering. We will use (3.40) later on in Section 3.2.2 when we discuss some algorithm design issues.

3.1.7 Block diagonal matrix

An $MC \times NC$ block diagonal matrix is defined as

$$\overline{\mathbf{H}} = \begin{bmatrix} \bar{\mathbf{H}}_{11} & \ldots & \bar{\mathbf{H}}_{1N} \\ \vdots & & \vdots \\ \bar{\mathbf{H}}_{M1} & \ldots & \bar{\mathbf{H}}_{MN} \end{bmatrix} \tag{3.41}$$

where each submatrix $\bar{\mathbf{H}}_{mn}$ is a $C \times C$ diagonal matrix.

For block diagonal matrices with block dimensions $C \times C$ the following properties hold:

Property 3.1.17 If $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ are two block diagonal matrices, then $\overline{\mathbf{A}} + \overline{\mathbf{B}}$ is also a block diagonal matrix.

Property 3.1.18 If $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ are two block diagonal matrices, then $\overline{\mathbf{A}}\,\overline{\mathbf{B}}$ is also a block diagonal matrix.

Property 3.1.19 The inverse of a block diagonal matrix of full rank is also a block diagonal matrix.

Property 3.1.20 The pseudoinverse of a block diagonal matrix is also a block diagonal matrix.

In contrast to diagonal matrices, the commutative law does not hold for block diagonal matrices.

3.1.8 Block circulant matrix

An $MC \times NC$ block circulant matrix is defined as

$$\widetilde{\mathbf{H}} = \begin{bmatrix} \tilde{\mathbf{H}}_{11} & \dots & \tilde{\mathbf{H}}_{1N} \\ \vdots & & \vdots \\ \tilde{\mathbf{H}}_{M1} & \dots & \tilde{\mathbf{H}}_{MN} \end{bmatrix} \tag{3.42}$$

where each submatrix $\tilde{\mathbf{H}}_{mn}$ is a $C \times C$ circulant matrix.

For block circulant matrices with block dimensions $C \times C$ the following properties hold:

Property 3.1.21 Block circulant matrices are block Toeplitz, i.e., $\tilde{\mathbf{H}}_{mn}$ is Toeplitz.

Property 3.1.22 If $\widetilde{\mathbf{A}}$ is a block circulant matrix, then $\widetilde{\mathbf{A}}^*$, $\widetilde{\mathbf{A}}^T$, and $\widetilde{\mathbf{A}}^H$ are also block circulant matrices.

Property 3.1.23 If $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ are two block circulant matrices, then $\widetilde{\mathbf{A}} + \widetilde{\mathbf{B}}$ is also a block circulant matrix.

Property 3.1.24 If $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ are two block circulant matrices, then $\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}$ is also a block circulant matrix.

Property 3.1.25 The similarity transform $\tilde{\mathbf{H}}_{mn} = \mathbf{F}^{-1}\bar{\mathbf{H}}_{mn}\mathbf{F}$ is the eigenvalue decomposition of the circulant submatrix $\tilde{\mathbf{H}}_{mn}$.

Property 3.1.26 The transform $\overline{\mathbf{H}} = \mathbf{T}_M \widetilde{\mathbf{H}} \mathbf{T}_N^{-1}$ transforms a block circulant matrix $\widetilde{\mathbf{H}}$ into a block diagonal matrix $\overline{\mathbf{H}}$.

Property 3.1.27 The transform $\widetilde{\mathbf{H}} = \mathbf{T}_M^{-1} \overline{\mathbf{H}} \mathbf{T}_N$ transforms a block diagonal matrix $\overline{\mathbf{H}}$ into a block circulant matrix $\widetilde{\mathbf{H}}$.

Property 3.1.28 The eigenvalues of the circulant submatrix $\tilde{\mathbf{H}}_{mn}$ are the DFT coefficients of the first column of $\tilde{\mathbf{H}}_{mn}$, i.e., $\tilde{\mathbf{H}}_{mn} = \mathcal{C}(\mathbf{h}_{mn})$, $\bar{\mathbf{h}}_{mn} = \mathbf{F}\mathbf{h}_{mn}$.

Property 3.1.29 The inverse of a square block circulant matrix of full rank is also a block circulant matrix.

Property 3.1.30 The pseudoinverse of a block circulant matrix is also a block circulant matrix.

In contrast to circulant matrices, the commutative law does not hold for block circulant matrices.

3.2 Convolution

3.2.1 Linear convolution

Linear convolution of two sequences Let x and w be two sequences

$$x \triangleq \{x_{-T_\mathrm{x}}, \dots, x_0, \dots, x_{T_\mathrm{x}}\} \tag{3.43}$$

$$w \triangleq \{w_{-N_\mathrm{w}}, \dots, w_0, \dots, w_{N_\mathrm{w}}\} \,. \tag{3.44}$$

The linear convolution of w and x is defined as

$$w * x = x * w = u \triangleq \{u_{-T_\mathrm{x}-N_\mathrm{w}}, \dots, u_0, \dots, u_{T_\mathrm{x}+N_\mathrm{w}}\} \tag{3.45}$$

where

$$u_t = (w * x)_t \triangleq \sum_{n=-N_\mathrm{w}}^{N_\mathrm{w}} w_n x_{t-n} = \sum_{n=-T_\mathrm{x}}^{T_\mathrm{x}} w_{t-n} x_n \,. \tag{3.46}$$

Fig. 3.1 illustrates the linear convolution of (3.46) and also reveals the boundary effects, caused by the finite length of the two sequences x and w.

Linear convolution as a product of two polynomials Alternatively, we can describe the convolution $u = w * x$ by a multiplication of two polynomials

$$u(z) = w(z)x(z) \tag{3.47}$$

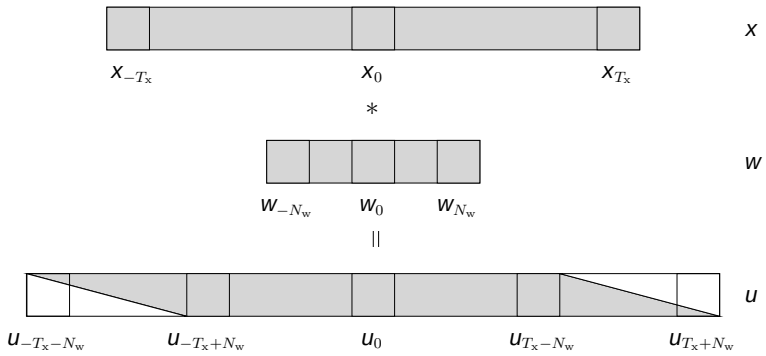

Figure 3.1: Linear convolution of two sequences of finite length ($u = w * x$). The boundary effects of u_t, where u_t is built from a sum of fewer than $2N_w + 1$ terms, are shown graphically: Fading in on the left for $t < -T_x + N_w$ and fading out on the right for $t > T_x - N_w$.

where $x(z)$, $w(z)$ and $u(z)$ are double-sided z-transforms (Laurent series) of x, w and u, i.e.,

$$x(z) \triangleq \sum_{t=-T_x}^{T_x} x_t z^{-t} \tag{3.48}$$

$$w(z) \triangleq \sum_{n=-N_w}^{N_w} w_n z^{-n} \tag{3.49}$$

$$u(z) \triangleq \sum_{t=-T_x-N_w}^{T_x+N_w} u_t z^{-t} \tag{3.50}$$

By setting $z = e^{j\omega}$ we obtain the frequency response, and sampling z on the unit circle, i.e. $z = e^{jk\frac{2\pi}{C}}$, gives the discrete Fourier domain.

Linear convolution of two vectors The convolution of two sequences can also be described as the convolution of two vectors, i.e., $\mathbf{u} = \mathbf{w} * \mathbf{x}$ with

$$\mathbf{x} \triangleq (x_{-T_x}, \ldots, x_0, \ldots, x_{T_x})^T \tag{3.51}$$

$$\mathbf{w} \triangleq (w_{-N_w}, \ldots, w_0, \ldots, w_{N_w})^T \tag{3.52}$$

$$\mathbf{u} \triangleq (u_{-T_x-N_w}, \ldots, u_0, \ldots, u_{T_x+N_w})^T \ . \tag{3.53}$$

We define the convolution operator $*$ as

$$\mathbf{u} = \mathbf{w} * \mathbf{x} \triangleq \mathcal{T}(\mathbf{w})\mathbf{x} \tag{3.54}$$

$$= \mathbf{x} * \mathbf{w} \triangleq \mathcal{T}(\mathbf{x})\mathbf{w} \tag{3.55}$$

$$\begin{pmatrix} u_{-T_x-N_w} \\ \vdots \\ u_{-T_x} \\ \vdots \\ u_0 \\ \vdots \\ u_{T_x} \\ \vdots \\ u_{T_x+N_w} \end{pmatrix} = \begin{bmatrix} w_{-N_w} & & \\ \vdots & \ddots & \\ w_0 & & w_{-N_w} \\ \vdots & \ddots & \vdots \\ w_{N_w} & & w_0 \\ & \ddots & \vdots \\ & & w_{N_w} \end{bmatrix} \begin{pmatrix} x_{-T_x} \\ \vdots \\ x_0 \\ \vdots \\ x_{T_x} \end{pmatrix} \tag{3.56}$$

$\mathcal{T}(\mathbf{w})$ is a lower triangular Toeplitz matrix having $\mathbf{w}$ padded with zeros in its first column [65]. Note that $\mathcal{T}(\mathbf{w})$ has dimension $(2(T_x+N_w)+1) \times (2T_x+1)$ whereas $\mathcal{T}(\mathbf{x})$ is a $(2(T_x + N_w) + 1) \times (2N_w + 1)$ lower triangular Toeplitz matrix. In fact, the dimensions of the matrix $\mathcal{T}(.)$ are defined by the argument and the subsequent vector. The extension to the convolution of three sequences $u = w * a * s$ is straightforward and is defined as

$$\mathbf{u} = \mathbf{w} * \mathbf{a} * \mathbf{s} = \mathcal{T}(\mathbf{w})\left(\mathcal{T}(\mathbf{a})\mathbf{s}\right) \tag{3.57}$$

$$= \mathcal{T}(\mathcal{T}(\mathbf{w})\mathbf{a})\mathbf{s} \tag{3.58}$$

We say that the linear convolution of two time series $u = w * x$, the multiplication of their corresponding polynomials of the double-sided z-transform $u(z) = w(z)x(z)$, and the convolution of two vectors which contain the coefficients of the time series $\mathbf{u} = \mathbf{w} * \mathbf{x}$ are *isomorphic*, i.e.,

$$u = w * x \cong u(z) = w(z)x(z) \cong \mathbf{u} = \mathbf{w} * \mathbf{x} \tag{3.59}$$

as all representations yield the same result, i.e., the elements of u, $u(z)$ and $\mathbf{u}$ are identical. In Section 3.2.3 we will describe an efficient method for computing the linear convolution of two time series, by exploiting a fast algorithm to compute the circular convolution.

3.2.2 Circular convolution

Circular convolution of two sequences Let $\tilde{x}$ and $\tilde{w}$ be two sequences of finite length $C \geq 2\max(T_\mathrm{x}, N_\mathrm{w}) + 1$, which are zero-centered (C odd)

$$\tilde{x} \triangleq \{0, \ldots, 0, x_{-T_\mathrm{x}}, \ldots, x_0, \ldots, x_{T_\mathrm{x}}, 0, \ldots, 0\}_C \tag{3.60}$$

$$\tilde{w} \triangleq \{0, \ldots, 0, w_{-N_\mathrm{w}}, \ldots, w_0, \ldots, w_{N_\mathrm{w}}, 0, \ldots, 0\}_C \,. \tag{3.61}$$

We define the *circular convolution* as

$$\tilde{u} = \tilde{w} \circledast \tilde{x} \tag{3.62}$$

with

$$\tilde{u}_t = (\tilde{w} \circledast \tilde{x})_t \triangleq \sum_{n=-N_\mathrm{w}}^{N_\mathrm{w}} w_n x_{<t-n>_C} \quad t \in \{-\lfloor C/2 \rfloor, \cdots, \lfloor C/2 \rfloor\} \tag{3.63}$$

$$\triangleq \sum_{n=-T_\mathrm{x}}^{T_\mathrm{x}} w_{<t-n>_C} x_n \quad t \in \{-\lfloor C/2 \rfloor, \cdots, \lfloor C/2 \rfloor\} \,. \tag{3.64}$$

The sequence $\tilde{u}$ also has length C. The *generalized remainder* $< . >_C$ is defined in Appendix D.1 and returns values from $\{-\lfloor C/2 \rfloor, \cdots, \lfloor C/2 \rfloor\}$. Depending on C, T_x, and N_w, we distinguish between three different cases, see also Fig. 3.2:

1. $\boxed{C > 2\,(T_\mathrm{x} + N_\mathrm{w}) + 1}$

   $$\tilde{u} = \{0, \ldots, 0, u_{-T_\mathrm{x}-N_\mathrm{w}}, \ldots, u_0, \ldots, u_{T_\mathrm{x}+N_\mathrm{w}}, 0, \ldots, 0\}_C \tag{3.65}$$

   The non-zero elements of the circulant convolution $\tilde{u} = \tilde{w} \circledast \tilde{x}$ coincide with the elements of the linear convolution $u = w * x$.

2. $\boxed{C = 2\,(T_\mathrm{x} + N_\mathrm{w}) + 1}$

   $$\tilde{u} = \{u_{-T_\mathrm{x}-N_\mathrm{w}}, \ldots, u_0, \ldots, u_{T_\mathrm{x}+N_\mathrm{w}}\}_C \tag{3.66}$$

   The linear and the circular convolution are equal, i.e. $\tilde{u} = \tilde{w} \circledast \tilde{x} = w * x = u$.

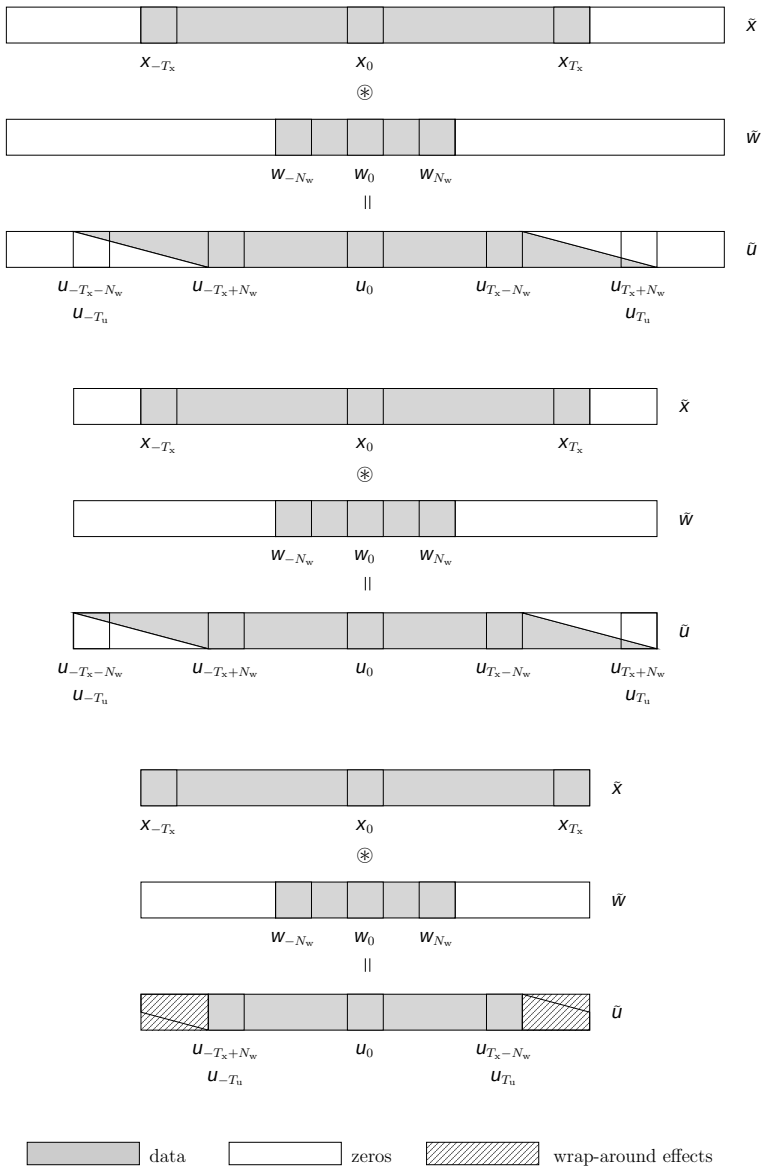

Figure 3.2: Circular convolution of two sequences $\tilde{u} = \tilde{w} \circledast \tilde{x}$ with
$C > 2\,(T_\mathrm{x} + N_\mathrm{w}) + 1$, $T_\mathrm{u} = T_\mathrm{x} + N_\mathrm{w}$ (top),
$C = 2\,(T_\mathrm{x} + N_\mathrm{w}) + 1$, $T_\mathrm{u} = T_\mathrm{x} + N_\mathrm{w} = C$ (middle),
$C < 2\,(T_\mathrm{x} + N_\mathrm{w}) + 1$, $T_\mathrm{u} = C - T_\mathrm{x} - N_\mathrm{w} - 1$ (bottom).

3. $\boxed{2\left(T_{\mathrm{x}}+N_{\mathrm{w}}\right)+1>C\geq 2\max\left(T_{\mathrm{x}},N_{\mathrm{w}}\right)+1}$

$$\tilde{u}=\left\{u_{-\lfloor C/2\rfloor},\ldots,u_{-T_{\mathrm{u}}},\ldots,u_0,\ldots,u_{T_{\mathrm{u}}},\ldots,u_{\lfloor C/2\rfloor}\right\}_C \tag{3.67}$$

where the subsequence $\mathcal{P}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\left(\tilde{u}\right)\triangleq\left\{u_{-T_{\mathrm{u}}},\ldots,u_0,\ldots,u_{T_{\mathrm{u}}}\right\}$ coincides with the $2T_{\mathrm{u}}+1$ center elements of the linear convolution $u=w*x$, i.e.,

$$\mathcal{P}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\left(\tilde{u}\right)=\mathcal{P}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\left(u\right)\triangleq\left\{u_{-T_{\mathrm{u}}},\ldots,u_0,\ldots,u_{T_{\mathrm{u}}}\right\} \tag{3.68}$$

which holds for

$$T_{\mathrm{u}}\leq C-T_{\mathrm{x}}-N_{\mathrm{w}}-1\,. \tag{3.69}$$

The upper bound in (3.69) was obtained from $2T_{\mathrm{u}}+1=2C-2\left(T_{\mathrm{x}}+N_{\mathrm{w}}\right)-1$, see Fig. 3.2.

Circular convolution as a multiplication of two polynomials Alternatively, we can describe the circular convolution $\tilde{u}=\tilde{w}\circledast\tilde{x}$ for a given length C by a multiplication of two polynomials followed by a circular polynomial projection

$$\begin{aligned}\tilde{u}(z)&\triangleq\tilde{\mathcal{P}}_C\left(u(z)\right) &(3.70)\\ &=\tilde{\mathcal{P}}_C\left(w(z)\,x(z)\right) &(3.71)\end{aligned}$$

where $\tilde{\mathcal{P}}_C\left(.\right)$ is the *circular polynomial projection operator* defined in Appendix D.2. Similar to (3.68) we have

$$\mathcal{P}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\left(\tilde{u}(z)\right)=\mathcal{P}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\left(u(z)\right)\triangleq\sum_{t=-T_{\mathrm{u}}}^{T_{\mathrm{u}}}u_t z^{-t} \tag{3.72}$$

which holds for

$$T_{\mathrm{u}}\leq\begin{cases}T_{\mathrm{x}}+N_{\mathrm{w}} & \text{for } 2\left(T_{\mathrm{x}}+N_{\mathrm{w}}\right)+1\leq C\\ C-T_{\mathrm{x}}-N_{\mathrm{w}}-1 & \text{for } 2\left(T_{\mathrm{x}}+N_{\mathrm{w}}\right)+1>C\geq 2\max\left(T_{\mathrm{x}},N_{\mathrm{w}}\right)+1\,.\end{cases} \tag{3.73}$$

just as for $\tilde{u}=\tilde{w}\circledast\tilde{x}$. This means that the center $2T_{\mathrm{u}}+1$ elements of the linear and circular convolution are equal.

In the case where $C<2T_{\mathrm{x}}+1$ or $C<2N_{\mathrm{w}}+1$, we can write with (D.111)

$$\begin{aligned}\tilde{u}(z)&=\tilde{\mathcal{P}}_C\left(\tilde{\mathcal{P}}_C\left(w(z)\right)\tilde{\mathcal{P}}_C\left(x(z)\right)\right) &(3.74)\\ &=\tilde{\mathcal{P}}_C\left(\tilde{w}(z)\,\tilde{x}(z)\right) &(3.75)\end{aligned}$$

where $\tilde{x}(z)\triangleq\tilde{\mathcal{P}}_C\left(x(z)\right)$ and $\tilde{w}(z)\triangleq\tilde{\mathcal{P}}_C\left(w(z)\right)$.

Note, that if we find a fast algorithm for the computation of a circular convolution, we immediately also have a fast algorithm for a linear convolution, because due to (3.72) there are always $2T_{\mathrm{u}}+1$ elements of the circular convolution which coincide with $2T_{\mathrm{u}}+1$ elements of the linear convolution. From (3.73) we see that increasing C also increases T_{u} for small values of C.

Circular convolution of two vectors We define the following vectors

$$\begin{aligned}\tilde{\mathbf{x}}&\triangleq\left(x_0,\ldots,x_{T_{\mathrm{x}}},0,\ldots,0,x_{-T_{\mathrm{x}}},\ldots,x_{-1}\right)^T &(3.76)\\ \tilde{\mathbf{w}}&\triangleq\left(w_0,\ldots,w_{N_{\mathrm{w}}},0,\ldots,0,w_{-N_{\mathrm{w}}},\ldots,w_{-1}\right)^T &(3.77)\\ \tilde{\mathbf{u}}&\triangleq\left(u_0,\ldots,u_{T_{\mathrm{u}}},0,\ldots,0,u_{-T_{\mathrm{u}}},\ldots,u_{-1}\right)^T\,. &(3.78)\end{aligned}$$

The vectors are zero-padded in the center such that they all have length C. Here we remove the constraint that C must be odd. Note, that the elements of the sequence x and those of the vector $\mathbf{x}$, defined in (3.51), are arranged similarly in ascending order. However, the elements of $\tilde{x}$ and $\tilde{\mathbf{x}}$ are arranged in a different manner. We use the following definition for the circular convolution of two vectors

$$\begin{aligned}\tilde{\mathbf{u}}=\tilde{\mathbf{x}}\circledast\tilde{\mathbf{w}}=\tilde{\mathbf{w}}\circledast\tilde{\mathbf{x}}&\triangleq\mathcal{C}^{-1}(\mathcal{C}(\tilde{\mathbf{w}})\,\mathcal{C}(\tilde{\mathbf{x}})) &(3.79)\\ &=\mathcal{C}\left(\tilde{\mathbf{w}}\right)\tilde{\mathbf{x}}=\mathcal{C}\left(\tilde{\mathbf{x}}\right)\tilde{\mathbf{w}} &(3.80)\end{aligned}$$

where $\mathcal{C}\left(.\right)$ and $\mathcal{C}^{-1}(.)$ are defined in (3.18) and (3.19), respectively. The definition (3.79) requires a product of two matrices, whereas (3.80) uses only a matrix-vector product. Extending both sides of (3.79) to a circulant matrix, and using the definitions $\tilde{\mathbf{U}}\triangleq\mathcal{C}(\tilde{\mathbf{u}})$, $\tilde{\mathbf{W}}\triangleq\mathcal{C}(\tilde{\mathbf{w}})$, and $\tilde{\mathbf{X}}\triangleq\mathcal{C}(\tilde{\mathbf{x}})$, gives

$$\begin{aligned}\mathcal{C}(\tilde{\mathbf{u}})=\mathcal{C}(\tilde{\mathbf{x}}\circledast\tilde{\mathbf{w}})&=\mathcal{C}\left(\tilde{\mathbf{w}}\right)\mathcal{C}(\tilde{\mathbf{x}}) &(3.81)\\ \tilde{\mathbf{U}}&=\tilde{\mathbf{W}}\tilde{\mathbf{X}}\,. &(3.82)\end{aligned}$$

We see that the circular convolution of two vectors is isomorphic to the product of two circulant matrices.

As an example, constructing the circulant matrix $\tilde{\mathbf{W}} = \mathcal{C}(\tilde{\mathbf{w}})$ with $\tilde{\mathbf{w}}$ defined in (3.77), yields

$$\tilde{\mathbf{W}} = \begin{bmatrix} w_0 & w_{-1} & \cdots & w_{-N_\mathrm{w}} & 0 & \cdots & 0 & w_{N_\mathrm{w}} & \cdots & w_1 \\ w_1 & w_0 & & & \ddots & & & & \ddots & \vdots \\ \vdots & & \ddots & & & \ddots & & & & w_{N_\mathrm{w}} \\ w_{N_\mathrm{w}} & & & \ddots & & & \ddots & & & 0 \\ 0 & \ddots & & & \ddots & & & \ddots & & \vdots \\ \vdots & & \ddots & & & \ddots & & & \ddots & 0 \\ 0 & & & \ddots & & & \ddots & & & w_{-N_\mathrm{w}} \\ w_{-N_\mathrm{w}} & & & & \ddots & & & \ddots & & \vdots \\ \vdots & \ddots & & & & \ddots & & & w_0 & w_{-1} \\ w_{-1} & \cdots & w_{-N_\mathrm{w}} & 0 & \cdots & 0 & w_{N_\mathrm{w}} & \cdots & w_1 & w_0 \end{bmatrix} \tag{3.83}$$

Note, arranging the elements w_n in $\tilde{\mathbf{w}}$ as given in (3.77), causes the center element w_0 to lie on the diagonal of $\tilde{\mathbf{W}}$. This representation has the useful property that the elements of $\mathcal{C}^{-1}(\tilde{\mathbf{W}}^T)$ and $\mathcal{C}^{-1}(\tilde{\mathbf{W}})$ are arranged in circular time-reversed order, $\mathcal{C}^{-1}(\tilde{\mathbf{W}}^T) = \mathbf{J}_\mathrm{c}\mathcal{C}^{-1}(\tilde{\mathbf{W}})$ where $\mathbf{J}_\mathrm{c}$ is defined in (3.6). This means that transposition of the filter matrix $\tilde{\mathbf{W}}$ or the input-signal matrix $\tilde{\mathbf{X}}$ causes a circular time reversal of the underlying filter or signal sequence, respectively. This is equal to changing $w(z)$ to $w(z^{-1})$. Since $\tilde{\mathbf{X}} = \mathcal{C}(\tilde{\mathbf{x}})$ is built similarly to $\tilde{\mathbf{W}}$, x_0 lies on the diagonal of $\tilde{\mathbf{X}}$. As a consequence, $\tilde{\mathbf{U}} = \tilde{\mathbf{W}}\tilde{\mathbf{X}}$ will have u_0 on its main diagonal.

Circular convolution of three vectors Of course the circular convolution operation can be extended straightforwardly to the circular convolution of three or more vectors

$$\begin{aligned} \tilde{\mathbf{u}} = \tilde{\mathbf{w}} \circledast \tilde{\mathbf{a}} \circledast \tilde{\mathbf{s}} &= \mathcal{C}^{-1}(\mathcal{C}(\tilde{\mathbf{w}})\,\mathcal{C}(\tilde{\mathbf{a}})\,\mathcal{C}(\tilde{\mathbf{s}})) && (3.84)\\ &= \mathcal{C}(\tilde{\mathbf{w}})\,\mathcal{C}(\tilde{\mathbf{a}})\tilde{\mathbf{s}} && (3.85)\\ &= \mathcal{C}(\mathcal{C}(\tilde{\mathbf{w}})\tilde{\mathbf{a}})\tilde{\mathbf{s}} && (3.86)\\ &= \mathcal{C}^{-1}(\mathcal{C}(\mathcal{C}(\tilde{\mathbf{w}})\tilde{\mathbf{a}})\,\mathcal{C}(\tilde{\mathbf{s}}))\,. && (3.87) \end{aligned}$$

The circular convolution can be described in matrix form with the help of circulant matrices as

$$\tilde{\mathbf{U}} = \tilde{\mathbf{W}}\tilde{\mathbf{A}}\tilde{\mathbf{S}}\,. \tag{3.88}$$

To summarize, we have the following isomorphism

$$\begin{aligned} \tilde{u} = \tilde{w} \circledast \tilde{x} &\cong \tilde{u}(z) = \tilde{\mathcal{P}}_C\,(\tilde{w}(z)\,\tilde{x}(z)) \\ &\cong \tilde{\mathbf{u}} = \tilde{\mathbf{x}} \circledast \tilde{\mathbf{w}} \cong \tilde{\mathbf{U}} = \tilde{\mathbf{W}}\tilde{\mathbf{X}} \cong \bar{\mathbf{U}} = \bar{\mathbf{W}}\bar{\mathbf{X}}\,. \end{aligned} \tag{3.89}$$

Rearrangement of vector elements For implementation reasons, one is sometimes interested in rearranging the vector elements for a circular convolution, e.g. $\tilde{\mathbf{w}} \circledast \tilde{\mathbf{x}}$. From (3.40) with $m = 0$ we have

$$\tilde{\mathbf{u}} = \tilde{\mathbf{w}} \circledast \tilde{\mathbf{x}} = \tilde{\mathbf{J}}^n\,\tilde{\mathbf{w}} \circledast \tilde{\mathbf{J}}^{-n}\,\tilde{\mathbf{x}} = \tilde{\mathbf{w}}' \circledast \tilde{\mathbf{x}}' \tag{3.90}$$

which reveals that the output vector $\tilde{\mathbf{u}}$ remains unaffected if we rotate the elements of $\tilde{\mathbf{w}}$ anti-clockwise and rotate the elements of $\tilde{\mathbf{x}}$ clockwise by the same number, and vice versa. More generally, if we also want to rotate the elements of the output vector $\tilde{\mathbf{u}}$, again from using (3.40) we have

$$\tilde{\mathbf{u}} = \tilde{\mathbf{w}} \circledast \tilde{\mathbf{x}} \quad\Longleftrightarrow\quad \tilde{\mathbf{J}}^m\,\tilde{\mathbf{u}} = \tilde{\mathbf{J}}^{m+n}\,\tilde{\mathbf{w}} \circledast \tilde{\mathbf{J}}^{-n}\,\tilde{\mathbf{x}} \quad\Longleftrightarrow\quad \tilde{\mathbf{u}}' = \tilde{\mathbf{w}}' \circledast \tilde{\mathbf{x}}'\,. \tag{3.91}$$

with $\tilde{\mathbf{u}}' = \tilde{\mathbf{J}}^m\,\tilde{\mathbf{u}}$, $\tilde{\mathbf{w}}' = \tilde{\mathbf{J}}^{m+n}\,\tilde{\mathbf{w}}$, and $\tilde{\mathbf{x}}' = \tilde{\mathbf{J}}^{-n}\,\tilde{\mathbf{x}}$. Note that $\tilde{\mathbf{J}}^{-n} = (\tilde{\mathbf{J}}^n)^T$. Eq. (3.91) is a very powerful tool for algorithm design purposes.

3.2.3 Fast computation of the convolution

Fast computation of a circular convolution We now derive an efficient method for computing a circular convolution via the product of two circulant

matrices. Starting with (3.82), we apply the similarity transform given in (3.21) and obtain

$$\mathbf{F}\tilde{\mathbf{U}}\mathbf{F}^{-1} = \mathbf{F}\tilde{\mathbf{W}}\mathbf{F}^{-1}\mathbf{F}\tilde{\mathbf{X}}\mathbf{F}^{-1} \tag{3.92}$$

$$\bar{\mathbf{U}} = \bar{\mathbf{W}}\bar{\mathbf{X}}\,. \tag{3.93}$$

Eq. (3.93) is a product of two diagonal matrices which requires with

$$\bar{\mathbf{u}} = \bar{\mathbf{w}} \odot \bar{\mathbf{x}} \tag{3.94}$$

only C multiplications, where $\odot$ denotes the element-wise multiplication of two vectors, $\bar{\mathbf{u}} \triangleq \operatorname{diag}\left(\bar{\mathbf{U}}\right)$, $\bar{\mathbf{w}} \triangleq \operatorname{diag}\left(\bar{\mathbf{W}}\right)$, and $\bar{\mathbf{x}} \triangleq \operatorname{diag}\left(\bar{\mathbf{X}}\right)$. By further exploiting the computational efficiency of the FFT and IFFT, i.e., $\bar{\mathbf{x}} = \mathbf{F}\tilde{\mathbf{x}} = \text{FFT}\left(\tilde{\mathbf{x}}\right)$, $\bar{\mathbf{w}} = \mathbf{F}\tilde{\mathbf{w}} = \text{FFT}\left(\tilde{\mathbf{w}}\right)$, and $\tilde{\mathbf{u}} = \mathbf{F}^{-1}\bar{\mathbf{u}} = \text{IFFT}\left(\bar{\mathbf{u}}\right)$, we finally obtain

$$\tilde{\mathbf{u}} = \tilde{\mathbf{w}} \circledast \tilde{\mathbf{x}} = \text{IFFT}\left(\text{FFT}\left(\tilde{\mathbf{w}}\right) \odot \text{FFT}\left(\tilde{\mathbf{x}}\right)\right) \tag{3.95}$$

as a fast implementation of the circular convolution. The extension of (3.95) to more than two vectors is straightforward, e.g., (3.84) is computed as

$$\tilde{\mathbf{u}} = \text{IFFT}\left(\text{FFT}\left(\tilde{\mathbf{w}}\right) \odot \text{FFT}\left(\tilde{\mathbf{a}}\right) \odot \text{FFT}\left(\tilde{\mathbf{s}}\right)\right)\,. \tag{3.96}$$

Fast computation of a linear convolution From Section 3.2.2 we know that the linear convolution of two sequences can be computed by the circular convolution with the appropriate length C, e.g., $w(z)x(z) = \tilde{\mathcal{P}}_C\left(w(z)\,x(z)\right)$ for $C \geq 2\left(T_\text{x} + N_\text{w}\right) + 1$. Since with (3.95) there exists a fast algorithm for the computation of the circular convolution, we automatically also have a fast algorithm for the linear convolution. If $C < 2\left(T_\text{x} + N_\text{w}\right) + 1$, from (3.69) and (3.73) we know that there are $T_\text{u} = C - T_\text{x} - N_\text{w} - 1$ elements of $\tilde{\mathbf{u}} = \tilde{\mathbf{w}} \circledast \tilde{\mathbf{x}}$ which coincide with those of $\mathbf{u} = \mathbf{w} * \mathbf{x}$, see also Fig. 3.2.

3.3 Complex conjugation, time reversal, and correlation

With linear time we mean that $-\infty \geq t_\text{lin} \geq \infty$ and with circular time we mean $t_\text{circ} = <t_\text{lin}>_C$. With *linear time reversal* we understand the mapping $t_\text{lin} \rightarrow -t_\text{lin}$ and with *circular time reversal* the mapping $t_\text{circ} \rightarrow -t_\text{circ} = <-t_\text{lin}>_C$. The relation between linear and circular time reversal is illustrated in Fig. 3.3.

3.3.1 Linear time reversal

Time reversal of a sequence Let $x = \{x_{-T_\text{x}}, \ldots, x_{T_\text{x}}\}$ be a sequence of length $2T_\text{x}+1$. With $\bar{x} = \{x_{T_\text{x}}, \ldots, x_{-T_\text{x}}\}$ we denote the time-reversed ordered sequence. Furthermore, we define $x^* = \left\{x^*_{-T_\text{x}}, \ldots, x^*_{T_\text{x}}\right\}$ and consequently $\bar{x}^* = \left\{x^*_{T_\text{x}}, \ldots, x^*_{-T_\text{x}}\right\}$.

Time reversal of a polynomial The z-transform of a sequence x can be written as

$$x(z) = \mathbf{x}^T\mathbf{z} = \mathbf{z}^T\mathbf{x} \tag{3.97}$$

$$\mathbf{x} = \left(x_{-T_\text{x}}, \ldots, x_0, \ldots, x_{T_\text{x}}\right)^T \tag{3.98}$$

$$\mathbf{z} = \left(z^{T_\text{x}}, \ldots, z, 1, z^{-1} \ldots, z^{-T_\text{x}}\right)^T\,. \tag{3.99}$$

Complex conjugating both sides of (3.97) yields

$$x^*(z) \triangleq \left(x(z)\right)^* = \left(\mathbf{x}^T\right)^*\mathbf{z}^* = \mathbf{x}^H\mathbf{J}\,\mathbf{z} = x_*(z^{-1}) \tag{3.100}$$

where we used $\mathbf{z}^* = \mathbf{J}\mathbf{z}$ with $\mathbf{J}$ from (3.9) and

$$\mathbf{x}^* = \left(x^*_{-T_\text{x}}, \ldots, x^*_0, \ldots, x^*_{T_\text{x}}\right)^T \tag{3.101}$$

$$\mathbf{z}^* = \left(z^{-T_\text{x}}, \ldots, z^{-1}, 1, z \ldots, z^{T_\text{x}}\right)^T\,. \tag{3.102}$$

We see that the filter coefficients are complex conjugate, and the filter is time reversed. Furthermore, with $x(z^{-1}) = \mathbf{x}^T\mathbf{z}^*$ and $x_*(z) \triangleq \left(\mathbf{x}^T\right)^*\mathbf{z}$ we have the following relations

$$x(z) = \sum_{n=N_1}^{N_2} x_n z^{-n} \tag{3.103}$$

$$x(z^{-1}) = \sum_{n=N_1}^{N_2} x_n z^{+n} = \sum_{n=-N_2}^{-N_1} x_{-n} z^{-n} \tag{3.104}$$

$$x_*(z) = \sum_{n=N_1}^{N_2} x^*_n z^{-n} \tag{3.105}$$

$$x^*(z) = \sum_{n=N_1}^{N_2} x^*_n z^{+n} = \sum_{n=-N_2}^{-N_1} x^*_{-n} z^{-n} = x_*(z^{-1})\,. \tag{3.106}$$

$x(z^{-1})$ corresponds to a linear time reversal of $x(z)$, in $x_*(z)$ the coefficients of $x(z)$ are complex conjugate, and $x^*(z)$ is the combination of both.

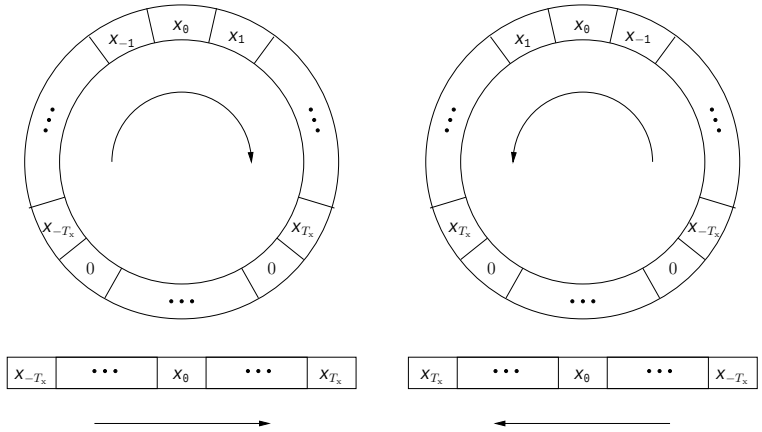

Figure 3.3: Linear and circular time-reversal: (left) $x(z)$: $\mathbf{x}$ and $\tilde{\mathbf{x}}$, (right) $x(z^{-1})$: $\mathbf{J}\,\mathbf{x}$ and $\mathbf{J}_{\mathrm{c}}\,\tilde{\mathbf{x}}$.

Time reversal of vector The elements of a vector $\mathbf{x}$ are time reversed by premultiplication with the exchange matrix, i.e. $\mathbf{J}\mathbf{x}$.

Linear correlation Complex conjugation and time reversal play an important role for the linear convolution operation. Let x and w denote two sequences of finite length. Since $w * x$ is a linear convolution, time reversing and complex conjugation of one sequence yields a linear correlation, e.g., $\bar{w}^* * x$ and $w * \bar{x}^*$. Likewise, $w^*(z)x(z)$ and $w(z)x^*(z)$ are linear correlations described by polynomials, and $\mathbf{J}\mathbf{w}^* * \mathbf{x}$ and $\mathbf{w} * \mathbf{J}\mathbf{x}^*$ are linear correlations described by vectors.

3.3.2 Circular time reversal

Circular time reversal of a sequence Let $\tilde{x}$ be a zero-centered sequence of length C. The circular time reversal is equal to the linear time reversal.

Circular time reversal of a polynomial Let

$$\tilde{x}(z) \triangleq \tilde{\mathcal{P}}_C\left(x(z)\right) = \sum_{n=-\lfloor C/2 \rfloor}^{\lfloor C/2 \rfloor} \tilde{x}_n z^{-n} \tag{3.107}$$

then $\tilde{x}(z^{-1})$, $\tilde{x}_*(z)$, and $\tilde{x}^*(z)$ are defined similar to (3.104), (3.105), and (3.106), respectively. Furthermore we have

$$\tilde{x}(z^{-1}) = \tilde{\mathcal{P}}_C\left(x(z^{-1})\right) \tag{3.108}$$

which means, if $\tilde{x}(z)$ is built from a polynomial $x(z) = \sum_{n=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} x_n z^{-n}$ with $2T_{\mathrm{x}} + 1 > C$, then it does not matter whether we first time reverse $x(z)$ and then apply $\tilde{\mathcal{P}}_C\left(.\right)$ or if we first apply $\tilde{\mathcal{P}}_C\left(.\right)$ and then time reverse $\tilde{x}(z)$.

Circular time reversal of vector Let $\tilde{\mathbf{x}}$ be defined as in (3.76). The elements of $\tilde{\mathbf{x}}$ are circular time reversed by premultiplication with the circulant exchange matrix

$$\mathbf{J}_{\mathrm{c}}\tilde{\mathbf{x}} = \left(x_0, x_{-1}, \ldots, x_{-T_{\mathrm{x}}}, 0, \ldots, 0, x_{T_{\mathrm{x}}}, \ldots, x_1\right)^T . \tag{3.109}$$

Circular time reversal of a circulant matrix Let $\tilde{\mathbf{X}} = \mathcal{C}\left(\tilde{\mathbf{x}}\right)$. Then $\mathcal{C}^{-1}(\tilde{\mathbf{X}}^T)$ returns a vector whose elements are circular time reversed, as in (3.109).

Circular correlation Complex conjugation and time reversal also play an important role for the circular convolution operation. Let $\tilde{x}$ and $\tilde{w}$ denote two zero-centered sequences of length C. Since $\tilde{w} \circledast \tilde{x}$ is a circular convolution, time reversing and complex conjugation of a sequence yields a circular correlation, e.g., $\tilde{\bar{w}}^* \circledast \tilde{x}$ or $\tilde{w} * \tilde{\bar{x}}^*$. Likewise $\tilde{\mathcal{P}}_C\left(w^*(z)\,x(z)\right)$ and $\tilde{\mathcal{P}}_C\left(w(z)\,x^*(z)\right)$ are circular correlations described by polynomials, $\mathbf{J}_{\mathrm{c}}\tilde{\mathbf{w}}^* \circledast \tilde{\mathbf{x}}$ and $\tilde{\mathbf{w}} \circledast \mathbf{J}_{\mathrm{c}}\tilde{\mathbf{x}}^*$ are circular correlations described by vectors, and $\tilde{\mathbf{W}}^H\tilde{\mathbf{X}}$ and $\tilde{\mathbf{W}}\tilde{\mathbf{X}}^H$ are circular correlations described by circulant matrices. $\mathbf{J}_{\mathrm{c}}$ is defined in (3.6). Special case: $\tilde{x} \circledast \tilde{\bar{x}}^* \cong \tilde{\mathbf{x}} \circledast \mathbf{J}_{\mathrm{c}}\tilde{\mathbf{x}}^* \cong \tilde{\mathbf{X}}\tilde{\mathbf{X}}^H$ corresponds to a circular autocorrelation.

Summary We have the following single-channel isomorphism

$$\tilde{x}(z^{-1}) \cong \mathbf{J}_{\mathrm{c}}\tilde{\mathbf{x}} \;\cong \tilde{\mathbf{X}}^T \tag{3.110}$$

$$\tilde{x}_*(z) \cong \tilde{\mathbf{x}}^* \;\cong \tilde{\mathbf{X}}^* \tag{3.111}$$

$$\tilde{x}^*(z) \cong \mathbf{J}_{\mathrm{c}}\tilde{\mathbf{x}}^* \cong \tilde{\mathbf{X}}^H . \tag{3.112}$$

3.4 Deconvolution

3.4.1 Linear deconvolution

Linear deconvolution of a sequence Let a be a given sequence. We define the deconvolution of a such that $a * a^{-1} = \{\ldots, 0, 1, 0, \ldots\} = \delta(n)$ becomes a zero-centered sequence. If a has finite length, a^{-1} has usually infinite length. However, for practical applications we truncate a^{-1} such that $\mathcal{P}_C\left(a^{-1}\right) * a \approx \{\ldots, 0, 1, 0, \ldots\}$.

Linear deconvolution of a polynomial Let $a(z)$ be the two sided z-transform of the sequence a with $\|a(z)\|_{\mathcal{F}} < \infty$. With $a^{-1}(z)$ we denote the *stable* inverse of $a(z)$ such that $a(z)\,a^{-1}(z) = 1$ and $\left\|a^{-1}(z)\right\|_{\mathcal{F}} < \infty$. We thereby require that no roots of $a(z)$ lie on the unit circle, $a(e^{j\omega}) \neq 0$ for $\omega \in [-\pi, \pi]$. Note, stability can be exchanged with non-causality. Let $w(z) = a^{-1}(z)$ and $a(z)$ be causal. If $a(z)$ is *minimum phase* (all roots lie inside the unit circle), then $a^{-1}(z) = \sum_{n=0}^{\infty} w_n z^{-n}$. If $a(z)$ is *maximum phase* (all roots lie outside the unit circle), then $a^{-1}(z) = \sum_{n=-\infty}^{-1} w_n z^{-n}$. If $a(z)$ is *mixed phase* (the roots lie in- and outside the unit circle), then $a^{-1}(z) = \sum_{n=-\infty}^{\infty} w_n z^{-n}$.

3.4.2 Circular deconvolution

Circular deconvolution of a sequence Let

$$\tilde{a} \triangleq \{0, \ldots, 0, a_{-N_{\mathrm{w}}}, \ldots, a_0, \ldots, a_{N_{\mathrm{a}}} 0, \ldots, 0\}_C \tag{3.113}$$

be a zero-centered sequence of length C. The circular deconvolution of $\tilde{a}$ is defined such that $\tilde{a} \circledast \tilde{a}^{-1} = \{\ldots, 0, 1, 0, \ldots\}_C$ is also zero-centered. Note, opposite to a^{-1} in the linear deconvolution, $\tilde{a}^{-1}$ is of finite length C.

Circular deconvolution of a polynomial Let $\tilde{a}(z) \triangleq \tilde{\mathcal{P}}_C\left(a(z)\right)$. From the linear deconvolution we have $a(z)\,a^{-1}(z) = 1$ by definition. Therefore, with (D.111),

$$\tilde{\mathcal{P}}_C\left(a(z)\,a^{-1}(z)\right) = 1 \tag{3.114}$$

$$\tilde{\mathcal{P}}_C\left(\tilde{a}(z)\,\tilde{\mathcal{P}}_C\left(a^{-1}(z)\right)\right) = 1 . \tag{3.115}$$

We define the circular-deconvolution polynomial as $\tilde{a}^{-1}(z) \triangleq \tilde{\mathcal{P}}_C\left(a^{-1}(z)\right)$. Furthermore, from (D.115) we have

$$\tilde{a}^{-1}(z) = \tilde{\mathcal{P}}_C\left(a^{-1}(z)\right) = \tilde{\mathcal{P}}_C\left(\left(\tilde{\mathcal{P}}_C\left(a(z)\right)\right)^{-1}\right) . \tag{3.116}$$

If C is chosen *large enough*, then $\tilde{a}(z)^{-1}$ is approximately equal to the C center elements of $a^{-1}(z)$, e.g.

$$a^{-1}(z) \approx \mathcal{P}_C\left(a^{-1}(z)\right) \approx \tilde{\mathcal{P}}_C\left(a^{-1}(z)\right) = \tilde{a}^{-1}(z) . \tag{3.117}$$

Circular deconvolution of a vector Let

$$\tilde{\mathbf{a}} \triangleq (a_0, \ldots, a_{N_{\mathrm{a}}}, 0, \ldots, 0, a_{-N_{\mathrm{a}}}, \ldots, a_{-1})^T \tag{3.118}$$

be a vector of length C. We define the circular deconvolution of a vector $\tilde{\mathbf{a}}$ such that

$$\tilde{\mathbf{a}}^{-1} \circledast \tilde{\mathbf{a}} = \mathbf{e}_1 \tag{3.119}$$

where $\mathbf{e}_1$ denotes the first unit vector. Extending both sides of (3.119) to a circulant matrix similar to (3.81) we obtain

$$\mathcal{C}(\tilde{\mathbf{a}}^{-1})\mathcal{C}(\tilde{\mathbf{a}}) = \mathbf{I} \tag{3.120}$$

$$\mathcal{C}(\tilde{\mathbf{a}}^{-1}) = (\mathcal{C}(\tilde{\mathbf{a}}))^{-1} \tag{3.121}$$

and therefore

$$\tilde{\mathbf{a}}^{-1} \triangleq \mathcal{C}^{-1}((\mathcal{C}(\tilde{\mathbf{a}}))^{-1}) . \tag{3.122}$$

In fact, (3.120) corresponds to $\tilde{\mathbf{A}}^{-1}\tilde{\mathbf{A}} = \mathbf{I}$ and therefore we see that $\tilde{\mathbf{A}}$ must have full rank, which is true if the elements of $\mathbf{F}\tilde{\mathbf{a}}$, the eigenvalues of $\tilde{\mathbf{A}}$, are non-zero. A fast implementation of (3.122), which is equal to $\tilde{\mathbf{a}}^{-1} = \mathbf{F}^{-1}\left(\mathbf{F}\tilde{\mathbf{a}}\right)^{((-1))}$, is given in [69, 71]

$$\tilde{\mathbf{a}}^{-1} = \mathrm{IFFT}\{(\mathrm{FFT}\,\{\tilde{\mathbf{a}}\})^{((-1))}\} \tag{3.123}$$

where $(.)^{((-1))}$ denotes the element-wise inversion of a vector.

Summary We have the following single-channel isomorphism

$$\tilde{x}^{-1}(z) \cong \mathcal{C}^{-1}(\mathcal{C}(\tilde{\mathbf{x}})^{-1}) \cong \tilde{\mathbf{X}}^{-1}\,. \tag{3.124}$$

3.5 Multichannel extension

We now extend the operators, described in the previous subsections for the single-channel case, to the multichannel case. Since we have seen that many linear-time operations can be efficiently implemented or approximated by the corresponding circular-time operations, we will focus only on the extension of the circular-time operations.

Roughly speaking, a polynomial is replaced by a polynomial matrix and a circulant matrix is replaced by block-circulant matrix.

3.5.1 Multichannel convolution

This subsection is the multichannel extension of Section 3.2.

Multichannel linear convolution of two polynomial matrices Let

$$\mathbf{x}(z) = \sum_{t=-\infty}^{\infty} \mathbf{x}_t z^{-t} \tag{3.125}$$

$$\mathbf{W}(z) = \sum_{n=-\infty}^{\infty} \mathbf{W}_n z^{-n}. \tag{3.126}$$

be two Laurent-series or polynomial matrices. The multichannel linear convolution is defined as

$$\mathbf{u}(z) = \mathbf{W}(z)\mathbf{x}(z)\,. \tag{3.127}$$

Multichannel circular convolution of two polynomial matrices Let

$$\tilde{\mathbf{x}}(z) = \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} \mathbf{x}_t z^{-t} \tag{3.128}$$

$$\tilde{\mathbf{W}}(z) = \sum_{n=-N_{\mathrm{w}}}^{N_{\mathrm{w}}} \mathbf{W}_n z^{-n}\,. \tag{3.129}$$

The multichannel circular convolution is defined as

$$\tilde{\mathbf{u}}(z) = \tilde{\mathcal{P}}_C\left(\tilde{\mathbf{W}}(z)\tilde{\mathbf{x}}(z)\right)\,. \tag{3.130}$$

Similar to the single-channel case described in Section 3.2.2, we distinguish between three different cases depending on C, N_{w}, and T_{x}.

Multichannel circular convolution with block circulant matrices Let $\widetilde{\mathbf{X}}^{MC\times C} = [\tilde{\mathbf{X}}_m] = [\mathcal{C}(\tilde{\mathbf{x}}_m)]$ and $\widetilde{\mathbf{W}}^{MC\times MC} = [\tilde{\mathbf{W}}_{mn}] = [\mathcal{C}(\tilde{\mathbf{w}}_{mn})]$ be two block circulant matrices. We can define the multichannel circular convolution as

$$\widetilde{\mathbf{U}} = \widetilde{\mathbf{W}}\,\widetilde{\mathbf{X}} \tag{3.131}$$

where $\widetilde{\mathbf{U}}^{MC\times C} = [\tilde{\mathbf{U}}_m] = [\mathcal{C}(\tilde{\mathbf{u}}_m)]$ is also a block circulant matrix. Pre- and post-multiplying (3.131) with $\mathbf{T}_M$ and $\mathbf{T}_1^{-1}$, where $\mathbf{T}_M$ is defined in (3.7), yields

$$\mathbf{T}_M\widetilde{\mathbf{U}}\mathbf{T}_1^{-1} = \mathbf{T}_M\widetilde{\mathbf{W}}\mathbf{T}_M^{-1}\mathbf{T}_M\widetilde{\mathbf{X}}\mathbf{T}_1^{-1} \tag{3.132}$$

$$\overline{\mathbf{U}} = \overline{\mathbf{W}}\,\overline{\mathbf{X}} \tag{3.133}$$

where $\overline{\mathbf{U}} = \mathbf{T}_M\widetilde{\mathbf{U}}\mathbf{T}_1^{-1}$, $\overline{\mathbf{X}} = \mathbf{T}_M\widetilde{\mathbf{X}}\mathbf{T}_1^{-1}$, and $\overline{\mathbf{W}} = \mathbf{T}_M\widetilde{\mathbf{W}}\mathbf{T}_M^{-1}$ are all block diagonal matrices, see Section 3.1.7.

We now analyze the computational complexity of (3.131) and (3.133). The direct computation of (3.131) requires M^2C^3 multiplications and $M^2C^3 - MC^2$ additions. Since $\widetilde{\mathbf{U}}$ is a block circulant matrix, it suffices to compute only the first row of $\widetilde{\mathbf{U}}$, because the remaining $C-1$ rows are only permutations of the first row. Therefore, the computation reduces to M^2C^2 multiplications and $M^2C^2 - MC$ additions. However, the computation of (3.133), the product of two block diagonal matrices, requires only M^2C multiplications and $M^2C -$

MC additions. This is about an order of magnitude lower than with (3.131), as we usually have $C \gg M$. In addition, we need M FFTs for the computation of $\overline{\mathbf{X}}$, M IFFTs for $\widetilde{\mathbf{U}}$, and M^2 FFTs for $\overline{\mathbf{W}}$, if $\overline{\mathbf{W}}$ is not already available in the frequency domain. Since the complexity of an FFT / IFFT is about $C \log C$, which grows lower than C^2, it is still worthwhile to take (3.133) for large values of C.

Summary We have the following isomorphism

$$\tilde{\mathbf{u}}(z) = \tilde{\mathcal{P}}_C \left(\tilde{\mathbf{W}}(z) \tilde{\mathbf{x}}(z) \right) \;\cong\; \widetilde{\mathbf{U}} = \widetilde{\mathbf{W}} \widetilde{\mathbf{X}} \;\cong\; \overline{\mathbf{U}} = \overline{\mathbf{W}}\,\overline{\mathbf{X}}. \tag{3.134}$$

3.5.2 Complex conjugation, time reversal, and correlation

This subsection is the multichannel extension of Section 3.3.

Polynomial matrices Let

$$\mathbf{X}(z) = \sum_{n=N_1}^{N_2} \mathbf{X}_n z^{-n} \tag{3.135}$$

be a polynomial matrix or a matrix polynomial. We then have the following relations

$$\mathbf{X}(z^{-1}) = \sum_{n=N_1}^{N_2} \mathbf{X}_n z^{+n} = \sum_{n=-N_2}^{-N_1} \mathbf{X}_{-n} z^{-n} \tag{3.136}$$

$$\mathbf{X}_*(z) = \sum_{n=N_1}^{N_2} \mathbf{X}_n^* z^{-n} \tag{3.137}$$

$$\mathbf{X}^*(z) = \sum_{n=N_1}^{N_2} \mathbf{X}_n^* z^{+n} = \sum_{n=-N_2}^{-N_1} \mathbf{X}_{-n}^* z^{-n} = \mathbf{X}_*(z^{-1}) \tag{3.138}$$

$$\mathbf{X}^T(z) = \sum_{n=N_1}^{N_2} \mathbf{X}_n^T z^{-n} \tag{3.139}$$

$$\mathbf{X}^H(z) = \sum_{n=N_1}^{N_2} \mathbf{X}_n^H z^{+n} = \sum_{n=-N_2}^{-N_1} \mathbf{X}_{-n}^H z^{-n} = \mathbf{X}_*^T(z^{-1}). \tag{3.140}$$

Block circulant matrices Let $\widetilde{\mathbf{X}}^{MC \times NC} = [\tilde{\mathbf{X}}_{mn}] = [\mathcal{C}(\tilde{\mathbf{x}}_{mn})]$ be a block circulant matrix. We then have $\widetilde{\mathbf{X}}^T = [\tilde{\mathbf{X}}_{nm}^T] = [\mathcal{C}(\mathbf{J}_\mathrm{c} \tilde{\mathbf{x}}_{nm})]$, $\widetilde{\mathbf{X}}^* = [\tilde{\mathbf{X}}_{mn}^*] = [\mathcal{C}(\tilde{\mathbf{x}}_{mn}^*)]$, and $\widetilde{\mathbf{X}}^H = [\tilde{\mathbf{X}}_{nm}^H] = [\mathcal{C}(\mathbf{J}_\mathrm{c} \tilde{\mathbf{x}}_{nm}^*)]$.

Circular correlation Let $\tilde{\mathbf{W}}(z)$ and $\tilde{\mathbf{X}}(z)$ be two polynomial matrices, and $\widetilde{\mathbf{W}}$ and $\widetilde{\mathbf{X}}$ be two block circulant matrices, where $\tilde{\mathbf{W}}(z) \cong \widetilde{\mathbf{W}}$ and $\tilde{\mathbf{X}}(z) \cong \widetilde{\mathbf{X}}$. Then $\tilde{\mathcal{P}}_C \left(\tilde{\mathbf{W}}^H(z)\, \tilde{\mathbf{X}}(z) \right) \cong \widetilde{\mathbf{W}}^H \widetilde{\mathbf{X}}$ and $\tilde{\mathcal{P}}_C \left(\tilde{\mathbf{W}}(z)\, \tilde{\mathbf{X}}^H(z) \right) \cong \widetilde{\mathbf{W}} \widetilde{\mathbf{X}}^H$ are circular correlations described by polynomial matrices and block circulant matrices, respectively, if the products exist. The special case $\tilde{\mathcal{P}}_C \left(\tilde{\mathbf{X}}(z)\, \tilde{\mathbf{X}}^H(z) \right) \cong \widetilde{\mathbf{X}} \widetilde{\mathbf{X}}^H$ corresponds to a circular autocorrelation.

3.5.3 Multichannel deconvolution

This subsection is the multichannel extension of Section 3.4.

Multichannel linear deconvolution of a polynomial matrix Let $\mathbf{A}(z)$ be a square polynomial matrix with $\|\mathbf{A}(z)\|_{\mathcal{F}} < \infty$. With $\mathbf{A}^{-1}(z)$ we denote the *stable* inverse of $\mathbf{A}(z)$, i.e. $\left\|\mathbf{A}^{-1}(z)\right\|_{\mathcal{F}} < \infty$, such that $\mathbf{A}(z)\mathbf{A}^{-1}(z) = \mathbf{I}$. We require that no roots of $\det \mathbf{A}(z)$ lie on the unit circle, i.e., $\det \mathbf{A}(e^{j\omega}) \neq 0$ for $\omega \in [-\pi, \pi]$. Depending on the roots of $\det \mathbf{A}(z)$, we can also distinguish between a minimum-, maximum-, or mixed-phase system, just as for a polynomial.

If $\mathbf{A}(z)$ is a rectangular matrix, we can define in a similar way the Moore-Penrose pseudoinverse $\mathbf{A}^{\#}(z)$.

Multichannel circular deconvolution of a polynomial matrix Let $\tilde{\mathbf{A}}(z) \triangleq \tilde{\mathcal{P}}_C\left(\mathbf{A}(z)\right)$ be a square polynomial matrix. From the multichannel linear deconvolution we have $\mathbf{A}(z)\,\mathbf{A}^{-1}(z) = \mathbf{I}$ by definition. Therefore, with (D.111),

$$\tilde{\mathcal{P}}_C \left(\mathbf{A}(z)\, \mathbf{A}^{-1}(z) \right) = \mathbf{I} \tag{3.141}$$

$$\tilde{\mathcal{P}}_C \left(\tilde{\mathbf{A}}(z)\, \tilde{\mathcal{P}}_C \left(\mathbf{A}^{-1}(z) \right) \right) = \mathbf{I}. \tag{3.142}$$

We define $\tilde{\mathbf{A}}^{-1}(z) \triangleq \tilde{\mathcal{P}}_C\left(\mathbf{A}^{-1}(z)\right)$ to be the circular-deconvolution polynomial matrix. Furthermore, from (D.115) we have

$$\tilde{\mathbf{A}}^{-1}(z) = \tilde{\mathcal{P}}_C\left(\mathbf{A}^{-1}(z)\right) = \tilde{\mathcal{P}}_C\left(\left(\tilde{\mathcal{P}}_C\left(\mathbf{A}(z)\right)\right)^{-1}\right). \tag{3.143}$$

If C is chosen *large enough*, then

$$\mathbf{A}^{-1}(z) \approx \mathcal{P}_C\left(\mathbf{A}^{-1}(z)\right) \approx \tilde{\mathcal{P}}_C\left(\mathbf{A}^{-1}(z)\right) = \tilde{\mathbf{A}}^{-1}(z). \tag{3.144}$$

Circular deconvolution of a block circulant matrix Let $\tilde{\mathbf{A}}(z) \triangleq \tilde{\mathcal{P}}_C\left(\mathbf{A}(z)\right)$ be an $M \times M$ polynomial matrix and $\widetilde{\mathbf{A}}$ be an $MC \times MC$ block circulant matrix, with $\tilde{\mathbf{A}}_{mn} = [\widetilde{\mathbf{A}}]_{mn} = \mathcal{C}(\tilde{\mathbf{a}}_{mn})$, and

$$\tilde{\mathbf{a}}_{mn} \triangleq \left(a_{mn,0}, \ldots, a_{mn,N_a}, 0, \ldots, 0, a_{mn,-N_a}, \ldots, a_{mn,-1}\right)^T \tag{3.145}$$

containing the coefficients of $[\tilde{\mathbf{A}}(z)]_{mn}$. We then have the isomorphism

$$\tilde{\mathbf{A}}^{-1}(z) \triangleq \tilde{\mathcal{P}}_C\left(\mathbf{A}^{-1}(z)\right) \cong \widetilde{\mathbf{A}}^{-1}. \tag{3.146}$$

Fast inversion of a block circulant matrix We can decompose the block circulant matrix $\widetilde{\mathbf{A}}$ into

$$\widetilde{\mathbf{A}} = \mathbf{T}_M^{-1}\,\overline{\mathbf{A}}\,\mathbf{T}_M \tag{3.147}$$

$$[\,\widetilde{\mathbf{A}}\,]_{mn} = \mathrm{diag}\left(\mathbf{F}^{-1}\,\bar{\mathbf{a}}_{mn}\right) \tag{3.148}$$

where $\overline{\mathbf{A}}$ is a block diagonal matrix and $\mathbf{T}_M$ is defined in (3.7). From (3.147) we then have

$$\overline{\mathbf{A}} = \mathbf{T}_M\,\widetilde{\mathbf{A}}\,\mathbf{T}_M^{-1} \tag{3.149}$$

$$[\,\overline{\mathbf{A}}\,]_{mn} = \mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{a}}_{mn}\right). \tag{3.150}$$

Inverting both sides of (3.147) yields

$$\widetilde{\mathbf{A}}^{-1} = \mathbf{T}_M^{-1}\,\overline{\mathbf{A}}^{-1}\,\mathbf{T}_M. \tag{3.151}$$

From (3.148), (3.150), and (3.151) we see, that the inversion of the block circulant matrix $\widetilde{\mathbf{A}}$ can be done by M^2 FFTs, M^2 IFFTs, and an inversion of a block diagonal matrix $\overline{\mathbf{A}}$. There exist fast algorithms to invert a block diagonal matrix, which require only C matrix inversions of an $M \times M$ matrix. This is due to the sparseness of a block diagonal matrix and because $\overline{\mathbf{A}}^{-1}$ is again a block diagonal matrix with only M^2C non-zero elements.

If $\tilde{\mathbf{A}}(z) \triangleq \tilde{\mathcal{P}}_C\left(\mathbf{A}(z)\right)$ is an $M \times N$ rectangular polynomial matrix, we can proceed in a similar way, i.e. $\tilde{\mathcal{P}}_C\left(\mathbf{X}^{\#}(z)\right) \cong \widetilde{\mathbf{X}}^{\#}$.

3.6 Summary

In the limit where C goes towards infinity, all the operations in the circular time domain coincide with those in the linear time domain, e.g., convolution, deconvolution, correlation, time reversal, etc. Actually, many operations can be implemented more efficiently in the circular time domain, for instance circular convolution with the help of the FFT / IFFT. Other operations are inherently less complex in their computation. For example for the circular deconvolution we have to compute only a finite number of elements, whereas the linear deconvolution requires in general an infinite number.

For practical applications, however, it is mostly sufficient if the operations in the linear time domain are well approximated. Therefore, if in the circular time domain C is chosen *large enough*, the operations in the linear time domain can be sufficiently well approximated.

Circulant and block circulant matrices have shown to be very useful for the computation of many operations which appear in multichannel signal processing, not only because of the isomorphic mapping between polynomial matrices and block circulant matrices, but also because of their interesting properties related to Linear Algebra. Block circulant matrices are also very closely related to FIR matrices, within an FIR-matrix algebra, as pointed out by Lambert [69,71].

A summary of single-channel and multichannel isomorphisms between polynomial matrices and (block) circulant matrices are given in Fig. 3.1 and Fig. 3.2, respectively.

operation	polynomials	circulant matrices	diagonal matrices
	$w(z)$	$\tilde{\mathbf{W}}$	$\bar{\mathbf{W}} = \mathbf{F}\,\tilde{\mathbf{W}}\,\mathbf{F}^{-1}$
time reversal	$w(z^{-1})$	$\tilde{\mathbf{W}}^T$	$\mathbf{J}_c\,\bar{\mathbf{W}}\,\mathbf{J}_c$
complex conjugation	$w_*(z)$	$\tilde{\mathbf{W}}^*$	$\mathbf{J}_c\,\bar{\mathbf{W}}^*\,\mathbf{J}_c$
	$w^*(z) = w_*(z^{-1})$	$\tilde{\mathbf{W}}^H$	$\bar{\mathbf{W}}^*$
circ. inversion	$\tilde{\mathcal{P}}_C\,(\,w^{-1}(z)\,)$	$\tilde{\mathbf{W}}^{-1}$	$\bar{\mathbf{W}}^{-1}$
circ. convolution	$\tilde{\mathcal{P}}_C\,(\,w(z)\;x(z)\,)$	$\tilde{\mathbf{W}}\,\tilde{\mathbf{X}}$	$\bar{\mathbf{W}}\,\bar{\mathbf{X}}$

Table 3.1: Isomorphic mapping — single-channel case.

polynomials	block circulant matrices	block diagonal matrices
$\mathbf{W}(z)$	$\widetilde{\mathbf{W}} = [\,\tilde{\mathbf{W}}_{mn}\,]$	$\overline{\mathbf{W}} = (\mathbf{I}\otimes\mathbf{F})\,\widetilde{\mathbf{W}}\,(\mathbf{I}\otimes\mathbf{F}^{-1})$
$\mathbf{W}(z^{-1})$	$[\,\tilde{\mathbf{W}}^T_{mn}\,]$	$(\mathbf{I}\otimes\mathbf{J}_c)\,\overline{\mathbf{W}}\,(\mathbf{I}\otimes\mathbf{J}_c)$
$\mathbf{W}^T(z)$	$[\,\tilde{\mathbf{W}}_{nm}\,]$	$\overline{\mathbf{W}}^T$
$\mathbf{W}_*(z)$	$\widetilde{\mathbf{W}}^*$	$(\mathbf{I}\otimes\mathbf{J}_c)\,\overline{\mathbf{W}}^*\,(\mathbf{I}\otimes\mathbf{J}_c)$
$\mathbf{W}^H(z) = \mathbf{W}^T_*(z^{-1})$	$\widetilde{\mathbf{W}}^H = [\,\tilde{\mathbf{W}}^H_{nm}\,]$	$\overline{\mathbf{W}}^H$
$\mathbf{W}^T(z^{-1})$	$\widetilde{\mathbf{W}}^T = [\,\tilde{\mathbf{W}}^T_{nm}\,]$	$(\mathbf{I}\otimes\mathbf{J}_c)\,\overline{\mathbf{W}}^T\,(\mathbf{I}\otimes\mathbf{J}_c)$
$\mathbf{W}_*(z^{-1})$	$[\,\tilde{\mathbf{W}}^H_{mn}\,]$	$\overline{\mathbf{W}}^*$
$\mathbf{W}^T_*(z)$	$[\,\tilde{\mathbf{W}}^*_{nm}\,]$	$(\mathbf{I}\otimes\mathbf{J}_c)\,\overline{\mathbf{W}}^H\,(\mathbf{I}\otimes\mathbf{J}_c)$
$\tilde{\mathcal{P}}_C\,(\,\mathbf{W}^{-1}(z)\,)$	$\widetilde{\mathbf{W}}^{-1}$	$\overline{\mathbf{W}}^{-1}$
$\tilde{\mathcal{P}}_C\,(\,\mathbf{W}^{\#}(z)\,)$	$\widetilde{\mathbf{W}}^{\#}$	$\overline{\mathbf{W}}^{\#}$
$\tilde{\mathcal{P}}_C\,(\,\mathbf{W}(z)\;\mathbf{x}(z)\,)$	$\widetilde{\mathbf{W}}\,\tilde{\mathbf{X}}$	$\overline{\mathbf{W}}\,\overline{\mathbf{X}}$

Table 3.2: Isomorphic mapping — multichannel case.

Chapter 4

Single-channel identification and inverse modeling

For a single-input system, system identification and inverse modeling of an instantaneous mixing system, described in Chapter 1, degenerates to a single-channel gain and inverse-gain estimation. We analyze this problem in more detail, as this is also the most simple case of a single-channel system identification and also of inverse modeling of an FIR filter. We thereby investigate three different learning concepts for the adaptation, namely online, batch (off line), and block-wise learning. Afterwards, we extend these concepts to the case where the unknown system consists of an FIR filter.

Usually, the assumption is made that the multichannel filters $\mathbf{A}(z)$, $\mathbf{H}(z)$, $\mathbf{W}(z)$, and also the signal vectors $\mathbf{s}(z)$, $\mathbf{x}(z)$, $\hat{\mathbf{x}}(z)$, and $\mathbf{u}(z)$ are described by two-sided Laurent series with infinitely many terms, e.g.

$$a(z) = \sum_{k=-\infty}^{\infty} a_k z^{-k} . \tag{4.1}$$

As an example we choose the LMS1-Hx, whose update equation for the identification of a mixing matrix is given by $\mathbf{H}_{t+1} = \mathbf{H}_t + \mu_t \mathbf{e}_{\mathrm{x}t}\mathbf{s}_t^H$. The natural extension of the problem is to replace $\mathbf{A}$ by $\mathbf{A}(z)$, which is referred to as a *convolutive mixing* of the source signals. Intuitively, we expect the extension of the update equation to become $\mathbf{H}_{k+1}(z) = \mathbf{H}_k(z) + \mu_k \mathbf{e}_{\mathrm{x}k}(z)\mathbf{s}^H(z)$, where

$\mathbf{s}^H(z) = \mathbf{s}_*(z^{-1})$ and k denotes the iteration index. In other words, every matrix and vector in the system model or update equation is replaced by a polynomial matrix. In reality, however, we deal with filters and time series of finite length and we do not exactly know, if this simple extension still holds, and if not, how we have to modify the update equations.

The use of the polynomial projection operator $\mathcal{P}(.)$, defined in Appendix D.2, allows a compact description of the system and the update equations. The equations can then be easily transformed into a matrix / vector notation (isomorphism), which is more convenient for an implementation. Since we assume long filters, which is common in acoustical signal processing, the time-consuming convolution can be carried out in the frequency domain by exploiting the efficiency of the so-called fast convolution.

In this chapter we derive the basic framework for single-channel system identification and inverse modeling of an FIR filter, based on overlap-save techniques. The update equations are derived from those of Chapter 2 and are summarized in Table E.5 and Table E.9. It is interesting to see that many of them are related to known algorithms for single-channel blind deconvolution.

4.1 Identification of an unknown gain

Before we start to analyze the general single-channel case, where the unknown system is modeled as an FIR filter, we analyze the most simple system, namely one described by a simple complex gain. We introduce the use of the polynomial projection operator defined in Appendix D.2 to describe the system model and to derive the update equations.

4.1.1 Model

In the single-channel case, we can describe the unknown system by a complex scalar a. The known input sequence is described by its two-sided z-transform

$$s(z) \triangleq \sum_{t=-T_\mathrm{s}}^{T_\mathrm{s}} s_t z^{-t} \tag{4.2}$$

where the length of the input sequence is $2T_\mathrm{s}+1$. The input-output behavior is then described by

$$x_t = a s_t + n_t = z^t \, \mathcal{P}_{t,t}\left(a s(z) + n(z)\right) \tag{4.3}$$

$$x(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} x_t z^{-t} = \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \mathcal{P}_{t,t}\left(a s(z) + n(z)\right) \tag{4.4}$$

$$= \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}\left(a s(z) + n(z)\right) \tag{4.5}$$

where $x(z)$ is the known output sequence of finite length $2T_\mathrm{x}+1$ and $n(z)$ is the sensor noise. Note that we define the observation time to be symmetric around the time origin, i.e. from $-T_\mathrm{x}$ to $+T_\mathrm{x}$. We allow that $T_\mathrm{x} \neq T_\mathrm{s}$, however, we require that $T_\mathrm{s} \geq T_\mathrm{x}$. $\mathcal{P}(.)$ denotes the polynomial projection operator defined in Appendix D.2.

We give three different methods to identify the unknown gain a: (i) an online learning algorithm, where the update is carried out sample by sample, (ii) a batch algorithm, where the update depends on the estimation error of the whole data sequence, and (iii) a block-wise algorithm, where the update is carried out block wise.

4.1.2 Online learning algorithm

With an online learning algorithm we process the estimation and adaptation on a sample rate basis. Let $h = \hat{a}$ denote the estimation of a. We then have

$$\hat{x}_t = h_t s_t = z^t \, \mathcal{P}_{t,t}\left(h_t s_t\right) \tag{4.6}$$

and the estimation error

$$e_{\mathrm{x}t} \triangleq x_t - \hat{x}_t \,. \tag{4.7}$$

For the update at discrete time t we aim at minimizing $|e_{\mathrm{x}t}|^2$. As an example, if we use LMS1-Hx we have

$$h_{t+1} = h_t + \mu e_{\mathrm{x}t} s_t^* \tag{4.8}$$

$$= h_t + \mu \, \mathcal{P}_{t,t}\left(e_\mathrm{x}(z)\right) \, \mathcal{P}_{t,t}^*\left(s(z)\right) \tag{4.9}$$

$$= h_t + \mu \, \mathcal{P}_{0,0}\left(e_{\mathrm{x}t} z^{-t} s^*(z)\right) \tag{4.10}$$

$$= \mathcal{P}_{0,0}\left(h_t + \mu \, e_{\mathrm{x}t} z^{-t} s^*(z)\right) \tag{4.11}$$

where we used $\mathcal{P}_{t,t}\left(e_\mathrm{x}(z)\right) = e_{\mathrm{x}t} z^{-t}$, $\mathcal{P}_{t,t}^*\left(s(z)\right) = s_t^* z^{+t}$, and $\mathcal{P}_{0,0}\left(h_t\right) = h_t$. Other update equations for gain identification are given in Table E.4.

4.1.3 Batch learning algorithm

With a batch learning algorithm we use the estimation error of the whole data sequence for the update of $h = \hat{a}$. Let k denote the iteration index. At iteration k, the estimation sequence $\hat{x}_k(z)$ is

$$\hat{x}_{t,k} = h_k s_t = z^t \mathcal{P}_{t,t}(h_k s_t) \tag{4.12}$$

$$\hat{x}_k(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \hat{x}_{t,k} z^{-t} \tag{4.13}$$

$$= \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}(h_k s(z)) \tag{4.14}$$

and the estimation error sequence $e_{\mathrm{x}k}(z)$ is

$$e_{\mathrm{x}t,k} \triangleq x_t - \hat{x}_{t,k} \tag{4.15}$$

$$e_{\mathrm{x}k}(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} e_{\mathrm{x}t,k} z^{-t} \tag{4.16}$$

$$= x(z) - \hat{x}_k(z). \tag{4.17}$$

At iteration k we wish to minimize $\|e_{\mathrm{x}k}(z)\|_{\mathcal{F}}^2$ where the norm $\|.\|_{\mathcal{F}}$ is defined in (C.18). Batch learning is equal to averaging the update over the whole data sequence of $L = 2T_\mathrm{x} + 1$ samples. As an example, if we use LMS1-Hx and average the update (4.8) over the whole sequence, we get

$$h_{k+1} = h_k + \frac{\mu}{2T_\mathrm{x}+1} \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} e_{\mathrm{x}t,k} s_t^* \tag{4.18}$$

$$= h_k + \frac{\mu}{2T_\mathrm{x}+1} \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \mathcal{P}_{t,t}(e_{\mathrm{x}k}(z))\, \mathcal{P}_{t,t}^*(s(z)) \tag{4.19}$$

$$= h_k + \frac{\mu}{2T_\mathrm{x}+1} \mathcal{P}_{0,0}(e_{\mathrm{x}k}(z) s^*(z)) \tag{4.20}$$

$$= \mathcal{P}_{0,0}\left(h_k + \frac{\mu}{2T_\mathrm{x}+1} e_{\mathrm{x}k}(z) s^*(z)\right) \tag{4.21}$$

where we used (D.43), $\mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}(e_{\mathrm{x}k}(z)) = e_{\mathrm{x}k}(z)$, and $\mathcal{P}_{0,0}(h_k) = h_k$. Other batch learning algorithms can be derived with Table E.4, analogously to LMS1-Hx.

4.1.4 Block-wise learning algorithm

We now modify the system model slightly such that the input, the noise, and the output sequence have infinite length. The system output is still given by $x_t = as_t + n_t$. We partition the output sequence into consecutive, non-overlapping blocks of block length $L = 2T_\mathrm{x}+1$ samples. With k we denote the block index. We can then describe the output sequence as $x(z) = \sum_{k=-\infty}^{\infty} x_k(z) z^{-kL}$ with

$$x_k(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} x_{kL+t} z^{-t} = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x},}(as_k(z) + n_k(z)) \tag{4.22}$$

where

$$s_k(z) \triangleq \sum_{t=-T_\mathrm{s}}^{T_\mathrm{s}} s_{kL+t} z^{-t} = \mathcal{P}_{-T_\mathrm{s},T_\mathrm{s},}\left(z^{kL} s(z)\right) \tag{4.23}$$

is the input sequence of block k and

$$n_k(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} n_{kL+t} z^{-t} = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x},}\left(z^{kL} n(z)\right) \tag{4.24}$$

is the noise sequence of block k. Note, we have $n(z) = \sum_{k=-\infty}^{\infty} n_k(z) z^{-kL}$, however, $s(z) = \sum_{k=-\infty}^{\infty} s_k(z) z^{-kL}$ is only true if $T_\mathrm{s} = T_\mathrm{x}$. Usually we have $T_\mathrm{s} > T_\mathrm{x}$. Analogously, we can partition the estimation sequence as $\hat{x}(z) = \sum_{k=-\infty}^{\infty} \hat{x}_k(z) z^{-kL}$ where at block k we have

$$\hat{x}_{t,k} = h_k s_t = z^t \mathcal{P}_{t,t}(h_k s(z)) \tag{4.25}$$

$$\hat{x}_k(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \hat{x}_{kL+t,k} z^{-t} = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x},}(h_k s_k(z)). \tag{4.26}$$

Accordingly, the estimation error sequence can be written as $e_\mathrm{x}(z) = \sum_{k=-\infty}^{\infty} e_{\mathrm{x}k}(z) z^{-kL}$ with

$$e_{\mathrm{x}k}(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} e_{\mathrm{x}kL+t,k} z^{-t} \tag{4.27}$$

$$= x_k(z) - \hat{x}_k(z) \tag{4.28}$$

$$= \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x},}([a - h_k] s_k(z) + n_k(z)). \tag{4.29}$$

The update is performed at the block rate, where in block k we aim at minimizing $||e_{\mathrm{x}k}(z)||^2_{\mathcal{F}}$. This is equal to averaging the update over a whole block of $L = 2T_{\mathrm{x}} + 1$ samples. As an example, if we use LMS1-Hx we have

$$h_{k+1} = h_k + \frac{\mu}{2T_{\mathrm{x}}+1} \sum_{t=kL-T_{\mathrm{x}}}^{kL+T_{\mathrm{x}}} e_{\mathrm{x}t} s_t^* \tag{4.30}$$

$$= h_k + \frac{\mu}{2T_{\mathrm{x}}+1} \sum_{t=kL-T_{\mathrm{x}}}^{kL+T_{\mathrm{x}}} \mathcal{P}_{t,t}\left(e_{\mathrm{x}k}(z) z^{-kL}\right) \mathcal{P}^*_{t,t}\left(s_k(z) z^{-kL}\right) \tag{4.31}$$

$$= h_k + \frac{\mu}{2T_{\mathrm{x}}+1} \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} \mathcal{P}_{t,t}\left(e_{\mathrm{x}k}(z)\right) \mathcal{P}^*_{t,t}\left(s_k(z)\right) \tag{4.32}$$

$$= h_k + \frac{\mu}{2T_{\mathrm{x}}+1} \mathcal{P}_{0,0}\left(e_{\mathrm{x}k}(z) s_k^*(z)\right) \tag{4.33}$$

$$= \mathcal{P}_{0,0}\left(h_k + \frac{\mu}{2T_{\mathrm{x}}+1} e_{\mathrm{x}k}(z) s_k^*(z)\right) \tag{4.34}$$

Other block-wise learning algorithms can be derived analogously to LMS1-Hx with the sample-wise update equations of Table E.4.

The block-wise learning algorithm is different from the batch algorithm in the sense that instead of using the same input block of $2T_{\mathrm{x}}+1$ samples for every iteration, a new input block with $2T_{\mathrm{x}} + 1$ samples is taken for every iteration. Furthermore, by setting T_{x} to zero ($L = 1$) the block-wise learning algorithm degenerates to the online learning algorithm of Section 4.1.2.

4.2 Single-channel identification

We now extend the model of the unknown single-channel system from being an unknown gain a to an FIR filter $a(z)$, see Fig. 4.1.

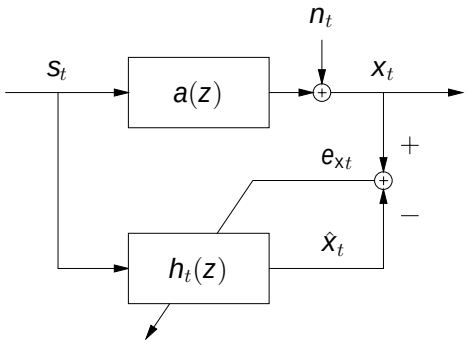

Figure 4.1: Single-channel identification. The filter $h(z)$ is adapted such that $h(z) \approx a(z)$.

4.2.1 Model

For the single-channel case, we can describe the unknown system by a single polynomial, the two sided z-transform

$$a(z) \triangleq \sum_{n=-N_{\mathrm{a}}}^{N_{\mathrm{a}}} a_n z^{-n}. \tag{4.35}$$

With this definition, we allow the system to be non-causal, as $a(z)$ also has positive powers of z. For a causal system we have $a_{-1} = \cdots = a_{-N_{\mathrm{a}}} = 0$, or $\mathcal{P}_{-N_{\mathrm{a}},-1}\left(a(z)\right) = 0$. The known input sequence $s(z)$ is given in (4.2). The input-output behavior is then described by

$$x_t = z^t \mathcal{P}_{t,t}\left(a(z)s(z) + n(z)\right) \tag{4.36}$$

$$x(z) \triangleq \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} x_t z^{-t} = \mathcal{P}_{-T_{\mathrm{x}},T_{\mathrm{x}}}\left(a(z)s(z) + n(z)\right) \tag{4.37}$$

where $x(z)$ is the output sequence of finite length $2T_{\mathrm{x}} + 1$ and $n(z)$ is the sensor-noise sequence. The description is similar to the one in Section 4.1.1 except that a is replaced by $a(z)$.

The error criterion which we aim at minimizing is

$$E\left\{|e_{\mathrm{x}t}|^2\right\} = ||a(z) - h_t(z)||^2_{\mathcal{F}} \sigma_{\mathrm{s}}^2 + \sigma_{\mathrm{n}}^2 \tag{4.38}$$

with $\sigma_{\mathrm{s}}^2 \triangleq E\left\{|s_t|^2\right\}$, $\sigma_{\mathrm{n}}^2 \triangleq E\left\{|n_t|^2\right\}$, and the assumption that $s(z)$ and $n(z)$ are uncorrelated, i.e. $\langle s(z), n(z)\rangle_{\mathcal{F}} = 0$. This is equal to minimizing $\|a(z) - h_t(z)\|_{\mathcal{F}}^2$.

Just as in Section 4.1, we now describe an online, a batch, and a block-wise learning algorithm for identification, with the extension, that an FIR filter is adapted.

4.2.2 Online learning algorithm

With an online learning algorithm we process the estimation and adaptation at the sample rate. Let $h(z) = \hat{a}(z)$ denote the estimation of $a(z)$

$$h(z) \triangleq \sum_{n=-N_{\mathrm{h}}}^{N_{\mathrm{h}}} h_n z^{-n} \tag{4.39}$$

with $N_{\mathrm{h}} \leq N_{\mathrm{a}}$. We then have

$$\hat{x}_t = z^t \, \mathcal{P}_{t,t}\left(h_t(z) s(z)\right) \tag{4.40}$$

where $h_t(z)$ is the estimate of $a(z)$ at time t and

$$e_{\mathrm{x}t} \triangleq x_t - \hat{x}_t \tag{4.41}$$

is the corresponding estimation error at discrete time t. For the update at t we aim at minimizing $|e_{\mathrm{x}t}|^2$. As an example, if we use LMS1-Hx we have

$$h_{n,t+1} = h_{n,t} + \mu e_{\mathrm{x}t} s_{t-n}^* \tag{4.42}$$

$$h_{t+1}(z) = h_t(z) + \mu \, \mathcal{P}_{t,t}\left(e_{\mathrm{x}}(z)\right) \, \mathcal{P}_{t-N_{\mathrm{h}},t+N_{\mathrm{h}}}^*\left(s(z)\right) \tag{4.43}$$

$$= h_t(z) + \mu \, \mathcal{P}_{-N_{\mathrm{h}},N_{\mathrm{h}}}\left(e_{\mathrm{x}t} z^{-t} s^*(z)\right) \tag{4.44}$$

$$= \mathcal{P}_{-N_{\mathrm{h}},N_{\mathrm{h}}}\left(h_t(z) + \mu e_{\mathrm{x}t} z^{-t} s^*(z)\right) . \tag{4.45}$$

where we used $\mathcal{P}_{-N_{\mathrm{h}},N_{\mathrm{h}}}\left(h_t(z)\right) = h_t(z)$. Other update equations for single-channel identification can be derived from Table E.4.

4.2.3 Batch learning algorithm

With a batch learning algorithm we use the estimation error of the whole data sequence for the update of $h(z) = \hat{a}(z)$. Let k denote the iteration index. At

iteration k, the estimation sequence $\hat{x}_k(z)$ is

$$\hat{x}_{t,k} = z^t \, \mathcal{P}_{t,t}\left(h_k(z) s(z)\right) \tag{4.46}$$

$$\hat{x}_k(z) \triangleq \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} \hat{x}_{t,k} z^{-t} = \mathcal{P}_{-T_{\mathrm{x}},T_{\mathrm{x}}}\left(h_k(z) s(z)\right) . \tag{4.47}$$

The linear convolution (4.47) is illustrated in Fig. 4.2 (top). The estimation error sequence $e_{\mathrm{x}k}(z)$ is

$$e_{\mathrm{x}t,k} \triangleq x_t - \hat{x}_{t,k} \tag{4.48}$$

$$e_{\mathrm{x}k}(z) \triangleq \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} e_{\mathrm{x}t,k} z^{-t} \tag{4.49}$$

$$= x(z) - \hat{x}_k(z) \tag{4.50}$$

$$= \mathcal{P}_{-T_{\mathrm{x}},T_{\mathrm{x}}}\left(\left[a(z) - h_k(z)\right] s(z) + n(z)\right) . \tag{4.51}$$

At iteration k we wish to minimize $\|e_{\mathrm{x}k}(z)\|_{\mathcal{F}}^2$. This is equal to averaging the update over the whole data sequence of $2T_{\mathrm{x}} + 1$ samples. As an example, if we use LMS1-Hx we have

$$h_{n,k+1} = h_{n,k} + \frac{\mu}{2T_{\mathrm{x}} + 1} \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} e_{\mathrm{x}t,k} \, s_{t-n}^* \tag{4.52}$$

$$h_{k+1}(z) = h_k(z) + \frac{\mu}{2T_{\mathrm{x}} + 1} \sum_{n=-N_{\mathrm{h}}}^{N_{\mathrm{h}}} \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} \mathcal{P}_{t,t}\left(e_{\mathrm{x}k}(z)\right) \, \mathcal{P}_{t-n,t-n}^*\left(s(z)\right) \tag{4.53}$$

$$= h_k(z) + \frac{\mu}{2T_{\mathrm{x}} + 1} \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} \mathcal{P}_{t,t}\left(e_{\mathrm{x}k}(z)\right) \, \mathcal{P}_{t-N_{\mathrm{h}},t+N_{\mathrm{h}}}^*\left(s(z)\right) \tag{4.54}$$

$$= \mathcal{P}_{-N_{\mathrm{h}},N_{\mathrm{h}}}\left(h_k(z) + \frac{\mu}{2T_{\mathrm{x}} + 1} e_{\mathrm{x}k}(z) s^*(z)\right) . \tag{4.55}$$

where we used (D.54), $\mathcal{P}_{-T_{\mathrm{x}},T_{\mathrm{x}}}\left(e_{\mathrm{x}k}(z)\right) = e_{\mathrm{x}k}(z)$, and $\mathcal{P}_{-N_{\mathrm{h}},N_{\mathrm{h}}}\left(h_k(z)\right) = h_k(z)$ in the last step. We require that $T_{\mathrm{s}} \geq T_{\mathrm{x}} + N_{\mathrm{h}}$. Other update equations for a batch learning algorithm can be derived with Table E.4, analogously to LMS1-Hx.

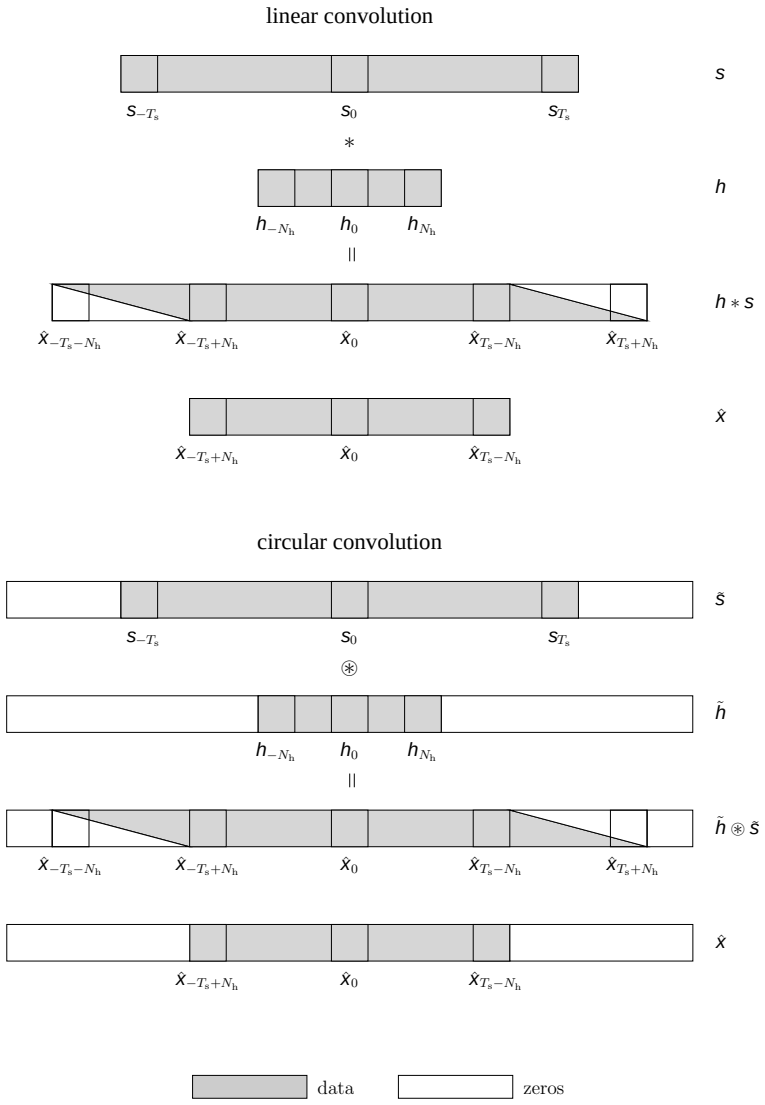

Figure 4.2: Batch learning algorithm ($T_\mathrm{x} = T_\mathrm{s} - N_\mathrm{h}$, $C \geq 2\,(T_\mathrm{s} + N_\mathrm{h}) + 1$): (top) linear convolution $\hat{x}(z) = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}\,(h(z)\,s(z))$, (bottom) circular convolution $\hat{x}(z) = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}\left(\tilde{\mathcal{P}}_C\,(h(z)\,s(z))\right)$.

4.2.4 Block-wise learning algorithm

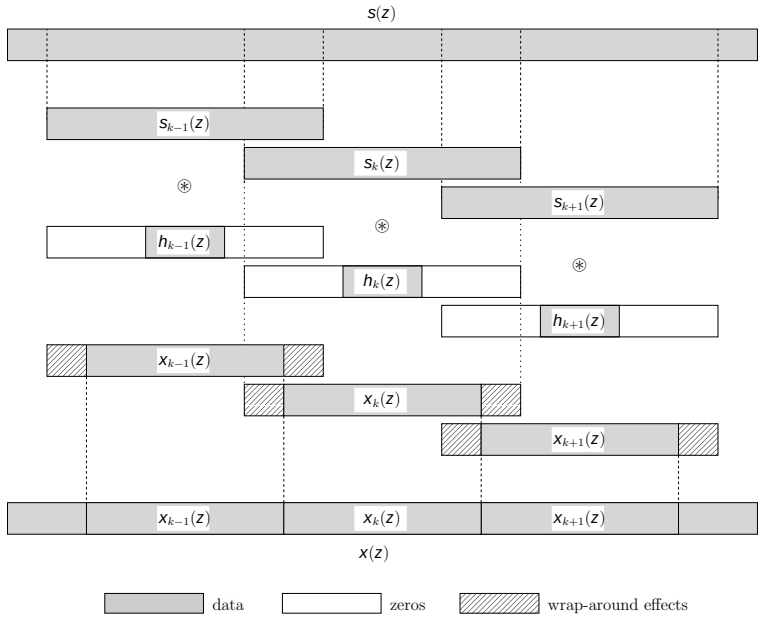

Figure 4.3: Overlap-save technique. The output sequence is partitioned into non-overlapping blocks.

We change the system model slightly such that the input, the noise, and the output sequence have infinite length. The system output is still given by $x_t = \mathcal{P}_{t,t}\,(a(z)s(z) + n(z))$. We now apply the overlap-save technique [84], where the output sequence is partitioned into consecutive, non-overlapping blocks of length $L = 2T_\mathrm{x} + 1$, as shown in Fig. 4.3. With k we denote the block index. We can then describe the output sequence as $x(z) = \sum_{k=-\infty}^{\infty} x_k(z) z^{-kL}$ with

$$x_k(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} x_{kL+t} z^{-t} = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}},\,(a(z)s_k(z) + n_k(z)) \tag{4.56}$$

where $s_k(z)$ and $n_k(z)$ are defined in (4.23) and (4.24). According to the overlap-save technique [84], we require that $T_\mathrm{s} \geq T_\mathrm{x} + N_\mathrm{h}$. Therefore two consecutive input blocks $s_{k-1}(z)$ and $s_k(z)$ overlap. We can partition the esti-

mation sequence as $\hat{x}(z) = \sum_{k=-\infty}^{\infty} \hat{x}_k(z) z^{-kL}$ where at block k we have

$$\hat{x}_{t,k} = z^t \, \mathcal{P}_{t,t}\left(h_k(z) s_k(z)\right) \tag{4.57}$$

$$\hat{x}_k(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \hat{x}_{t+kL,k} \, z^{-t} = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x},}\left(h_k(z) s_k(z)\right) . \tag{4.58}$$

Accordingly, the estimation error sequence can be written as $e_\mathrm{x}(z) = \sum_{k=-\infty}^{\infty} e_{\mathrm{x}k}(z) z^{-kL}$ with

$$e_{\mathrm{x}k}(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} e_{\mathrm{x}kL+t,k} z^{-t} \tag{4.59}$$

$$= x_k(z) - \hat{x}_k(z) \tag{4.60}$$

$$= \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x},}\left([a(z) - h_k(z)] s_k(z) + n_k(z)\right) . \tag{4.61}$$

The update is proceeded at the block rate, where in block k we aim at minimizing $\|e_{\mathrm{x}k}(z)\|_{\mathcal{F}}^2$. This is equal to averaging the update over a whole block of $L = 2T_\mathrm{x} + 1$ samples. As an example, if we use LMS1-Hx we have

$$h_{n,k+1} = h_{n,k} + \frac{\mu}{2T_\mathrm{x}+1} \sum_{t=kL-T_\mathrm{x}}^{kL+T_\mathrm{x}} e_{\mathrm{x}t} \, s_{t-n}^* \tag{4.62}$$

$$h_{k+1}(z) = h_k(z) + \frac{\mu}{2T_\mathrm{x}+1} \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \mathcal{P}_{t,t}\left(e_{\mathrm{x}k}(z)\right) \mathcal{P}_{t-N_\mathrm{h},t+N_\mathrm{h}}^*\left(s_k(z)\right) \tag{4.63}$$

$$= \mathcal{P}_{-N_\mathrm{h},N_\mathrm{h}}\left(h_k(z) + \frac{\mu}{2T_\mathrm{x}+1} e_{\mathrm{x}k}(z) s_k^*(z)\right) . \tag{4.64}$$

Note the similarity between (4.64) and (4.55). Other block-wise learning algorithms can be derived analogously to LMS1-Hx with the sample-wise update equations of Table E.4.

4.2.5 Some remarks

The block-wise learning algorithm is related to the batch learning algorithm, but instead of using the same block of $2T_\mathrm{x} + 1$ samples for every iteration, a new input block with $2T_\mathrm{x} + 1$ samples is chosen for every iteration. Furthermore, by setting T_x to zero ($L = 1$) the block-wise learning algorithm degenerates to an online learning algorithm.

With $N_\mathrm{a} = 1$ and $N_\mathrm{h} = 1$ the model and the algorithms described in this section reduce to those in Section 4.1. We first focused on this most simple case, because it already shows some fundamental concepts and introduces the mathematical formulation in a simple way.

4.2.6 Identification of a causal system

In a first step, we treat a causal system as a special case of a non-causal system. By making only minor changes in the system description and in the update equation, we obtain almost the same algorithms as in previous sections. The unknown causal system $a(z)$ and the corresponding estimation $h(z)$ are described by

$$a(z) \triangleq \sum_{n=0}^{N_\mathrm{a}} a_n z^{-n} \tag{4.65}$$

$$h(z) \triangleq \sum_{n=0}^{N_\mathrm{h}} h_n z^{-n} . \tag{4.66}$$

The system output $x(z)$, the estimation $\hat{x}(z)$, and the error signal $e_\mathrm{x}(z)$ are derived analogously to the non-causal system. Minor changes are made in the update equations, namely in the parameters of the projection operator, and we use again the LMS1-Hx for demonstration. The update equations of the online learning algorithm given in Section 4.2.2 become

$$h_{t+1}(z) = h_t(z) + \mu \mathcal{P}_{t,t}\left(e_\mathrm{x}(z)\right) \mathcal{P}_{t-N_\mathrm{h},t}^*\left(s(z)\right) \tag{4.67}$$

$$= \mathcal{P}_{0,N_\mathrm{h}}\left(h_t(z) + \mu e_{\mathrm{x}t} z^{-t} s^*(z)\right) \tag{4.68}$$

those of the batch algorithm from Section 4.2.3 change to

$$h_{k+1}(z) = h_k(z) + \frac{\mu}{2T_\mathrm{x}+1} \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \mathcal{P}_{t,t}\left(e_{\mathrm{x}k}(z)\right) \mathcal{P}_{t-N_\mathrm{h},t}^*\left(s(z)\right) \tag{4.69}$$

$$= \mathcal{P}_{0,N_\mathrm{h}}\left(h_k(z) + \frac{\mu}{2T_\mathrm{x}+1} e_{\mathrm{x}k}(z) s^*(z)\right) \tag{4.70}$$

and those of the block-wise learning algorithm from Section 4.2.4 become

$$h_{k+1}(z) = h_k(z) + \frac{\mu}{2T_\mathrm{x}+1} \sum_{t=kL-T_\mathrm{x}}^{kL+T_\mathrm{x}} \mathcal{P}_{t,t}\left(z^{-kL} e_{\mathrm{x}k}(z)\right) \mathcal{P}^*_{t-N_\mathrm{h},t}\left(s_k(z)\right) \tag{4.71}$$

$$= \mathcal{P}_{0,N_\mathrm{h}}\left(h_k(z) + \frac{\mu}{2T_\mathrm{x}+1} e_{\mathrm{x}k}(z) s_k^*(z)\right) . \tag{4.72}$$

In these steps, we have essentially omitted the computation of the update of the non-causal filter part.

4.2.7 Extension to circular convolution

We now wish to consider the case where long filters are involved. Since the filter operation is a linear convolution, we are interested in an efficient implementation of the linear convolution. From Section 3.2.3 we know that there exists a fast implementation of the circular convolution by using the FFT. We also know that every linear convolution can be carried out by a circular convolution of appropriate size. Therefore we modify the filter and update equations such that we can apply a circular convolution, and find the necessary conditions.

Batch learning algorithm

Non-causal system We consider the learning algorithm of Section 4.2.3, where a non-causal filter is estimated. At iteration k, the estimation sequence $\hat{x}_k(z)$ is given in (4.47). We carry out the following modification

$$\hat{x}_k(z) = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}\left(h_k(z)s(z)\right) \tag{4.73}$$

$$= \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}\left(\tilde{\mathcal{P}}_C\left(h_k(z)s(z)\right)\right) \tag{4.74}$$

With the help of the circular projection operator $\tilde{\mathcal{P}}\left(.\right)$, defined in Section D.2, we have actually included the computation of a circular convolution. Equality holds if

$$C \geq 2T_\mathrm{s} + 1 \geq 2(T_\mathrm{x} + N_\mathrm{h}) + 1 \tag{4.75}$$

where the inequality on the right guarantees that every element of $\hat{x}_k(z)$ is free from boundary effects, e.g., $\hat{x}_t$ is a sum of $2N_\mathrm{h}+1$ elements, and the inequality on the left guarantees that at least the $2T_\mathrm{x}+1$ center elements of the circular convolution $\tilde{\mathcal{P}}_C\left(h_k(z)s(z)\right)$ coincide with those of the linear convolution $h_k(z)s(z)$. The circular convolution (4.74) is illustrated in Fig. 4.2 (bottom).

We can proceed the same way for the update equations. Again we use LMS1-Hx as an example. Starting With (4.55), we carry out the following modification

$$h_{k+1}(z) = h_k(z) + \frac{\mu}{2T_\mathrm{x}+1} \mathcal{P}_{-N_\mathrm{h},N_\mathrm{h}}\left(e_{\mathrm{x}k}(z)s^*(z)\right) \tag{4.76}$$

$$= h_k(z) + \frac{\mu}{2T_\mathrm{x}+1} \mathcal{P}_{-N_\mathrm{h},N_\mathrm{h}}\left(\tilde{\mathcal{P}}_C\left(e_{\mathrm{x}k}(z)s^*(z)\right)\right) \tag{4.77}$$

which holds for (4.75), just as for the filtering in (4.74).

Causal system In case of a causal system, (4.75) changes to

$$C \geq 2T_\mathrm{s} + 1 \geq T_\mathrm{s} + T_\mathrm{x} + N_\mathrm{h} + 1 \tag{4.78}$$

which means that for the same FFT size C, $L = T_\mathrm{s} + T_\mathrm{x} + 1$ output elements coincide with the linear convolution, as opposed to the non-causal case where we have $L = 2T_\mathrm{x} + 1$.

Some algorithms have more than one convolution in their update equation. If the same technique is applied, where a circular convolution is incorporated into the update equations, the corresponding FFT size C actually depends on the length of all sequences which are involved in the convolution. However, if T_x is at least as large as N_h, the error of the wrap-around effect of the circular convolution harms the estimation $h(z)$ only slightly, especially when $s(z)$ is a white signal.

Block-wise learning algorithm

As already mentioned in Section 4.2.5, the difference between a batch and block-wise learning algorithm is basically whether the update iterations are done with the same, or with new input data. Therefore we can do the same modifications for a block-wise learning algorithm, as for the batch algorithm, described in the previous section. By simply substituting $s_k(z)$ in place of $s(z)$

in (4.74) and (4.77), we obtain the filtering and update equations of LMS1-Hx, where k now denotes the block index. The same constraints (4.75) and (4.78) for C hold.

4.3 Efficient implementation of single-channel system identification by transformation into the frequency domain

In the following, we are interested in computationally efficient implementations of the concepts described in Section 4.2.7. To this end, we transform the filtering and the update of the filter coefficients into the frequency domain.

4.3.1 Online learning algorithm

Usually, for an online learning algorithm, with adaptation being carried out at the sample rate, the filtering and the adaptation remain in the time domain. However, it might still be worth carrying out the adaptation in the frequency domain (or more generally, the *transform domain*) if the autocorrelation matrix of the input signal s has a large eigenvalue spread, which slows down the convergence rate. The DFT tends to decorrelate the input signal for a large DFT length C and therefore a bin-wise step-size normalization can be applied to speed up the convergence [77].

4.3.2 Batch learning algorithm

The whole batch learning algorithm for system identification is given on page 104 from (4.80) to (4.95). Since we have all data available, we can adapt a non-causal filter.

The following comments can be made:

- In (4.80), the constraint on the minimum size of the number of input samples $2T_\mathrm{s}+1$ is given for the case where we have $2T_\mathrm{x}+1$ output samples and where we wish to adapt $2N_\mathrm{a}+1$ filter coefficients. If the number of input samples is limited, we can also read (4.80) as $T_\mathrm{x} \leq T_\mathrm{s}-N_\mathrm{h}$. We then see that only $2(T_\mathrm{s}-N_\mathrm{h})+1$ output samples should be used to build the estimation error. The constraint on the minimum FFT size is $C \geq 2T_\mathrm{s}+1$ and guarantees that the output samples in $\tilde{\hat{\mathbf{x}}}_k$ are free of wrap-around effects stemming from the circular convolution. The FFT size C can be even and is usually chosen to be a power of two.

- The initial settings $(k=0)$ for the algorithm are given from (4.82) to (4.87). Note that the elements with index zero are arranged to be the first elements of the time-domain vectors. From there, the causal part (or samples from the future) is located in the first elements of the vector, whereas the non-causal part (or samples from the past) is located at the end of the vectors. This makes it easy to transform the filter and update equations, derived in the z-domain, into the domain of circulant matrices, with the isomorphic mapping $\hat{x}(z)=\tilde{\mathcal{P}}_C\left(h(z)s(z)\right) \cong \tilde{\hat{\mathbf{X}}}=\tilde{\mathbf{H}}\tilde{\mathbf{S}} \cong \bar{\hat{\mathbf{X}}}=\bar{\mathbf{H}}\bar{\mathbf{S}}$.

- In (4.88), we define the projection matrix

$$\mathbf{P}_{\bar{\mathbf{h}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_\mathrm{h},N_\mathrm{h}}\,\mathbf{F}^{-1} = \mathbf{F}\begin{bmatrix} \mathbf{I}_{N_\mathrm{h}+1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{N_\mathrm{h}} \end{bmatrix}^{C\times C}\mathbf{F}^{-1} \tag{4.79}$$

  which is used in (4.95) to reset the center elements of $\tilde{\mathbf{h}}_{k+1}$ to zero after every update. This is the same technique which is widely known in frequency-domain adaptive filtering [36, 96]. $\tilde{\mathbf{P}}_{-N_\mathrm{h},N_\mathrm{h}}$ is defined according to (3.12). Sometimes this *filter projection operation* is omitted to reduce the computational complexity of the algorithm. The price for this is that the output signal is disturbed by wrap-around effects of the circular convolution. However, simulations have shown that the filter projection operation (4.95) helps for a faster and more smooth convergence of the algorithm. A good compromise is to carry out projection operations once every few blocks.

- In (4.89), the filtering (convolution) $\tilde{\mathcal{P}}_C\left(h_k(z)s(z)\right) \cong \tilde{\mathbf{H}}_k\,\tilde{\mathbf{S}}$, with $\tilde{\mathbf{H}}_k = \mathcal{C}(\tilde{\mathbf{h}}_k)$ and $\tilde{\mathbf{S}}=\mathcal{C}(\tilde{\mathbf{s}})$, is carried out in the frequency domain. In (4.90) the $2T_\mathrm{x}+1$ elements which belong to the linear convolution $\hat{x}_k(z) = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}\left(\tilde{\mathcal{P}}_C\left(h_k(z)s(z)\right)\right)$ are extracted, as seen in (4.91).

Batch learning algorithm for SISO system identification

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_\mathrm{s} + 1 \geq 2\,(T_\mathrm{x} + N_\mathrm{h}) + 1 \tag{4.80}$$
$$L = 2\,T_\mathrm{x} + 1 \tag{4.81}$$
$$\tilde{\mathbf{x}} = (x_0, \ldots, x_{T_\mathrm{x}}, 0, \ldots, 0, x_{-T_\mathrm{x}}, \ldots, x_{-1})^T \tag{4.82}$$
$$\bar{\mathbf{X}} = \mathrm{diag}\,[\mathbf{F}\,\tilde{\mathbf{x}}] \tag{4.83}$$
$$\tilde{\mathbf{s}} = (s_0, \ldots, s_{T_\mathrm{s}}, 0, \ldots, 0, s_{-T_\mathrm{s}}, \ldots, s_{-1})^T \tag{4.84}$$
$$\bar{\mathbf{S}} = \mathrm{diag}\,[\mathbf{F}\,\tilde{\mathbf{s}}] \tag{4.85}$$
$$\tilde{\mathbf{h}}_0 = (h_0, \ldots, h_{N_\mathrm{h}}, 0, \ldots, 0, h_{-N_\mathrm{h}}, \ldots, h_{-1})^T \tag{4.86}$$
$$\bar{\mathbf{H}}_0 = \mathrm{diag}\left[\mathbf{F}\,\tilde{\mathbf{h}}_0\right] \tag{4.87}$$
$$\mathbf{P}_{\bar{\mathbf{h}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_\mathrm{h}, N_\mathrm{h}}\,\mathbf{F}^{-1} \tag{4.88}$$

For every iteration $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\bar{\tilde{\mathbf{X}}}_k = \bar{\mathbf{H}}_k\,\bar{\mathbf{S}} \tag{4.89}$$
$$\tilde{\hat{\mathbf{x}}}_k = \tilde{\mathbf{P}}_{-T_\mathrm{x}, T_\mathrm{x}}\,\mathbf{F}^{-1}\,\mathrm{diag}\left(\bar{\tilde{\mathbf{X}}}_k\right) \tag{4.90}$$
$$= (\hat{x}_0, \ldots, \hat{x}_{T_\mathrm{x}}, 0, \ldots, 0, \hat{x}_{-T_\mathrm{x}}, \ldots, \hat{x}_{-1})^T \tag{4.91}$$

2. Adaptation error:

$$\tilde{\mathbf{e}}_{\mathrm{x}_k} = \tilde{\mathbf{x}} - \tilde{\hat{\mathbf{x}}}_k \tag{4.92}$$
$$\bar{\mathbf{E}}_{\mathrm{x}_k} = \mathrm{diag}\left[\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{x}_k}\right] \tag{4.93}$$

3. Update equations:

$$\bar{\mathbf{H}}'_{k+1} = \text{any update equation from Table E.5} \tag{4.94}$$
$$\bar{\mathbf{H}}_{k+1} = \mathrm{diag}\left[\mathbf{P}_{\bar{\mathbf{h}}}\;\mathrm{diag}\left(\bar{\mathbf{H}}'_{k+1}\right)\right] \tag{4.95}$$

Block-wise learning algorithm for SISO system identification

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_\mathrm{s} + 1 \geq T_\mathrm{s} + T_\mathrm{x} + N_\mathrm{h} + 1 \tag{4.96}$$
$$L = T_\mathrm{s} + T_\mathrm{x} + 1 \tag{4.97}$$
$$\tilde{\mathbf{h}}_0 = (h_0, \ldots, h_{N_\mathrm{h}}, 0, \ldots, 0)^T \tag{4.98}$$
$$\bar{\mathbf{H}}_0 = \mathrm{diag}\left[\mathbf{F}\,\tilde{\mathbf{h}}_0\right] \tag{4.99}$$
$$\mathbf{P}_{\bar{\mathbf{h}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{0, N_\mathrm{h}}\,\mathbf{F}^{-1} \tag{4.100}$$

For every block $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\tilde{\mathbf{x}}_k = (x_{kL}, \ldots, x_{kL+T_\mathrm{s}}, 0, \ldots, 0, x_{kL-T_\mathrm{x}}, \ldots, x_{kL-1})^T \tag{4.101}$$
$$\bar{\mathbf{X}}_k = \mathrm{diag}\,[\mathbf{F}\,\tilde{\mathbf{x}}_k] \tag{4.102}$$
$$\tilde{\mathbf{s}}_k = (s_{kL}, \ldots, s_{kL+T_\mathrm{s}}, 0, \ldots, 0, s_{kL-T_\mathrm{s}}, \ldots, s_{kL-1})^T \tag{4.103}$$
$$\bar{\mathbf{S}}_k = \mathrm{diag}\,[\mathbf{F}\,\tilde{\mathbf{s}}_k] \tag{4.104}$$
$$\bar{\tilde{\mathbf{X}}}_k = \bar{\mathbf{H}}_k\,\bar{\mathbf{S}}_k \tag{4.105}$$
$$\tilde{\hat{\mathbf{x}}}_k = \tilde{\mathbf{P}}_{-T_\mathrm{x}, T_\mathrm{s}}\,\mathbf{F}^{-1}\,\mathrm{diag}\left(\bar{\tilde{\mathbf{X}}}_k\right) \tag{4.106}$$
$$= (\hat{x}_{kL}, \ldots, \hat{x}_{kL+T_\mathrm{s}}, 0, \ldots, 0, \hat{x}_{kL-T_\mathrm{x}}, \ldots, \hat{x}_{kL-1})^T \tag{4.107}$$

2. Adaptation error:

$$\tilde{\mathbf{e}}_{\mathrm{x}_k} = \tilde{\mathbf{x}}_k - \tilde{\hat{\mathbf{x}}}_k \tag{4.108}$$
$$\bar{\mathbf{E}}_{\mathrm{x}_k} = \mathrm{diag}\left[\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{x}_k}\right] \tag{4.109}$$

3. Update equations:

$$\bar{\mathbf{H}}'_{k+1} = \text{any update equation from Table E.5} \tag{4.110}$$
$$\bar{\mathbf{H}}_{k+1} = \mathrm{diag}\left[\mathbf{P}_{\bar{\mathbf{h}}}\;\mathrm{diag}\left(\bar{\mathbf{H}}'_{k+1}\right)\right] \tag{4.111}$$

- In (4.92), we build the adaptation error in the time domain and $\tilde{\mathbf{e}}_{\mathrm{x}_k} \cong e_{\mathrm{x}k}(z)$ consists of $L = 2T_{\mathrm{x}} + 1$ non-zero elements. In (4.93) the error vector $\tilde{\mathbf{e}}_{\mathrm{x}_k}$ is transformed into the frequency domain, where it is used for the update.

- Any update equation listed in Table E.5 can be used for the adaptation. The RLS-type algorithms additionally require the update of a correlation matrix, which is also carried out in the frequency domain. As mentioned above, (4.95) constrains the center elements of $\tilde{\mathbf{h}}_k$ to be zero after every adaptation.

4.3.3 Block-wise learning algorithm

The whole block-wise learning algorithm for system identification is given on page 105 from (4.96) to (4.111). Conceptually, the block-wise and batch learning algorithm are almost equal, except that we now adapt a causal filter $h(z)$.

The following comments can be made:

- In contrast to the batch algorithm where we adapted a non-causal filter of length $2\,N_{\mathrm{h}} + 1$, we now adapt a causal filter of length $N_{\mathrm{h}} + 1$. If we use the same FFT length C as for the batch algorithm, we can choose a larger block length L, as given in (4.97). We also see from (4.108), (4.101), (4.106), and (4.107), that $L = T_{\mathrm{s}} + T_{\mathrm{x}} + 1$ output samples are now used to build the block estimation error.

- In (4.100), we define the projection matrix

$$\mathbf{P}_{\tilde{\mathbf{h}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{0,N_{\mathrm{h}}}\,\mathbf{F}^{-1} = \mathbf{F}\begin{bmatrix} \mathbf{I}_{N_{\mathrm{h}}+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}^{C\times C}\mathbf{F}^{-1} \tag{4.112}$$

  which is used in (4.111) to zero pad $\tilde{\mathbf{h}}_{k+1}$ after every update. Note the difference between (4.100) and (4.88).

- If we use the LMS1-Hx for the update and choose $C = 2T_{\mathrm{s}}+1$, the block-wise learning algorithm is equivalent to the one proposed by Ferrara in [36, 37], except for a permuted arrangement of the elements.

- Any update equation listed in Table E.5 can be used for the adaptation. The RLS-type algorithms additionally require the update of a correlation matrix.

4.4 Single-channel inverse modeling

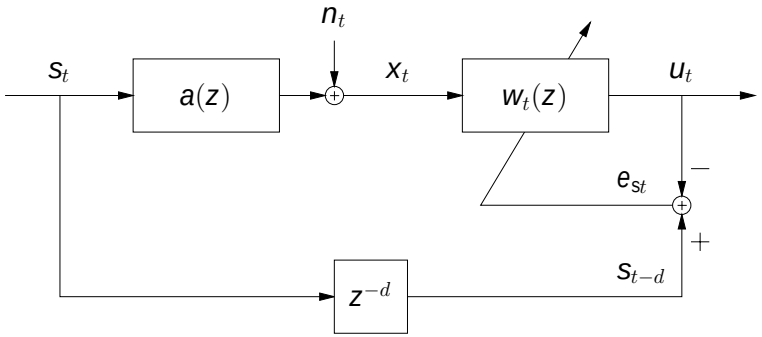

Figure 4.4: Causal realization of single-channel equalization. The causal filter $w(z)$ is adapted such that $g_t(z) = w_t(z)a(z) \approx z^{-d}$. The delay of d samples allows the inclusion of the non-causal part of the expansion of $a^{-1}(z)$.

The setup of inverse modeling, inverse-system identification, or system equalization is shown in Fig. 4.4. The model is the same as described in Section 4.2.1, where the unknown filter is allowed to be non-causal. The global system response is

$$g_t(z) = w_t(z)a(z)\,. \tag{4.113}$$

The error criterion which we aim at minimizing is

$$E\left\{|e_{\mathrm{s}t}|^2\right\} = \|1 - g_t(z)\|^2_{\mathcal{F}}\,\sigma^2_{\mathrm{s}} + \|w_t(z)\|^2_{\mathcal{F}}\,\sigma^2_{\mathrm{n}} \tag{4.114}$$

where $s(z)$ and $n(z)$ are assumed to be uncorrelated, i.e. $\langle s(z), n(z)\rangle_{\mathcal{F}} = 0$, and $\sigma^2_{\mathrm{s}} \triangleq E\left\{|s_t|^2\right\}$ and $\sigma^2_{\mathrm{n}} \triangleq E\left\{|n_t|^2\right\}$. In the noise-free case, this is equal to minimizing $\|1 - g_t(z)\|_{\mathcal{F}}$, which gives the *zero-forcing* solution $w(z) = a^{-1}(z)$.

Just as in Section 4.2, we now describe an online, a batch, and a block-wise learning algorithm, however, now for single-channel inverse modeling instead

of simply for a gain a. The derivations are almost the same, therefore we skip, now and again, a few steps.

4.4.1 Online learning algorithm

With an online learning algorithm we process the estimation and adaptation at the sample rate. Let $w(z) = \hat{a}^{-1}(z)$ denote the estimation of $a^{-1}(z)$

$$w(z) \triangleq \sum_{n=-N_\mathrm{w}}^{N_\mathrm{w}} w_n z^{-n}. \tag{4.115}$$

We then have

$$u_t = z^t \mathcal{P}_{t,t}\left(w_t(z)x(z)\right) \tag{4.116}$$

where $w_t(z)$ is $\hat{a}^{-1}(z)$ at discrete time t, and

$$e_{\mathrm{s}t} \triangleq s_t - u_t \tag{4.117}$$

is the corresponding estimation error. At time instant t, we take $|e_{\mathrm{s}t}|^2$ as the error criterion which we wish to minimize. As an example, the LMS3-Ws becomes

$$\begin{aligned} w_{n,t+1} &= w_{n,t} + \mu e_{\mathrm{s}t} x^*_{t-n} && (4.118)\\ w_{t+1}(z) &= w_t(z) + \mu\, \mathcal{P}_{t,t}\left(e_{\mathrm{s}t}(z)\right)\, \mathcal{P}^*_{t-N_\mathrm{w},t+N_\mathrm{w}}\left(x(z)\right) && (4.119)\\ &= \mathcal{P}_{-N_\mathrm{w},N_\mathrm{w}}\left(w_t(z) + \mu e_{\mathrm{s}t} z^{-t} x^*(z)\right) && (4.120) \end{aligned}$$

where $\mathcal{P}_{t,t}\left(e_{\mathrm{s}t}(z)\right) = e_{\mathrm{s}_t} z^{-t}$ and $e_\mathrm{s}(z) \triangleq \sum_t e_{\mathrm{s}t} z^{-t}$.

Other update equations for single-channel inverse modeling can be derived with the update equations for inverse-gain identification from Table E.8, similarly to LMS3-Ws.

4.4.2 Batch learning algorithm

For the batch learning algorithm we use the estimation error of the whole data sequence for the update of $w_k(z)$, where k denotes the iteration index. At

iteration k, the sequence output $u_k(z)$ is

$$\begin{aligned} u_{t,k} &= z^t \mathcal{P}_{t,t}\left(w_k(z)x(z)\right) && (4.121)\\ &= z^t \mathcal{P}_{t,t}\left(g_k(z)s(z) + w_k(z)n(z)\right) && (4.122)\\ u_k(z) &\triangleq \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} u_{t,k} z^{-t} = \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(w_k(z)x(z)\right) && (4.123)\\ &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(g_k(z)s(z) + w_k(z)n(z)\right) && (4.124) \end{aligned}$$

with $g_k(z) = w_k(z)a(z)$. The estimation error sequence $e_{\mathrm{s}k}(z)$ is

$$\begin{aligned} e_{\mathrm{s}t,k} &\triangleq s_t - u_{t,k} && (4.125)\\ e_{\mathrm{s}k}(z) &\triangleq \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} e_{\mathrm{s}t,k} z^{-t} && (4.126)\\ &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(s(z) - u_k(z)\right) && (4.127)\\ &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(\left[1 - g_k(z)\right] s(z) - w_k(z)n(z)\right). && (4.128) \end{aligned}$$

At iteration k we wish to minimize $\|e_{\mathrm{s}k}(z)\|^2_{\mathcal{F}}$. This is equal to averaging the update over the whole data sequence of $L = 2T_\mathrm{u} + 1$ samples. As an example, if we use LMS3-Ws we have

$$w_{n,k+1} = w_{n,k} + \frac{\mu}{2T_\mathrm{u}+1} \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} e_{\mathrm{s}t,k}\, x^*_{t-n} \tag{4.129}$$

$$w_{k+1}(z) = w_k(z) + \frac{\mu}{2T_\mathrm{u}+1} \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} \mathcal{P}_{t,t}\left(e_{\mathrm{s}k}(z)\right)\, \mathcal{P}^*_{t-N_\mathrm{w},t+N_\mathrm{w}}\left(x(z)\right) \tag{4.130}$$

$$= \mathcal{P}_{-N_\mathrm{w},N_\mathrm{w}}\left(w_k(z) + \frac{\mu}{2T_\mathrm{u}+1} e_{\mathrm{s}k}(z)x^*(z)\right) \tag{4.131}$$

where we used (D.54), $\mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(e_{\mathrm{s}k}(z)\right) = e_{\mathrm{s}k}(z)$, and $\mathcal{P}_{-N_\mathrm{w},N_\mathrm{w}}\left(w_k(z)\right) = w_k(z)$. Other batch learning algorithms can be derived with Table E.8, analogously to LMS3-Ws.

4.4.3 Block-wise learning algorithm

We change the system model slightly such that the input, the noise, and the output sequence have infinite length. Similarly to Section 4.2.4 we apply the

overlap-save technique, where the output sequence $u(z)$ is partitioned into consecutive, non-overlapping blocks of length $L = 2T_\mathrm{u} + 1$. With k we denote the block index. We can then describe the output sequence as $u(z) = \sum_{k=-\infty}^{\infty} u_k(z) z^{-kL}$ with

$$\begin{aligned} u_{t,k} &= z^t \mathcal{P}_{t,t}\left(w_k(z)x(z)\right) && (4.132)\\ &= z^t \mathcal{P}_{t,t}\left(g_k(z)s(z) + w_k(z)n(z)\right) && (4.133)\\ u_k(z) &\triangleq \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} u_{kL+t} z^{-t} && (4.134)\\ &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u},}\left(w_k(z)x_k(z)\right) && (4.135)\\ &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u},}\left(g_k(z)s_k(z) + w_k(z)n_k(z)\right) && (4.136) \end{aligned}$$

where $s_k(z)$ is the system input sequence defined in (4.23), with $T_\mathrm{s} \geq T_\mathrm{x} + N_\mathrm{a}$. According to the overlap-save technique [84], we require that $T_\mathrm{x} \geq T_\mathrm{u} + N_\mathrm{w}$. The estimation error sequence can be written as $e_\mathrm{s}(z) = \sum_{k=-\infty}^{\infty} e_{\mathrm{s}k}(z) z^{-kL}$ with (4.117) and

$$\begin{aligned} e_{\mathrm{s}k}(z) &\triangleq \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} e_{\mathrm{s}kL+t,k} z^{-t} && (4.137)\\ &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u},}\left(s_k(z) - u_k(z)\right) && (4.138)\\ &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u},}\left([1 - g_k(z)]s_k(z) + w_k(z)n_k(z)\right)\,. && (4.139) \end{aligned}$$

The update is processed at the block rate, where in block k we aim at minimizing $\|e_{\mathrm{s}k}(z)\|_{\mathcal{F}}^2$. This is equal to averaging the update over a whole block of $L = 2T_\mathrm{u} + 1$ samples. As an example, if we use LMS3-Ws we have

$$w_{n,k+1} = w_{n,k} + \frac{\mu}{2T_\mathrm{u}+1} \sum_{t=kL-T_\mathrm{u}}^{kL+T_\mathrm{u}} e_{\mathrm{s}t}\, x_{t-n}^* \qquad (4.140)$$

$$w_{k+1}(z) = w_k(z) + \frac{\mu}{2T_\mathrm{u}+1} \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} \mathcal{P}_{t,t}\left(e_{\mathrm{s}k}(z)\right)\, \mathcal{P}^*_{t-N_\mathrm{w},t+N_\mathrm{w}}\left(x_k(z)\right) \qquad (4.141)$$

$$= \mathcal{P}_{-N_\mathrm{w},N_\mathrm{w}}\left(w_k(z) + \frac{\mu}{2T_\mathrm{u}+1} e_{\mathrm{s}k}(z) x_k^*(z)\right) \qquad (4.142)$$

where we used (D.54) in the last step. Further block-wise learning algorithms for single-channel inverse modeling can be derived analogously to LMS3-Ws with the sample-wise update equations of Table E.8.

4.4.4 Extension to circular convolution

We follow the same ideas as in Section 4.2.7, where we consider the case where long filters are involved. Again we modify the filter and update equations such that we can apply a circular convolution, and find the necessary conditions.

Batch learning algorithm

We consider the learning algorithm of Section 4.4.2, where we adapt a non-causal filter to possibly invert a nonminimum-phase system. To this end, we replace the linear convolution, given in (4.123), by a circular convolution, i.e.,

$$\begin{aligned} u_k(z) &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(w_k(z)x(z)\right) && (4.143)\\ &= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(\tilde{\mathcal{P}}_C\left(w_k(z)x(z)\right)\right)\,. && (4.144) \end{aligned}$$

Equality holds for

$$C \geq 2T_\mathrm{x} + 1 \geq 2(T_\mathrm{u} + N_\mathrm{w}) + 1\,. \qquad (4.145)$$

The inequality on the right guarantees that every element of $u_k(z)$ is free from boundary effects, e.g., u_t is a sum of $2N_\mathrm{w} + 1$ elements, and the inequality on the left guarantees that at least the $2T_\mathrm{u} + 1$ center elements of the circular convolution $\tilde{\mathcal{P}}_C\left(w_k(z)x(z)\right)$ coincide with those of the linear convolution $w_k(z)x(z)$.

We can proceed the same way for the update equations. Again we use LMS3-Ws as an example. Starting With (4.131), we obtain

$$\begin{aligned} w_{k+1}(z) &= w_k(z) + \frac{\mu}{2T_\mathrm{u}+1}\mathcal{P}_{-N_\mathrm{w},N_\mathrm{w}}\left(e_{\mathrm{s}k}(z)x^*(z)\right) && (4.146)\\ &= w_k(z) + \frac{\mu}{2T_\mathrm{u}+1}\mathcal{P}_{-N_\mathrm{w},N_\mathrm{w}}\left(\tilde{\mathcal{P}}_C\left(e_{\mathrm{s}k}(z)x^*(z)\right)\right) && (4.147) \end{aligned}$$

which holds for (4.145), just as for the filtering in (4.144).

Block-wise learning algorithm

Substituting $x_k(z)$ for $x(z)$ in (4.144) and (4.147) gives the filtering and update equations of LMS3-Ws. The same constraint (4.145) for C hold. k now denotes the block index.

4.5 Efficient implementation of single-channel inverse modeling

In the following, we are interested in computationally efficient implementations of the concepts described in Section 4.4.4.

4.5.1 Online learning algorithm

See comments in Section 4.3.1.

4.5.2 Batch learning algorithm

The whole batch learning algorithm for inverse modeling is given on page 114 from (4.148) to (4.163). The algorithm is designed such that it can adapt a non-causal filter.

The following comments can be made:

- In (4.148), the constraint on the minimum size of the number of input samples $2T_{\mathrm{x}}+1$ is given for the case where we have $2T_{\mathrm{u}}+1$ output samples and where we wish to adapt $2N_{\mathrm{w}}+1$ filter coefficients. The FFT size C is usually chosen to be a power of two.
- In (4.157), the filtering (convolution) $\tilde{\mathcal{P}}_C\left(w_k(z)x(z)\right) \cong \tilde{\mathbf{W}}_k\tilde{\mathbf{X}}$, with $\tilde{\mathbf{W}}_k = \mathcal{C}(\tilde{\mathbf{w}}_k)$ and $\tilde{\mathbf{X}} = \mathcal{C}(\tilde{\mathbf{x}})$, is carried out in the frequency domain. In (4.158) the $2T_{\mathrm{u}}+1$ elements which belong to the linear convolution $u_k(z) = \mathcal{P}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\left(\tilde{\mathcal{P}}_C\left(w_k(z)x(z)\right)\right)$ are extracted, as seen in (4.159).
- In (4.160), the adaptation error is build in the time domain and then transformed in (4.161) into the frequency domain where it is used afterwards for the update.
- Any update equation listed in Table E.9 can be used for the adaptation. The RLS-type algorithms additionally require the update of a correlation matrix.

For further information, see also the comments in Section 4.3.2 for the batch learning algorithm for system identification.

4.5.3 Block-wise learning algorithm

The whole block-wise learning algorithm for inverse modeling is given on page 115 from (4.164) to (4.179). As opposed to the the block-wise learning algorithms for system identification, described in Section 4.5.3, we adapt a non-causal filter, to cope also with a nonminimum-phase system $a(z)$. In fact, since we introduce a delay in the reference signal, shown in Fig. 4.4, a part of the non-causal part of $w_k(z)$ becomes causal.

The following comments can be made:

- In (4.169), we introduce a delay of d samples for the reference signal $s(z)$. With $-N_{\mathrm{w}} \leq d \leq N_{\mathrm{w}}$, we can steer the center of gravity of $w_k(z)$ within the $2N_{\mathrm{w}}+1$ filter taps. Special cases are: $d=-N_{\mathrm{w}}$ if $a(z)$ is minimum phase, $d=+N_{\mathrm{w}}$ if $a(z)$ is maximum phase, and $d=0$ if $a(z)$ is mixed phase. The parameter d can also be adjusted during the adaptation, depending on the shape of the envelope of $w_k(z)$.
- Any update equation listed in Table E.9 can be used for the adaptation. The RLS-type algorithms additionally require the update of a correlation matrix.

Further comments can be found in Section 4.3.3 for the batch learning algorithm for system identification.

Batch learning algorithm for SISO inverse modeling

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_\mathrm{x} + 1 \geq 2\,(T_\mathrm{u} + N_\mathrm{w}) + 1 \tag{4.148}$$
$$L = 2\,T_\mathrm{u} + 1 \tag{4.149}$$
$$\tilde{\mathbf{s}} = (s_0, \ldots, s_{T_\mathrm{u}}, 0, \ldots, 0, s_{-T_\mathrm{u}}, \ldots, s_{-1})^T \tag{4.150}$$
$$\bar{\mathbf{S}} = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{s}}) \tag{4.151}$$
$$\tilde{\mathbf{x}} = (x_0, \ldots, x_{T_\mathrm{x}}, 0, \ldots, 0, x_{-T_\mathrm{x}}, \ldots, x_{-1})^T \tag{4.152}$$
$$\bar{\mathbf{X}} = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}) \tag{4.153}$$
$$\tilde{\mathbf{w}}_0 = (w_0, \ldots, w_{N_\mathrm{w}}, 0, \ldots, 0, w_{-N_\mathrm{w}}, \ldots, w_{-1})^T \tag{4.154}$$
$$\bar{\mathbf{W}}_0 = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{w}}_0) \tag{4.155}$$
$$\mathbf{P}_{\tilde{\mathbf{w}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_\mathrm{w}, N_\mathrm{w}}\,\mathbf{F}^{-1} \tag{4.156}$$

For every iteration $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\bar{\mathbf{U}}_k = \bar{\mathbf{W}}_k\,\bar{\mathbf{X}} \tag{4.157}$$
$$\tilde{\mathbf{u}}_k = \tilde{\mathbf{P}}_{-T_\mathrm{u}, T_\mathrm{u}}\,\mathbf{F}^{-1}\,\mathrm{diag}\,(\bar{\mathbf{U}}_k) \tag{4.158}$$
$$= (u_0, \ldots, u_{T_\mathrm{u}}, 0, \ldots, 0, u_{-T_\mathrm{u}}, \ldots, u_{-1})^T \tag{4.159}$$

2. Adaptation error:

$$\tilde{\mathbf{e}}_{\mathrm{s}_k} = \tilde{\mathbf{s}} - \tilde{\mathbf{u}}_k \tag{4.160}$$
$$\bar{\mathbf{E}}_{\mathrm{s}_k} = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{s}_k}) \tag{4.161}$$

3. Update equations:

$$\bar{\mathbf{W}}'_{k+1} = \text{any update equation from Table E.9} \tag{4.162}$$
$$\bar{\mathbf{W}}_{k+1} = \mathrm{diag}\,(\mathbf{P}_{\tilde{\mathbf{w}}}\ \mathrm{diag}\,(\bar{\mathbf{W}}'_{k+1})) \tag{4.163}$$

Block-wise learning algorithm for SISO inverse modeling

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_\mathrm{x} + 1 \geq 2\,(T_\mathrm{u} + N_\mathrm{w}) + 1 \tag{4.164}$$
$$L = 2\,T_\mathrm{u} + 1 \tag{4.165}$$
$$\tilde{\mathbf{w}}_0 = (w_0, \ldots, w_{N_\mathrm{w}}, 0, \ldots, 0, w_{-N_\mathrm{w}}, \ldots, w_{-1})^T \tag{4.166}$$
$$\bar{\mathbf{W}}_0 = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{w}}_0) \tag{4.167}$$
$$\mathbf{P}_{\tilde{\mathbf{w}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_\mathrm{w}, N_\mathrm{w}}\,\mathbf{F}^{-1} \tag{4.168}$$

For every block $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\tilde{\mathbf{s}}_k = (s_{kL-d}, \ldots, s_{kL+T_\mathrm{u}-d}, 0, \ldots, 0, s_{kL-T_\mathrm{u}-d}, \ldots, s_{kL-1-d})^T \tag{4.169}$$
$$\bar{\mathbf{S}}_k = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{s}}_k) \tag{4.170}$$
$$\tilde{\mathbf{x}}_k = (x_{kL}, \ldots, x_{kL+T_\mathrm{x}}, 0, \ldots, 0, x_{kL-T_\mathrm{x}}, \ldots, x_{kL-1})^T \tag{4.171}$$
$$\bar{\mathbf{X}}_k = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}_k) \tag{4.172}$$
$$\bar{\mathbf{U}}_k = \bar{\mathbf{W}}_k\,\bar{\mathbf{X}}_k \tag{4.173}$$
$$\tilde{\mathbf{u}}_k = \tilde{\mathbf{P}}_{-T_\mathrm{u}, T_\mathrm{u}}\,\mathbf{F}^{-1}\,\mathrm{diag}\,(\bar{\mathbf{U}}_k) \tag{4.174}$$
$$= (u_{kL}, \ldots, u_{kL+T_\mathrm{u}}, 0, \ldots, 0, u_{kL-T_\mathrm{u}}, \ldots, u_{kL-1})^T \tag{4.175}$$

2. Adaptation error:

$$\tilde{\mathbf{e}}_{\mathrm{s}_k} = \tilde{\mathbf{s}}_k - \tilde{\mathbf{u}}_k \tag{4.176}$$
$$\bar{\mathbf{E}}_{\mathrm{s}_k} = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{s}_k}) \tag{4.177}$$

3. Update equations:

$$\bar{\mathbf{W}}'_{k+1} = \text{any update equation from Table E.9} \tag{4.178}$$
$$\bar{\mathbf{W}}_{k+1} = \mathrm{diag}\,(\mathbf{P}_{\tilde{\mathbf{w}}}\ \mathrm{diag}\,(\bar{\mathbf{W}}'_{k+1})) \tag{4.179}$$

4.6 Wiener filter

In analogy to the instantaneous mixing problem stated in Chapter 2, we can now formulate the *Wiener-Hopf equations* (WHE) for the single-channel convolutive case. To this end we define the following correlation sequences

$$r_{xs}(z) \triangleq \sum_{\tau=-\infty}^{\infty} r_{xs_\tau} z^{-\tau} \tag{4.180}$$

where we have, in the deterministic case,

$$r_{xs_\tau} \triangleq \lim_{T\to\infty} \frac{1}{2T+1} \sum_{t=-T}^{T} x_t s_{t-\tau}^* \tag{4.181}$$

and in the stochastic case

$$r_{xs_\tau} \triangleq E\{x_{t+\tau} s_t^*\} = E\left\{x_t s_{t-\tau}^*\right\} . \tag{4.182}$$

Likewise we define $r_{ss}(z)$, $r_{sx}(z)$, $r_{xx}(z)$, $r_{\hat{x}s}(z)$, $r_{ux}(z)$, $r_{e_{\mathrm{s}}s}(z)$, and $r_{e_{\mathrm{s}}x}(z)$.

4.6.1 Wiener filter $h^{\text{MMSE-x}}(z)$

Infinite-length Wiener filter The Wiener-Hopf equation for the single-channel identification problem can be derived by the *orthogonality principle*, which says that the error signal must be uncorrelated with the input signal

$$r_{e_{\mathrm{x}}s}(z) = 0 . \tag{4.183}$$

For an infinite-length filter $h(z) = \sum_{n=-\infty}^{\infty} h_n z^{-n}$ and the error signal $e_{\mathrm{x}t} = x_t - \hat{x}_t$ we have

$$\begin{aligned} r_{e_{\mathrm{x}}s}(z) &= r_{xs}(z) - r_{\hat{x}s}(z) && (4.184)\\ &= r_{xs}(z) - h(z) r_{ss}(z) . && (4.185) \end{aligned}$$

Using (4.185) in (4.183), we obtain the Wiener-Hopf equation

$$h(z) r_{ss}(z) = r_{xs}(z) . \tag{4.186}$$

Solving (4.186) for $h(z)$ yields

$$h^{\text{MMSE-x}}(z) = r_{xs}(z) r_{ss}^{-1}(z) \tag{4.187}$$

where $h^{\text{MMSE-x}}(z)$ is the Wiener filter which minimizes $E\left\{|e_{\mathrm{x}t}|^2\right\}$. Note the analogy between (4.187) and (2.8) defined in Section 2.2.1.

Finite-length Wiener filter The Wiener-Hopf equation for a finite-length filter $h(z) = \sum_{n=a}^{b} h_n z^{-n}$ is given by

$$\mathcal{P}_{a,b}\left(\mathcal{P}_{a,b}\left(h(z)\right) r_{ss}(z)\right) = \mathcal{P}_{a,b}\left(r_{xs}(z)\right) . \tag{4.188}$$

If we assume that the input signal is white, i.e. $r_{ss}(z) = r_{ss_0}$, then (4.188) becomes

$$h(z) r_{ss_0} = \mathcal{P}_{a,b}\left(r_{xs}(z)\right) . \tag{4.189}$$

Solving (4.189) for $h(z)$ gives

$$h^{\text{MMSE-x}}(z) = \frac{1}{r_{ss_0}} \mathcal{P}_{a,b}\left(r_{xs}(z)\right) . \tag{4.190}$$

4.6.2 Wiener filter $w^{\text{MMSE-s}}(z)$

Infinite-length Wiener filter The Wiener-Hopf equation for the single-channel inverse-modeling problem can also be derived by the *orthogonality principle* [57]. Again, the error signal must be uncorrelated to the input signal which gives now

$$r_{e_{\mathrm{s}}x}(z) = 0 . \tag{4.191}$$

For an infinite-length filter $w(z) = \sum_{n=-\infty}^{\infty} w_n z^{-n}$ and the error signal $e_{\mathrm{s}t} = s_t - u_t$ we have

$$\begin{aligned} r_{e_{\mathrm{s}}x}(z) &= r_{sx}(z) - r_{ux}(z) && (4.192)\\ &= r_{sx}(z) - w(z) r_{xx}(z) . && (4.193) \end{aligned}$$

Using (4.193) in (4.191), we obtain the Wiener-Hopf equation

$$w(z) r_{xx}(z) = r_{sx}(z) . \tag{4.194}$$

Solving (4.194) for $w(z)$ yields

$$w^{\text{MMSE-s}}(z) = r_{sx}(z) r_{xx}^{-1}(z) \tag{4.195}$$

where $w^{\text{MMSE-s}}(z)$ is the Wiener filter which minimizes $E\left\{|e_{\mathrm{s}t}|^2\right\}$. Note the analogy between (4.195) and (2.61) defined in Section 2.5.1.

Finite-length Wiener filter The Wiener-Hopf equation for a finite-length filter $w(z) = \sum_{n=a}^{b} w_n z^{-n}$ is

$$\mathcal{P}_{a,b}\left(\mathcal{P}_{a,b}\left(w(z)\right) r_{xx}(z)\right) = \mathcal{P}_{a,b}\left(r_{sx}(z)\right) \tag{4.196}$$

which can be written with (D.14) as

$$\sum_{\tau=a}^{b} \mathcal{P}_{\tau,\tau}\left(\mathcal{P}_{a,b}\left(w(z)\right) r_{xx}(z)\right) = \sum_{\tau=a}^{b} \mathcal{P}_{\tau,\tau}\left(r_{sx}(z)\right). \tag{4.197}$$

Unfortunately (4.196) is not as easily solvable as (4.188), since $x(z)$ is a non-white sequence and therefore $r_{xx}(z)$ does not consist of a single term. Thus, we have to setup a system of $b-a+1$ linear equations to obtain the filter coefficients w_n. These equations are obtained by evaluating (4.197) for every power z^{τ} for $a \leq \tau \leq b$. From the left side of (4.197) we rewrite

$$\mathcal{P}_{a,b}\left(w(z)\right) r_{xx}(z) = \left(\sum_{n=a}^{b} w_n z^{-n}\right)\left(\sum_{k=-\infty}^{\infty} r_{xx_k} z^{-k}\right) \tag{4.198}$$

$$= \sum_{n=a}^{b}\sum_{k=-\infty}^{\infty} w_n\, r_{xx_k} z^{-(n+k)} \tag{4.199}$$

$$= \sum_{m=-\infty}^{\infty} z^{-m} \sum_{n=a}^{b} w_n\, r_{xx_{m-n}} \tag{4.200}$$

where we used the substitution $m = k+n$. Inserting (4.200) into the left side of (4.197) yields

$$\sum_{\tau=a}^{b} \mathcal{P}_{\tau,\tau}\left(\mathcal{P}_{a,b}\left(w(z)\right) r_{xx}(z)\right) = \sum_{\tau=a}^{b} \mathcal{P}_{\tau,\tau}\left(\sum_{m=-\infty}^{\infty} z^{-m} \sum_{n=a}^{b} w_n\, r_{xx_{m-n}}\right) \tag{4.201}$$

$$= \sum_{\tau=a}^{b} z^{-\tau} \sum_{n=a}^{b} w_n\, r_{xx_{\tau-n}}. \tag{4.202}$$

The right side of (4.197) is

$$\sum_{\tau=a}^{b} \mathcal{P}_{\tau,\tau}\left(r_{sx}(z)\right) = \sum_{\tau=a}^{b} r_{sx_\tau} z^{-\tau}. \tag{4.203}$$

Evaluating now (4.197) for every power $z^{-\tau}$, $(a \leq \tau \leq b)$, and using (4.202) and (4.203) gives the following set of equations

$$\sum_{n=a}^{b} w_n\, r_{xx_{\tau-n}} = r_{sx_\tau} \qquad (a \leq \tau \leq b). \tag{4.204}$$

The equations in (4.204) can be written in matrix form

$$\begin{bmatrix} r_{xx_0} & \cdots & r_{xx_{a-b}} \\ \vdots & \ddots & \vdots \\ r_{xx_{b-a}} & \cdots & r_{xx_0} \end{bmatrix} \cdot \begin{bmatrix} w_a \\ \vdots \\ w_b \end{bmatrix} = \begin{bmatrix} r_{sx_a} \\ \vdots \\ r_{sx_b} \end{bmatrix}. \tag{4.205}$$

Solving this system of linear equations, which involves a matrix inversion of dimension $(b-a+1)\times(b-a+1)$, yields the coefficients w_n of $w^{\text{MMSE-s}}(z)$. Special cases are with $a=0$ and $b=N_{\text{w}}$ (causal Wiener filter), or with $a=-N_{\text{w}}$ and $b=N_{\text{w}}$.

Approximation of $w^{\text{MMSE-s}}(z)$

We now derive an approximation of the finite-length Wiener filter $w^{\text{MMSE-s}}(z)$ of (4.196). We thereby use the existence of a fast implementation of the circular convolution, and a fast implementation of the inverse of a circular matrix.

We use the following approximations: $\tilde{r}_{xx}(z) \triangleq \tilde{\mathcal{P}}_C\left(r_{xx}(z)\right) \approx r_{xx}(z)$ or $\tilde{r}_{xx}(z) \triangleq \mathcal{P}_C\left(r_{xx}(z)\right) \approx r_{xx}(z)$, and $\tilde{r}_{sx}(z) \triangleq \tilde{\mathcal{P}}_C\left(r_{sx}(z)\right) \approx r_{sx}(z)$ or $\tilde{r}_{sx}(z) \triangleq \mathcal{P}_C\left(r_{sx}(z)\right) \approx r_{sx}(z)$, where C is chosen *large enough*, such that $\|r_{xx}(z) - \tilde{r}_{xx}(z)\|_{\mathcal{F}}$ and $\|r_{sx}(z) - \tilde{r}_{sx}(z)\|_{\mathcal{F}}$ become very small. We can now state a new equation similar to (4.196)

$$\tilde{w}(z)\,\tilde{r}_{xx}(z) = \tilde{r}_{sx}(z) \tag{4.206}$$

$$\tilde{\mathcal{P}}_C\left(\tilde{w}(z)\,\tilde{r}_{xx}(z)\,\tilde{r}_{xx}^{-1}(z)\right) = \tilde{\mathcal{P}}_C\left(\tilde{r}_{sx}(z)\,\tilde{r}_{xx}^{-1}(z)\right) \tag{4.207}$$

The left hand side of (4.207) gives exactly $\tilde{w}(z)$ and the inversion in right hand side of (4.207) can be computed by the inversion of a circular matrix, which requires two FFT operations and an element-wise inversion of a C-dimensional vector (Eq. (3.123)). Finally we have

$$w^{\text{MMSE-s}}(z) \approx \mathcal{P}_{a,b}\left(\tilde{\mathcal{P}}_C\left(\tilde{r}_{sx}(z)\tilde{r}_{xx}^{-1}(z)\right)\right). \tag{4.208}$$

4.7 Performance measures

We use the following performance measures for single-channel identification and inverse modeling:

- Average MSE of e_{x} of block k:

$$J_{\text{MSE-x}}(k) = \frac{1}{2T_{\mathrm{x}}+1} \sum_{t=kL-T_{\mathrm{x}}}^{kL+T_{\mathrm{x}}} |e_{\mathrm{x}t}|^2 = \frac{1}{2T_{\mathrm{x}}+1} \left\|e_{\mathrm{x}k}(z)\right\|_{\mathcal{F}}^2 . \quad (4.209)$$

- Average MSE of e_{s} of block k:

$$J_{\text{MSE-s}}(k) = \frac{1}{2T_{\mathrm{u}}+1} \sum_{t=kL-T_{\mathrm{u}}}^{kL+T_{\mathrm{u}}} |e_{\mathrm{s}t}|^2 = \frac{1}{2T_{\mathrm{u}}+1} \left\|e_{\mathrm{s}k}(z)\right\|_{\mathcal{F}}^2 . \quad (4.210)$$

- Average block intersymbol interference $J_{\text{ISI}}(g_k(z))$ of block k, where $J_{\text{ISI}}(.)$ is defined in (6.131) and $g_k(z) = w_k(z)a(z)$.

4.8 Simulations

4.8.1 Hearing-instrument feedback-path

In this example, the filter $a(z)$ is a real measured hearing-instrument feedback-path from [115]. The impulse responses of $a(z)$ and $a^{-1}(z)$, as well as the corresponding transfer functions are shown in Fig. 4.5. We normalized $a(z)$ such that $\|a(z)\|_{\mathcal{F}}^2 = 1$. The sample frequency is $f_{\mathrm{s}} = 1/T = 16$ kHz, where T denotes the sampling period.

4.8.2 System identification

For the system-identification simulation we take the block-wise learning algorithm from page 105, except that we adapt a non-causal filter with $2\,N_{\mathrm{h}}+1$ filter coefficients. We have $T_{\mathrm{s}} = 1000$, $N_{\mathrm{h}} = 200$, $T_{\mathrm{x}} = T_{\mathrm{s}} - N_{\mathrm{h}} = 800$, the block size $L = 2T_{\mathrm{x}}+1 = 1601$, and the FFT size $C = 2048$. The input signal

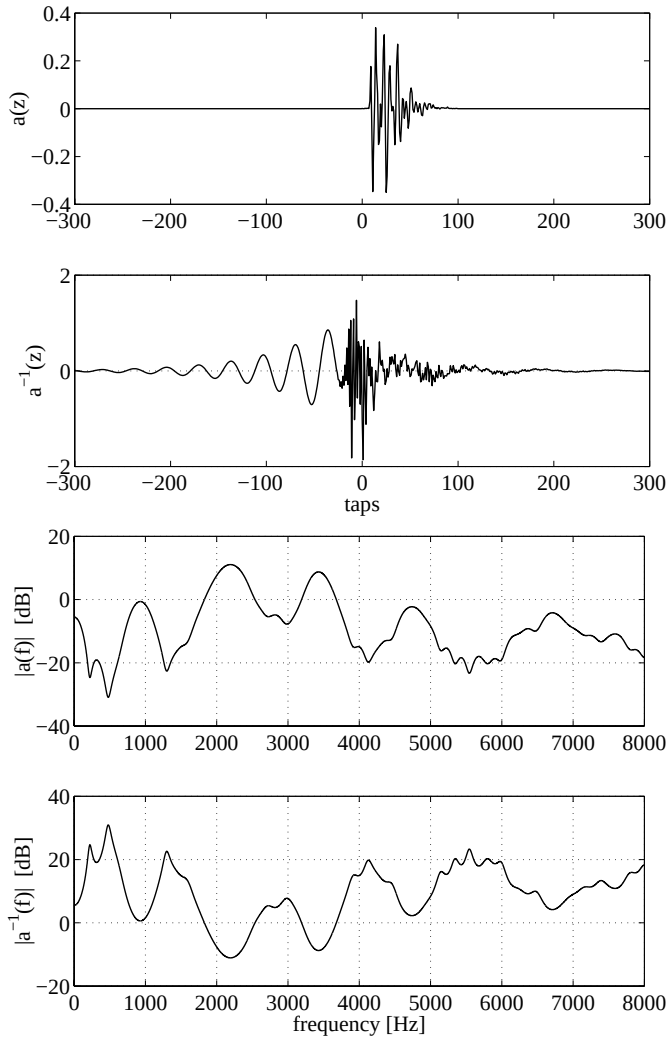

Figure 4.5: Hearing-instrument feedback-path $a(z)$: Top: Impulse response of $a(z)$ and $a^{-1}(z)$. Bottom: Magnitude of transfer function $|a(e^{j2\pi fT})|$ and $|a^{-1}(e^{j2\pi fT})|$. From the two-sided expansion of $a^{-1}(z)$ we see that $a(z)$ must be non-minimum phase.

and the sensor noise are white Gaussian signals with $\sigma_{\mathrm{s}} = 1$ and $\sigma_{\mathrm{n}} = 0.01$ (-40 dB), respectively. The parameters of the algorithms are

LMS1-Hx	$\mu = 0.15$
LMS2-Hx	$\mu = 0.1$
LMS3-Hs	$\mu = 0.1$
RLS1-Hx	$\lambda = 0.2$
RLS2-Hs	$\lambda = 0.2.$

The performance curves of $J_{\text{MSE-x}}$ and $J_{\text{MSE-s}}$ are show in Fig. 4.6, those for J_{ISI} in Fig. 4.7. $J_{\text{MSE-x}}$ is evaluated directly with $h_k(z)$, $J_{\text{MSE-s}}$ and J_{ISI} are evaluated with $\mathcal{P}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\left(h_k^{-1}(z)\right)$ and $g_k(z) = \mathcal{P}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\left(h_k^{-1}(z)\right) a(z)$, respectively, with $N_{\mathrm{w}} = 300$.

We can make the following observations:

- The RLS-based algorithms converge faster than the LMS-based algorithms and achieve about -40 dB for $J_{\text{MSE-x}}$.
- Among the LMS-based algorithms, the LMS1-Hx has the fastest convergence, and also the lowest values for $J_{\text{MSE-x}}$, $J_{\text{MSE-s}}$, and J_{ISI} in the steady state.
- The LMS3-Hs has a very slow convergence. This is caused by the high eigenvalue spread of the input autocorrelation matrix $\mathbf{R_{xx}} \cong r_{xx}(z) = a(z)\, r_{ss}(z)\, a^*(z) = |\, a(z)|^2$, see also $|a(f)|$ in Fig. 4.5.
- We see that $J_{\text{MSE-x}}$ approaches -40 dB, which is what we expect from (4.38), as -40 dB is just the SNR of the sensor signal x. However, $J_{\text{MSE-s}}$ does not go below about -20 dB. This observation was already made in Section 2.10 for the identification of an ill-conditioned mixing matrix.
- At first sight, the forgetting factor $\lambda = 0.2$ seems to be very small. However, since we have a large block length L, there are enough samples within a block for the estimation of the correlation matrices.

4.8.3 Inverse modeling

For the inverse-modeling-identification simulation we take the block-wise learning algorithm from page 115. We have $T_{\mathrm{x}} = 1000$, $N_{\mathrm{w}} = 300$, $T_{\mathrm{u}} = T_{\mathrm{x}} - N_{\mathrm{w}} = 700$, the block size $L = 2T_{\mathrm{u}} + 1 = 1401$, and the FFT size $C = 2048$. The input signal and the sensor noise are white Gaussian signals with $\sigma_{\mathrm{s}} = 1$ and $\sigma_{\mathrm{n}} = 0.01$ (-40 dB SNR), respectively. The parameters of the algorithms are

LMS1-Wx	$\mu = 0.15$
LMS2-Wx	$\mu = 0.1$
LMS3-Ws	$\mu = 0.1$
RLS1-Wx	$\lambda = 0.2$
RLS2-Ws	$\lambda = 0.2$

The performance curves of $J_{\text{MSE-x}}$ and $J_{\text{MSE-s}}$ are show in Fig. 4.8, those for J_{ISI} in Fig. 4.9. $J_{\text{MSE-s}}$ and J_{ISI} are evaluated directly with $w_k(z)$ and $g_k(z) = w_k(z)\, a(z)$, respectively, $J_{\text{MSE-x}}$ is evaluated with $\mathcal{P}_{-N_{\mathrm{h}},N_{\mathrm{h}}}\left(w_k^{-1}(z)\right)$ with $N_{\mathrm{h}} = 200$.

We can make the following observations:

- Again, the RLS-based algorithms converge faster than the LMS-based algorithms.
- Among the LMS-based algorithms, the LMS1-Wx has the fastest convergence and also the lowest values for $J_{\text{MSE-x}}$, $J_{\text{MSE-s}}$, and J_{ISI} in the steady state.
- The final steady-state value of $J_{\text{MSE-x}}$ is higher in all simulations than that for the system-identification simulations.
- $J_{\text{MSE-s}}$ reaches only about -20 dB, and not -40 dB for two reasons. Since we have normalized $\|a(z)\|_F = 1$ and $a(z)$ is not an allpass filter, we have $\|a^{-1}(z)\|_F > 1$. Inserting $a^{-1}(z)$ for $w_k(z)$ in (4.114), gives $\|a^{-1}(z)\|_F\, \sigma_{\mathrm{n}}^2 > \sigma_{\mathrm{n}}^2$, which is one reason why $J_{\text{MSE-s}}$ is higher than -40 dB. The second reason is because we use only finitely many coefficients to estimate $a^{-1}(z)$. We can decompose $a^{-1}(z) - w_k(z) = \left[\mathcal{P}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\left(a^{-1}(z)\right) - w_k(z)\right] + \left[a^{-1}(z) - \mathcal{P}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\left(a^{-1}(z)\right)\right]$. The second term does not vanish, because of the finite length of $w_k(z)$, and therefore the term $\|1 - \mathcal{P}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\left(a^{-1}(z)\right) a^{-1}(z)\|_F\, \sigma_{\mathrm{s}}^2$ remains also as residual error in $J_{\text{MSE-s}}$, even for $\mathcal{P}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\left(a^{-1}(z) - w_k(z)\right) = 0$.
- Again, the LMS3-Ws has a very slow convergence.

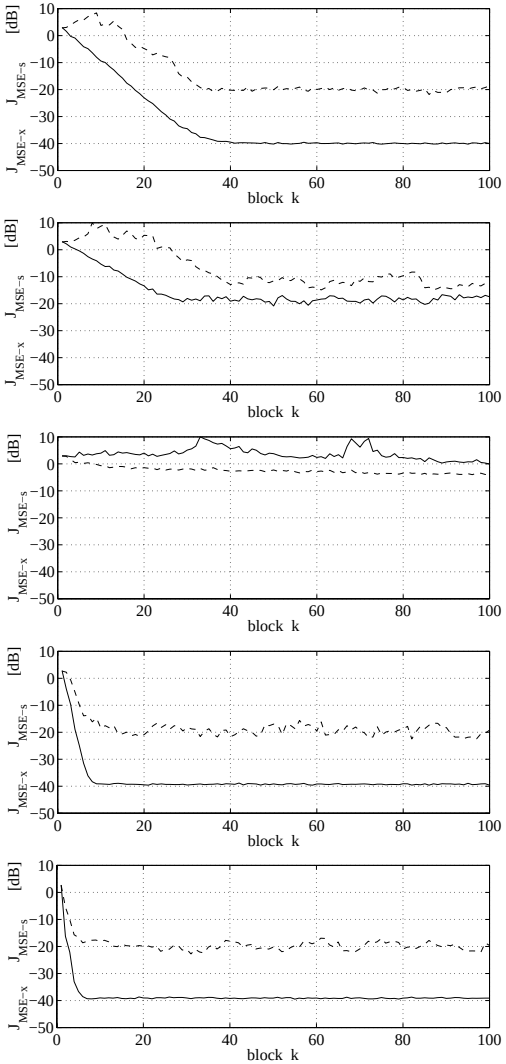

Figure 4.6: Performance curves of system identification: $J_{\text{MSE-x}}(k)$ (solid) and $J_{\text{MSE-s}}(k)$ (dashed). From top: LMS1-Hx, LMS2-Hx, LMS3-Hs, RLS1-Hx, and RLS2-Hs.

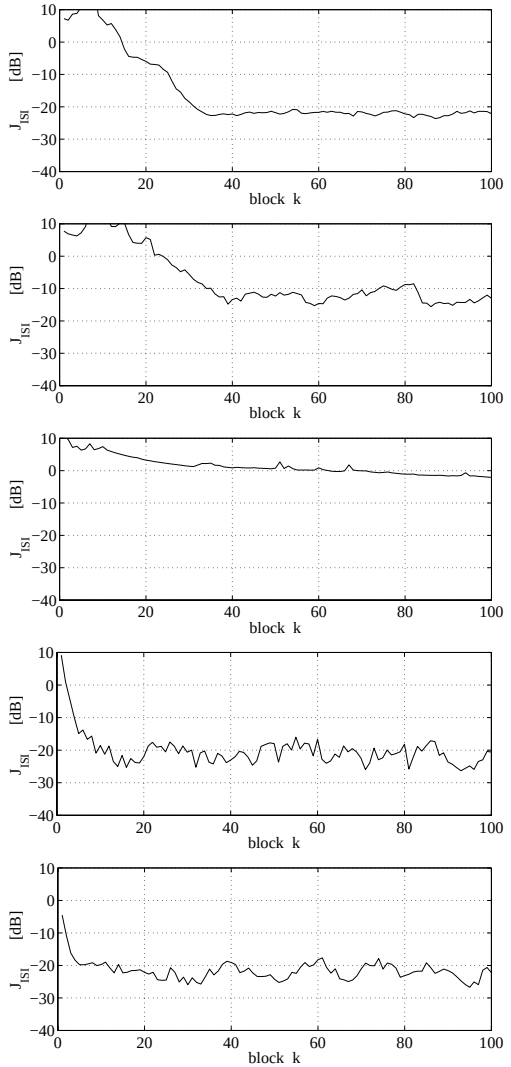

Figure 4.7: Performance curves of system identification: $J_{\text{ISI}}(k)$. From top: LMS1-Hx, LMS2-Hx, LMS3-Hs, RLS1-Hx, and RLS2-Hs.

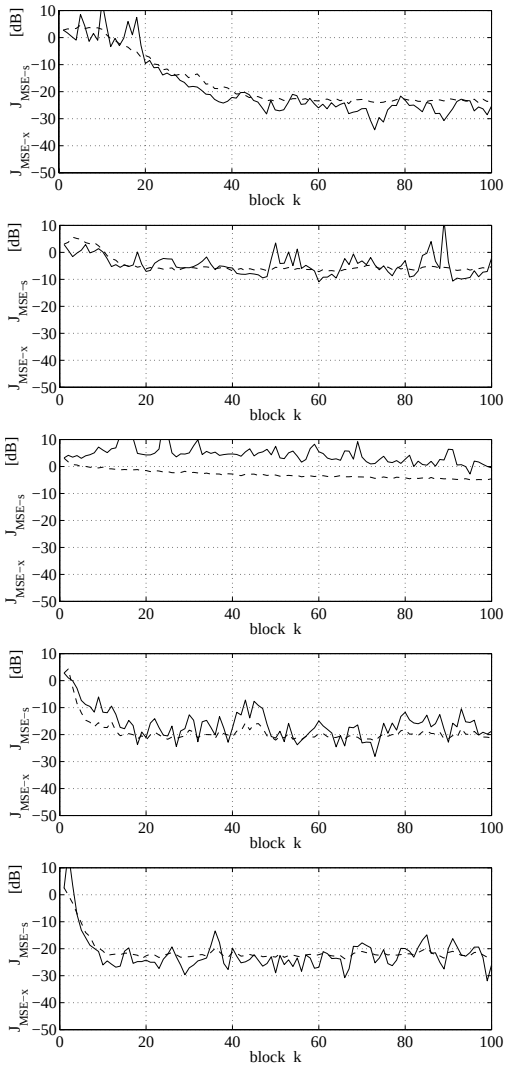

Figure 4.8: Performance curves of inverse modeling: $J_{\text{MSE-x}}(k)$ (solid) and $J_{\text{MSE-s}}(k)$ (dashed). From top: LMS1-Wx, LMS2-Wx, LMS3-Ws, RLS1-Wx, and RLS2-Ws.

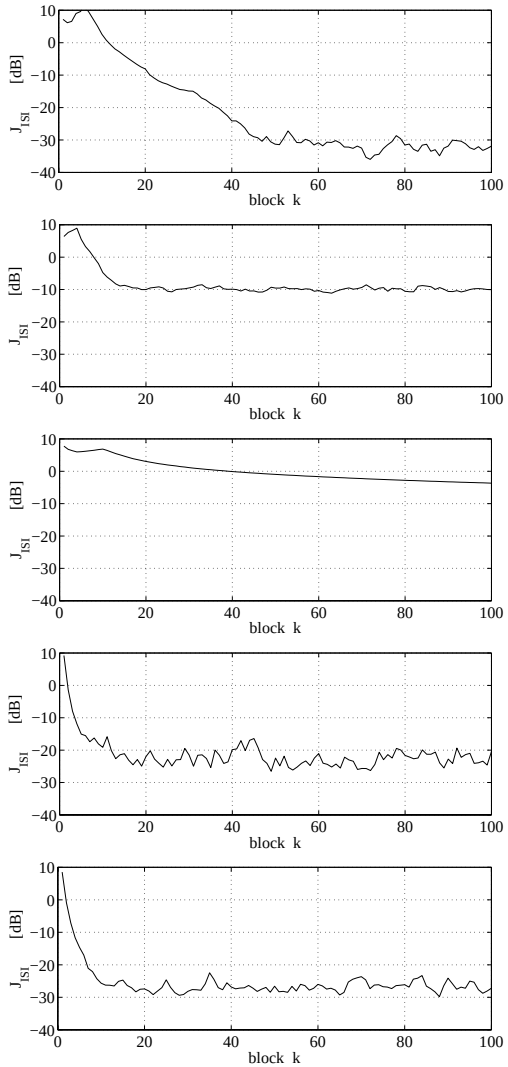

Figure 4.9: Performance curves of inverse modeling: $J_{\text{ISI}}(k)$. From top: LMS1-Wx, LMS2-Wx, LMS3-Ws, RLS1-Wx, and RLS2-Ws.

The performance curves reveal, that with a small $J_{\text{MSE-x}}$ or $J_{\text{MSE-s}}$, the J_{ISI} is also small, and vice versa. Furthermore, the simulations show, that the algorithms reveal a similar behavior as their counterparts in Section 2.

4.9 Summary

In this section we have described the single-channel case of system identification and inverse modeling. We have focused on the situation, where the unknown channel filter is *long*. Hence, from the computational point of view, it is worth carrying out the filtering and the adaptation in the frequency domain. Together with the concepts from Chapter 2, where we have analyzed the multichannel instantaneous-mixing case, we now have the tools necessary for dealing with the general multichannel case, as shown also in the commutative diagram in Fig. 1.2.

Alternatively to the overlap-save technique, where the output sequence is subdivided into consecutive, non-overlapping blocks, we could also use an adaptive algorithm with an overlap-add technique [18, 19, 98], where the input sequence is subdivided into non-overlapping blocks, as shown in Fig. 4.10.

The blind counter part of inverse modeling, i.e. blind deconvolution, will be described in Section 6.8.

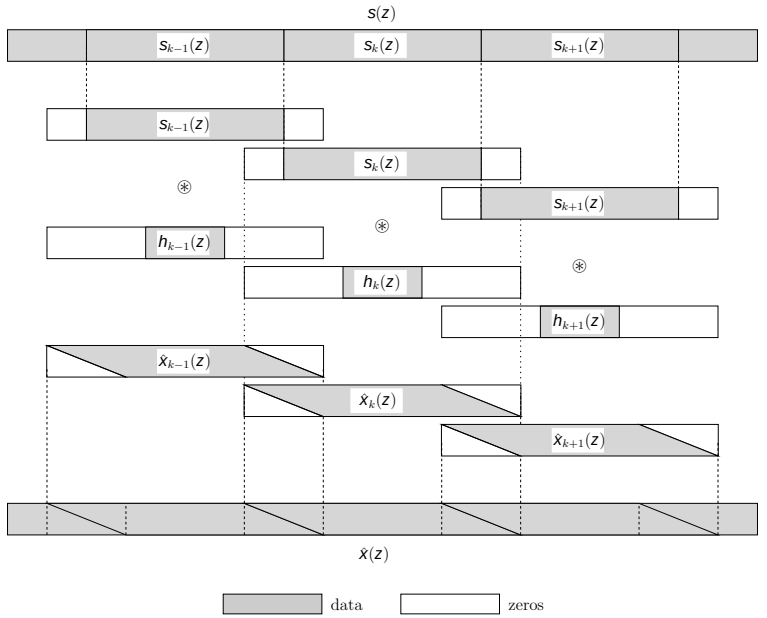

Figure 4.10: Overlap-add technique. The input sequence is partitioned into non-overlapping blocks.

Chapter 5

Multichannel identification and inverse modeling

In this chapter we combine the instantaneous-mixing case from Chapter 2 with the single-channel convolution case from Chapter 4, see also Fig. 1.2.

5.1 Rules for the multichannel extension

As shown in Fig. 1.2, we can extend either the single-channel convolutive-mixing case, or the multichannel instantaneous-mixing case to the multichannel convolutive-mixing case.

Single-channel to multichannel extension $(a(z) \rightarrow \mathbf{A}(z))$ To extend the update equations from Chapter 4 for the single-channel convolutive-mixing case to the multichannel counterpart, we use the following rules:

- A polynomial $a(z)$ is replaced by a polynomial matrix $\mathbf{A}(z)$ or a polynomial vector $\mathbf{a}(z)$.
- Complex conjugation is replaced by Hermitian transposition.

- A circulant matrix $\tilde{\mathbf{A}}$ is replaced by a block circulant matrix $\widetilde{\mathbf{A}}$.
- A diagonal matrix $\bar{\mathbf{A}}$ is replaced by a block diagonal matrix $\overline{\mathbf{A}}$.
- The Fourier matrix $\mathbf{F}$ and the inverse Fourier matrix $\mathbf{F}^{-1}$ are replaced by $\mathbf{T}$ and $\mathbf{T}^{-1}$, respectively, where $\mathbf{T}$ is defined in (3.7).

Extending the instantaneous-mixing case to the convolutive-mixing case ($\mathbf{A} \rightarrow \mathbf{A}(z)$) To extend the update equations from Chapter 2 for the instantaneous mixing case to the convolutive mixing case, we use the following rules:

- Depending on the context, a matrix $\mathbf{A}$ is replaced either by a polynomial matrix $\mathbf{A}(z)$, a block circulant matrix $\widetilde{\mathbf{A}}$, or a block diagonal matrix $\overline{\mathbf{A}}$.
- Depending on the context, a vector $\mathbf{a}$ is replaced either by a polynomial vector $\mathbf{a}(z)$, a block circulant matrix $\widetilde{\mathbf{A}}$, or a block diagonal matrix $\overline{\mathbf{A}}$.
- We apply the polynomial projection operator $\mathcal{P}$ or the circular polynomial projection operator $\tilde{\mathcal{P}}$ after every operation in the z-domain, as we are interested in filters and time sequences of finite length.

Similar rules are given in [34, 69, 71].

With these rules we extend the system description and the adaptive algorithms to the multichannel convolutive-mixing case.

5.2 Description of the multichannel system

Convolutive-mixing matrix We use the same multichannel convolutive-mixing model, as described in Section 1.2, except that the unknown system has now finite length

$$\mathbf{A}(z) \triangleq \sum_{n=-N_\mathrm{a}}^{N_\mathrm{a}} \mathbf{A}_n z^{-n} = [a_{ij}(z)] \tag{5.1}$$

$$a_{ij}(z) \triangleq \sum_{n=-N_\mathrm{a}}^{N_\mathrm{a}} a_{ij,n} z^{-n} \qquad \begin{matrix} i = 1, \ldots, M \\ j = 1, \ldots, M_\mathrm{s}\,. \end{matrix} \tag{5.2}$$

Environment The Multichannel extension of Section 4.2.1 is as follows: The input sequence $\mathbf{s}(z)$ of length $2T_\mathrm{s}+1$ is

$$\mathbf{s}(z) \triangleq \sum_{t=-T_\mathrm{s}}^{T_\mathrm{s}} \mathbf{s}_t z^{-t}. \tag{5.3}$$

The input-output behavior of the model is defined as

$$\mathbf{x}_t = z^t \mathcal{P}_{t,t} \left(\mathbf{A}(z)\, \mathbf{s}(z) + \mathbf{n}(z) \right) \tag{5.4}$$

$$\mathbf{x}(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \mathbf{x}_t z^{-t} = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}} \left(\mathbf{A}(z)\, \mathbf{s}(z) + \mathbf{n}(z) \right) \tag{5.5}$$

where $\mathbf{x}(z)$ is the output sequence of finite length $2T_\mathrm{x}+1$. The elements of $\mathbf{n}(z)$ contain the sensor-noise sequences, and have also length $2T_\mathrm{x}+1$.

5.3 Multichannel system identification

In multichannel system identification, we wish to find a polynomial matrix

$$\mathbf{H}(z) \triangleq \sum_{n=-N_\mathrm{h}}^{N_\mathrm{h}} \mathbf{H}_n z^{-n} = [h_{ij}(z)] \tag{5.6}$$

$$h_{ij}(z) \triangleq \sum_{n=-N_\mathrm{h}}^{N_\mathrm{h}} h_{ij,n} z^{-n} \qquad \begin{matrix} i = 1, \ldots, M \\ j = 1, \ldots, M_\mathrm{s}\,. \end{matrix} \tag{5.7}$$

such that

$$\hat{\mathbf{x}}(z) \triangleq \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}} \left(\mathbf{H}(z)\, \mathbf{s}(z) \right) \tag{5.8}$$

is an estimate of $\mathbf{x}(z)$ with the corresponding estimation error

$$\mathbf{e}_\mathrm{x}(z) \triangleq \mathbf{x}(z) - \hat{\mathbf{x}}(z) \tag{5.9}$$

$$= \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}} \left(\left[\mathbf{A}(z) - \mathbf{H}(z) \right] \mathbf{s}(z) + \mathbf{n}(z) \right). \tag{5.10}$$

5.3.1 Batch learning algorithm for multichannel system identification

This section is the multichannel extension of Section 4.2.3 and 4.2.7.

Description in the z-domain In analogy with (4.47), the multichannel estimation sequence is

$$\hat{\mathbf{x}}_{t,k} = z^t \, \mathcal{P}_{t,t} \left(\mathbf{H}_k(z) \, \mathbf{s}(z) \right) \tag{5.11}$$

$$\hat{\mathbf{x}}_k(z) \triangleq \sum_{t=-T_\mathrm{x}}^{T_\mathrm{x}} \hat{\mathbf{x}}_{t,k} z^{-t} = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}} \left(\mathbf{H}_k(z) \, \mathbf{s}(z) \right) \tag{5.12}$$

where k denotes the iteration index. Introducing a circular convolution similar to (4.74), gives

$$\hat{\mathbf{x}}_k(z) = \mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}} \left(\tilde{\mathcal{P}}_C \left(\mathbf{H}_k(z) \, \mathbf{s}(z) \right) \right) \tag{5.13}$$

which also holds for (4.75), i.e., $C \geq 2T_\mathrm{s} + 1 \geq 2(T_\mathrm{x} + N_\mathrm{h}) + 1$. For the adaptation we choose again the LMS1-Hx as an example. Extending (4.55) gives

$$\mathbf{H}_{n,k+1} = \mathcal{P}_{-N_\mathrm{h},N_\mathrm{h}} \left(\mathbf{H}_k(z) + \frac{\mu}{2T_\mathrm{x}+1} \mathbf{e}_{\mathrm{x}k}(z) \, \mathbf{s}^H(z) \right) \tag{5.14}$$

or

$$\mathbf{H}_{k+1}(z) = \mathbf{H}_k(z) + \frac{\mu}{2T_\mathrm{x}+1} \mathcal{P}_{-N_\mathrm{h},N_\mathrm{h}} \left(\tilde{\mathcal{P}}_C \left(\mathbf{e}_{\mathrm{x}k}(z) \, \mathbf{s}^H(z) \right) \right) \tag{5.15}$$

if we include a circular convolution similar to (4.77). Note, that the complex conjugation was replaced by a Hermitian transposition.

Fast implementation The whole batch learning algorithm for multichannel system identification is given on page 136 from (5.17) to (5.32). Since we have all data available, we adapt a non-causal filter. The algorithm is the multichannel extension of the algorithm described in Section 4.3.2.

The following comments can be made:

- The circular convolution $\tilde{\mathcal{P}}_C \left(\mathbf{H}_k(z) \, \mathbf{s}(z) \right)$, which is a part of (5.13) is calculated in (5.26) in the frequency-domain. Note, that we have the isomorphism $\tilde{\mathcal{P}}_C \left(\mathbf{H}_k(z) \, \mathbf{s}(z) \right) \cong \widetilde{\hat{\mathbf{X}}}_k = \widetilde{\mathbf{H}}_k \, \widetilde{\mathbf{S}} \cong \overline{\hat{\mathbf{X}}}_k = \overline{\mathbf{H}}_k \, \overline{\mathbf{S}}$. The projection operation $\mathcal{P}_{-T_\mathrm{x},T_\mathrm{x}}(.)$ in (5.13) is carried out in (5.27) in the time domain.

- Any update equation listed in Table E.6 can be used for the adaptation. The RLS-type algorithms additionally require the update of a block correlation matrix.
- The projection operation $\mathcal{P}_{-N_\mathrm{h},N_\mathrm{h}}(.)$ in (5.14) or (5.15), which constrains the polynomials in $\mathbf{H}_{k+1}(z)$ to have only $2\,N_\mathrm{h}+1$ terms, is carried out in (5.32). To this end, the M^2 filters are transformed into the time domain, padded with zeros, and transformed back into the frequency domain. The projection matrix $\tilde{\mathbf{P}}_{-N_\mathrm{h},N_\mathrm{h}}$ is defined according to (3.13).

5.3.2 Block-wise learning for multichannel system identification

This section is the multichannel extension of Section 4.2.4 and 4.2.7. The difference to the batch learning algorithm is that k denotes now the block index, and for each adaptation step, a new input block

$$\mathbf{s}_k(z) \triangleq \sum_{t=-T_\mathrm{s}}^{T_\mathrm{s}} \mathbf{s}_{kL+t} \, z^{-t} = \mathcal{P}_{-T_\mathrm{s},T_\mathrm{s},} \left(z^{kL} \, \mathbf{s}(z) \right) \tag{5.16}$$

is used, instead of the whole sequence $\mathbf{s}(z)$. Similarly to the single-channel case, we use an overlap-save technique.

Fast implementation The whole block-wise learning algorithm for multichannel system identification is given on page 137 from (5.33) to (5.48). We adapt a causal filter $\mathbf{H}(z)$. The algorithm is the multichannel extension of the algorithm described in Section 4.3.3.

The following comments can be made:

- Since we adapt a causal filter, the constraints in (5.33) are slightly different to (5.17). Similarly, the block length L in (5.34) is defined differently from that in (5.18).
- Any update equation listed in Table E.6 can be used for the adaptation. The RLS-type algorithms additionally require the update of a block correlation matrix.
- The projection matrix $\tilde{\mathbf{P}}_{0,N_\mathrm{h}}$ is defined according to (3.12).

Batch learning algorithm for MIMO system identification

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_\mathrm{s} + 1 \geq 2\,(T_\mathrm{x} + N_\mathrm{h}) + 1 \tag{5.17}$$

$$L = 2\,T_\mathrm{x} + 1 \tag{5.18}$$

$$\tilde{\mathbf{x}}_m = (x_{m,0}, \ldots, x_{m,T_\mathrm{x}}, 0, \ldots, 0, x_{m,-T_\mathrm{x}}, \ldots, x_{m,-1})^T \tag{5.19}$$

$$\overline{\mathbf{X}} = \left[\bar{\mathbf{X}}_m\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}_m)\right] \tag{5.20}$$

$$\tilde{\mathbf{s}}_m = (s_{m,0}, \ldots, s_{m,T_\mathrm{s}}, 0, \ldots, 0, s_{m,-T_\mathrm{s}}, \ldots, s_{m,-1})^T \tag{5.21}$$

$$\overline{\mathbf{S}} = \left[\bar{\mathbf{S}}_m\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{s}}_m)\right] \tag{5.22}$$

$$\tilde{\mathbf{h}}_{ij,0} = (h_{ij,0}, \ldots, h_{ij,N_\mathrm{h}}, 0, \ldots, 0, h_{ij,-N_\mathrm{h}}, \ldots, h_{ij,-1})^T \tag{5.23}$$

$$\overline{\mathbf{H}}_0 = \left[\bar{\mathbf{H}}_{ij,0}\right] = \left[\mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{h}}_{ij,0}\right)\right] \tag{5.24}$$

$$\mathbf{P}_{\tilde{\mathbf{h}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_\mathrm{h},N_\mathrm{h}}\,\mathbf{F}^{-1} \tag{5.25}$$

For every iteration $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\overline{\hat{\mathbf{X}}}_k = \overline{\mathbf{H}}_k\,\overline{\mathbf{S}} \tag{5.26}$$

$$\tilde{\hat{\mathbf{x}}}_{m,k} = \tilde{\mathbf{P}}_{-T_\mathrm{x},T_\mathrm{x}}\,\mathbf{F}^{-1}\,\mathrm{diag}\left(\bar{\hat{\mathbf{X}}}_{m,k}\right) \tag{5.27}$$

$$= (\hat{x}_{m,0}, \ldots, \hat{x}_{m,T_\mathrm{x}}, 0, \ldots, 0, \hat{x}_{m,-T_\mathrm{x}}, \ldots, \hat{x}_{m,-1})^T \tag{5.28}$$

2. Adaptation error:

$$\tilde{\mathbf{e}}_{\mathrm{x}_{m,k}} = \tilde{\mathbf{x}}_m - \tilde{\hat{\mathbf{x}}}_{m,k} \tag{5.29}$$

$$\overline{\mathbf{E}}_{\mathrm{x}_k} = \left[\bar{\mathbf{E}}_{\mathrm{x}_{m,k}}\right] = \left[\mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{x}_{m,k}}\right)\right] \tag{5.30}$$

3. Update equations:

$$\overline{\mathbf{H}}'_{k+1} = \text{any update equation from Table E.6} \tag{5.31}$$

$$\bar{\mathbf{H}}_{ij,k+1} = \mathrm{diag}\left(\mathbf{P}_{\tilde{\mathbf{h}}}\,\mathrm{diag}\left(\bar{\mathbf{H}}'_{ij,k+1}\right)\right) \tag{5.32}$$

Block-wise learning algorithm for MIMO system identification

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_\mathrm{s} + 1 \geq T_\mathrm{s} + T_\mathrm{x} + N_\mathrm{h} + 1 \tag{5.33}$$

$$L = T_\mathrm{s} + T_\mathrm{x} + 1 \tag{5.34}$$

$$\tilde{\mathbf{h}}_{ij,0} = (h_{ij,0}, \ldots, h_{ij,N_\mathrm{h}}, 0, \ldots, 0)^T \tag{5.35}$$

$$\overline{\mathbf{H}}_0 = \left[\bar{\mathbf{H}}_{ij,0}\right] = \left[\mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{h}}_{ij,0}\right)\right] \tag{5.36}$$

$$\mathbf{P}_{\tilde{\mathbf{h}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{0,N_\mathrm{h}}\,\mathbf{F}^{-1} \tag{5.37}$$

For every block $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\tilde{\mathbf{x}}_{m,k} = (x_{m,kL}, \ldots, x_{m,kL+T_\mathrm{s}}, 0, \ldots, 0, x_{m,kL-T_\mathrm{x}}, \ldots, x_{m,kL-1})^T \tag{5.38}$$

$$\overline{\mathbf{X}}_k = \left[\bar{\mathbf{X}}_{m,k}\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}_{m,k})\right] \tag{5.39}$$

$$\tilde{\mathbf{s}}_{m,k} = (s_{m,kL}, \ldots, s_{m,kL+T_\mathrm{s}}, 0, \ldots, 0, s_{m,kL-T_\mathrm{s}}, \ldots, s_{m,kL-1})^T \tag{5.40}$$

$$\overline{\mathbf{S}}_k = \left[\bar{\mathbf{S}}_{m,k}\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{s}}_{m,k})\right] \tag{5.41}$$

$$\overline{\hat{\mathbf{X}}}_k = \overline{\mathbf{H}}_k\,\overline{\mathbf{S}}_k \tag{5.42}$$

$$\tilde{\hat{\mathbf{x}}}_{m,k} = \tilde{\mathbf{P}}_{-T_\mathrm{x},T_\mathrm{s}}\,\mathbf{F}^{-1}\,\mathrm{diag}\left(\bar{\hat{\mathbf{X}}}_{m,k}\right) \tag{5.43}$$

$$= (\hat{x}_{m,kL}, \ldots, \hat{x}_{m,kL+T_\mathrm{s}}, 0, \ldots, 0, \hat{x}_{m,kL-T_\mathrm{x}}, \ldots, \hat{x}_{m,kL-1})^T \tag{5.44}$$

2. Adaptation error:

$$\tilde{\mathbf{e}}_{\mathrm{x}_{m,k}} = \tilde{\mathbf{x}}_m - \tilde{\hat{\mathbf{x}}}_{m,k} \tag{5.45}$$

$$\overline{\mathbf{E}}_{\mathrm{x}_k} = \left[\bar{\mathbf{E}}_{\mathrm{x}_{m,k}}\right] = \left[\mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{x}_{m,k}}\right)\right] \tag{5.46}$$

3. Update equations:

$$\overline{\mathbf{H}}'_{k+1} = \text{any update equation from Table E.6} \tag{5.47}$$

$$\bar{\mathbf{H}}_{ij,k+1} = \mathrm{diag}\left(\mathbf{P}_{\tilde{\mathbf{h}}}\,\mathrm{diag}\left(\bar{\mathbf{H}}'_{ij,k+1}\right)\right) \tag{5.48}$$

5.4 Multichannel inverse modeling

In multichannel inverse modeling, we wish to find a polynomial matrix

$$\mathbf{W}(z) \triangleq \sum_{n=-N_\mathrm{w}}^{N_\mathrm{w}} \mathbf{W}_n z^{-n} = [w_{ij}(z)] \tag{5.49}$$

$$w_{ij}(z) \triangleq \sum_{n=-N_\mathrm{w}}^{N_\mathrm{w}} w_{ij,n} z^{-n} \qquad i,j = 1,\ldots,M \tag{5.50}$$

such that

$$\mathbf{G}(z) = \mathbf{W}(z)\mathbf{A}(z) \tag{5.51}$$

becomes close to the unity matrix $\mathbf{I}$. Thus,

$$\mathbf{u}(z) \triangleq \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(\mathbf{W}(z)\,\mathbf{x}(z)\right) \tag{5.52}$$

is an estimate of $\mathbf{s}(z)$ with the corresponding extimation error

$$\mathbf{e}_\mathrm{s}(z) \triangleq \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(\mathbf{s}(z) - \mathbf{u}(z)\right) \tag{5.53}$$

$$= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(\left[\mathbf{I} - \mathbf{W}(z)\,\mathbf{A}(z)\right]\mathbf{s}(z) - \mathbf{W}(z)\,\mathbf{n}(z)\right)\,. \tag{5.54}$$

5.4.1 Batch learning algorithm for multichannel inverse modeling

This section is the multichannel extension of Section 4.4.2 and 4.4.4.

Description in the z-domain In analogy to (4.123), the multichannel estimation sequence is

$$\mathbf{u}_{t,k} = z^t\,\mathcal{P}_{t,t}\left(\mathbf{W}_k(z)\,\mathbf{x}(z)\right) \tag{5.55}$$

$$= z^t\,\mathcal{P}_{t,t}\left(\mathbf{G}_k(z)\,\mathbf{s}(z) + \mathbf{W}_k(z)\,\mathbf{n}(z)\right) \tag{5.56}$$

$$\mathbf{u}_k(z) \triangleq \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} \mathbf{u}_{t,k} z^{-t} = \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(\mathbf{W}_k(z)\,\mathbf{x}(z)\right) \tag{5.57}$$

$$= \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(\mathbf{G}_k(z)\,\mathbf{s}(z) + \mathbf{W}_k(z)\,\mathbf{n}(z)\right) \tag{5.58}$$

Introducing a circular convolution similar to (4.144), gives

$$\mathbf{u}_k(z) = \mathcal{P}_{-T_\mathrm{u},T_\mathrm{u}}\left(\tilde{\mathcal{P}}_C\left(\mathbf{W}_k(z)\,\mathbf{x}(z)\right)\right)\,. \tag{5.59}$$

which also holds for (4.145), i.e., $C \geq 2T_\mathrm{x} + 1 \geq 2(T_\mathrm{u} + N_\mathrm{w}) + 1$. For the adaptation we choose again the LMS3-Ws as an example. Extending (4.131) gives

$$\mathbf{W}_{k+1}(z) = \mathcal{P}_{-N_\mathrm{w},N_\mathrm{w}}\left(\mathbf{W}_k(z) + \frac{\mu}{2T_\mathrm{u}+1}\mathbf{e}_{\mathrm{s}k}(z)\,\mathbf{x}^H(z)\right) \tag{5.60}$$

or

$$\mathbf{W}_{k+1}(z) = \mathbf{W}_k(z) + \frac{\mu}{2T_\mathrm{u}+1}\mathcal{P}_{-N_\mathrm{w},N_\mathrm{w}}\left(\tilde{\mathcal{P}}_C\left(\mathbf{e}_{\mathrm{s}k}(z)\,\mathbf{x}^H(z)\right)\right) \tag{5.61}$$

if we include a circular convolution similar to (4.147). Note, that the complex conjugation was replaced by a Hermitian transposition.

Fast implementation The whole batch learning algorithm for multichannel inverse modeling is given on page 140 from (5.62) to (5.77). The algorithm is designed such that it can adapt a non-causal filter. The algorithm is the multichannel extension of the algorithm described in Section 4.5.2.

The following comment can be made:

- Any update equation listed in Table E.10 can be used for the adaptation. The RLS-type algorithms additionally require the update of a block correlation matrix.

See also the comments in Section 5.3.1.

Batch learning algorithm for MIMO inverse modeling

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_{\mathrm{x}} + 1 \geq 2\,(T_{\mathrm{u}} + N_{\mathrm{w}}) + 1 \tag{5.62}$$

$$L = 2\,T_{\mathrm{u}} + 1 \tag{5.63}$$

$$\tilde{\mathbf{s}}_m = (s_{m,0}, \ldots, s_{m,T_{\mathrm{u}}}, 0, \ldots, 0, s_{m,-T_{\mathrm{u}}}, \ldots, s_{m,-1})^T \tag{5.64}$$

$$\overline{\mathbf{S}} = \left[\bar{\mathbf{S}}_m\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{s}}_m)\right] \tag{5.65}$$

$$\tilde{\mathbf{x}}_m = (x_{m,0}, \ldots, x_{m,T_{\mathrm{x}}}, 0, \ldots, 0, x_{m,-T_{\mathrm{x}}}, \ldots, x_{m,-1})^T \tag{5.66}$$

$$\overline{\mathbf{X}} = \left[\bar{\mathbf{X}}_m\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}_m)\right] \tag{5.67}$$

$$\tilde{\mathbf{w}}_{ij,0} = (w_{ij,0}, \ldots, w_{ij,N_{\mathrm{w}}}, 0, \ldots, 0, w_{ij,-N_{\mathrm{w}}}, \ldots, w_{ij,-1})^T \tag{5.68}$$

$$\overline{\mathbf{W}}_0 = \left[\bar{\mathbf{W}}_{ij,0}\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{w}}_{ij,0})\right] \tag{5.69}$$

$$\mathbf{P}_{\tilde{\mathbf{w}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\,\mathbf{F}^{-1} \tag{5.70}$$

For every iteration $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\overline{\mathbf{U}}_k = \overline{\mathbf{W}}_k\,\overline{\mathbf{X}} \tag{5.71}$$

$$\tilde{\mathbf{u}}_{m,k} = \tilde{\mathbf{P}}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\,\mathbf{F}^{-1}\,\mathrm{diag}\left(\bar{\mathbf{U}}_{m,k}\right) \tag{5.72}$$

$$= (u_{m,0}, \ldots, u_{m,T_{\mathrm{u}}}, 0, \ldots, 0, u_{m,-T_{\mathrm{u}}}, \ldots, u_{m,-1})^T \tag{5.73}$$

2. Adaptation error:

$$\tilde{\mathbf{e}}_{\mathrm{s}_{m,k}} = \tilde{\mathbf{s}}_m - \tilde{\mathbf{u}}_{m,k} \tag{5.74}$$

$$\overline{\mathbf{E}}_{\mathrm{s}_k} = \left[\bar{\mathbf{E}}_{\mathrm{s}_{m,k}}\right] = \left[\mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{s}_{m,k}}\right)\right] \tag{5.75}$$

3. Update equations:

$$\overline{\mathbf{W}}'_{k+1} = \text{any update equation from Table E.10} \tag{5.76}$$

$$\bar{\mathbf{W}}_{ij,k+1} = \mathrm{diag}\left(\mathbf{P}_{\tilde{\mathbf{w}}}\ \mathrm{diag}\left(\bar{\mathbf{W}}'_{ij,k+1}\right)\right) \tag{5.77}$$

Block-wise learning algorithm for MIMO inverse modeling

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_{\mathrm{x}} + 1 \geq 2\,(T_{\mathrm{u}} + N_{\mathrm{w}}) + 1 \tag{5.78}$$

$$L = 2\,T_{\mathrm{u}} + 1 \tag{5.79}$$

$$\tilde{\mathbf{w}}_{ij,0} = (w_{ij,0}, \ldots, w_{ij,N_{\mathrm{w}}}, 0, \ldots, 0, w_{ij,-N_{\mathrm{w}}}, \ldots, w_{ij,-1})^T \tag{5.80}$$

$$\overline{\mathbf{W}}_0 = \left[\bar{\mathbf{W}}_{ij,0}\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{w}}_{ij,0})\right] \tag{5.81}$$

$$\mathbf{P}_{\tilde{\mathbf{w}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\,\mathbf{F}^{-1} \tag{5.82}$$

For every block $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\tilde{\mathbf{s}}_{m,k} = (s_{m,kL-d}, \ldots, s_{m,kL+T_{\mathrm{u}}-d}, 0, \ldots, 0, s_{m,kL-T_{\mathrm{u}}-d}, \ldots, s_{m,kL-d-1})^T \tag{5.83}$$

$$\overline{\mathbf{S}}_k = \left[\bar{\mathbf{S}}_{m,k}\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{s}}_{m,k})\right] \tag{5.84}$$

$$\tilde{\mathbf{x}}_{m,k} = (x_{m,kL}, \ldots, x_{m,kL+T_{\mathrm{x}}}, 0, \ldots, 0, x_{m,kL-T_{\mathrm{x}}}, \ldots, x_{m,kL-1})^T \tag{5.85}$$

$$\overline{\mathbf{X}}_k = \left[\bar{\mathbf{X}}_{m,k}\right] = \left[\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}_{m,k})\right] \tag{5.86}$$

$$\overline{\mathbf{U}}_k = \overline{\mathbf{W}}_k\,\overline{\mathbf{X}}_k \tag{5.87}$$

$$\tilde{\mathbf{u}}_{m,k} = \tilde{\mathbf{P}}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\,\mathbf{F}^{-1}\,\mathrm{diag}\left(\bar{\mathbf{U}}_{m,k}\right) \tag{5.88}$$

$$= (u_{m,kL}, \ldots, u_{m,kL+T_{\mathrm{u}}}, 0, \ldots, 0, u_{m,kL-T_{\mathrm{u}}}, \ldots, u_{m,kL-1})^T \tag{5.89}$$

2. Adaptation error:

$$\tilde{\mathbf{e}}_{\mathrm{s}_{m,k}} = \tilde{\mathbf{s}}_{m,k} - \tilde{\mathbf{u}}_{m,k} \tag{5.90}$$

$$\overline{\mathbf{E}}_{\mathrm{s}_k} = \left[\bar{\mathbf{E}}_{\mathrm{s}_{m,k}}\right] = \left[\mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{s}_{m,k}}\right)\right] \tag{5.91}$$

3. Update equations:

$$\overline{\mathbf{W}}'_{k+1} = \text{any update equation from Table E.10} \tag{5.92}$$

$$\bar{\mathbf{W}}_{ij,k+1} = \mathrm{diag}\left(\mathbf{P}_{\tilde{\mathbf{w}}}\ \mathrm{diag}\left(\bar{\mathbf{W}}'_{ij,k+1}\right)\right) \tag{5.93}$$

5.4.2 Block-wise learning for multichannel inverse modeling

This section is the multichannel extension of Section 4.4.3 and 4.4.4.

Fast implementation The whole block-wise learning algorithm for multichannel inverse modeling is given on page 141 from (5.78) to (5.93). We adapt a non-causal filter matrix, to cope also with a nonminimum-phase system $\mathbf{A}(z)$. The algorithm is the multichannel extension of the algorithm described in Section 4.5.3.

The following comment can be made:

- Any update equation listed in Table E.10 can be used for the adaptation. The RLS-type algorithms additionally require the update of a block correlation matrix.

See also the comments in Section 5.3.2.

5.5 Multichannel Wiener filter

We now extend the single-channel Wiener filters, given in Section 4.6 for the system-identification and inverse-modeling problem, to the multichannel case.

Correlation matrices in the z-domain In analogy to the instantaneous mixing problem stated in Chapter 2, we can formulate the *Wiener-Hopf equations* (WHE) for the multichannel convolutive-mixing case. To this end we define the following correlation sequences

$$\mathbf{R}_{\mathbf{xs}}(z) \triangleq \sum_{\tau=-\infty}^{\infty} \mathbf{R}_{\mathbf{xs}_\tau} z^{-\tau} \tag{5.94}$$

where we have in the deterministic case

$$\mathbf{R}_{\mathbf{xs}_\tau} \triangleq \lim_{T\to\infty} \frac{1}{2T+1} \sum_{t=-T}^{T} \mathbf{x}_t \mathbf{s}_{t-\tau}^H \tag{5.95}$$

and in the stochastic case

$$\mathbf{R}_{\mathbf{xs}_\tau} \triangleq E\left\{\mathbf{x}_{t+\tau}\mathbf{s}_t^H\right\} = E\left\{\mathbf{x}_t\mathbf{s}_{t-\tau}^H\right\}. \tag{5.96}$$

Likewise we define $\mathbf{R}_{\mathbf{ss}}(z)$, $\mathbf{R}_{\mathbf{sx}}(z)$, $\mathbf{R}_{\mathbf{xx}}(z)$, $\mathbf{R}_{\mathbf{ux}}(z)$, $\mathbf{R}_{\hat{\mathbf{x}}\mathbf{s}}(z)$, $\mathbf{R}_{\mathbf{e}_\mathrm{x}\mathbf{s}}(z)$, and $\mathbf{R}_{\mathbf{e}_\mathrm{s}\mathbf{x}}(z)$

5.5.1 Multichannel Wiener filter $\mathbf{H}^{\text{MMSE-x}}(z)$

Infinite-length Wiener filter The Wiener-Hopf equation for the multichannel identification problem can be derived by the *orthogonality principle*, which says, that the error-signal vector must be uncorrelated to the input-signal vector

$$\mathbf{R}_{\mathbf{e}_\mathrm{x}\mathbf{s}}(z) = \mathbf{0}. \tag{5.97}$$

For an infinite-length filter $\mathbf{H}(z) = \sum_{n=-\infty}^{\infty} \mathbf{H}_n z^{-n}$ and the error-signal vector $\mathbf{e}_{\mathrm{x}t} = \mathbf{x}_t - \hat{\mathbf{x}}_t$ we have

$$\begin{aligned}\mathbf{R}_{\mathbf{e}_\mathrm{x}\mathbf{s}}(z) &= \mathbf{R}_{\mathbf{xs}}(z) - \mathbf{R}_{\hat{\mathbf{x}}\mathbf{s}}(z) &(5.98)\\ &= \mathbf{R}_{\mathbf{xs}}(z) - \mathbf{H}(z)\mathbf{R}_{\mathbf{ss}}(z). &(5.99)\end{aligned}$$

Using (5.99) in (5.97), we finally obtain the Wiener-Hopf equation

$$\mathbf{H}(z)\mathbf{R}_{\mathbf{ss}}(z) = \mathbf{R}_{\mathbf{xs}}(z) \tag{5.100}$$

Solving (5.100) for $\mathbf{H}(z)$ yields

$$\mathbf{H}^{\text{MMSE-x}}(z) = \mathbf{R}_{\mathbf{xs}}(z)\mathbf{R}_{\mathbf{ss}}^{-1}(z). \tag{5.101}$$

where $\mathbf{H}^{\text{MMSE-x}}(z)$ is the Wiener filter which minimizes $E\left\{\|\mathbf{e}_{\mathrm{x}t}\|_2^2\right\}$. Note the analogy between (5.101), (4.187), and (2.8).

Finite-length Wiener filter The Wiener-Hopf equation for a finite-length multichannel filter $\mathbf{H}(z) = \sum_{n=a}^{b} \mathbf{H}_n z^{-n}$ is given by

$$\mathcal{P}_{a,b}\left(\mathcal{P}_{a,b}\left(\mathbf{H}(z)\right)\mathbf{R}_{\mathbf{ss}}(z)\right) = \mathcal{P}_{a,b}\left(\mathbf{R}_{\mathbf{xs}}(z)\right). \tag{5.102}$$

Furthermore, if the input-signal sequences $s_m(z)$ are white and mutually uncorrelated, we have $\mathbf{R}_{\mathbf{ss}}(z) = \mathbf{R}_{\mathbf{ss}_0}$, and therefore (5.102) simplifies to

$$\mathbf{H}(z)\,\mathbf{R}_{\mathbf{ss}_0} = \mathcal{P}_{a,b}\left(\mathbf{R}_{\mathbf{xs}}(z)\right). \tag{5.103}$$

Solving (5.103) yiels then the finite-length multichannel Wiener filter

$$\mathbf{H}^{\text{MMSE-x}}(z) = \mathcal{P}_{a,b}\left(\mathbf{R}_{\mathbf{xs}}(z)\right)\,\mathbf{R}_{\mathbf{ss}_0}^{-1}\,. \tag{5.104}$$

Note the analogy between (5.104), (4.190), and (2.8).

5.5.2 Multichannel Wiener filter $\mathbf{W}^{\text{MMSE-s}}(z)$

Infinite-length Wiener filter The Wiener-Hopf equation for the multichannel inverse-modeling problem can be derived also by the orthogonality principle. Again, the error-signal vector must be uncorrelated to the input-signal vector which gives now

$$\mathbf{R}_{\mathbf{e_s x}}(z) = \mathbf{0}\,. \tag{5.105}$$

For an infinite-length filter $\mathbf{W}(z) = \sum_{n=-\infty}^{\infty}\mathbf{W}_n z^{-n}$ and the error-signal vector $\mathbf{e}_{\mathrm{s}t} = \mathbf{s}_t - \mathbf{u}_t$ we have

$$\begin{aligned}\mathbf{R}_{\mathbf{e_s x}}(z) &= \mathbf{R}_{\mathbf{sx}}(z) - \mathbf{R}_{\mathbf{ux}}(z) &(5.106)\\ &= \mathbf{R}_{\mathbf{sx}}(z) - \mathbf{W}(z)\mathbf{R}_{\mathbf{xx}}(z)\,. &(5.107)\end{aligned}$$

Using (5.107) in (5.105), we finally obtain the Wiener-Hopf equation

$$\mathbf{W}(z)\mathbf{R}_{\mathbf{xx}}(z) = \mathbf{R}_{\mathbf{sx}}(z)\,. \tag{5.108}$$

Solving (5.108) for $\mathbf{W}(z)$ yields

$$\mathbf{W}^{\text{MMSE-s}}(z) = \mathbf{R}_{\mathbf{sx}}(z)\mathbf{R}_{\mathbf{xx}}^{-1}(z) \tag{5.109}$$

where $\mathbf{W}^{\text{MMSE-s}}(z)$ is the multichannel Wiener filter which minimizes $E\left\{\|e_{\mathrm{s}t}\|_2^2\right\}$. Note the analogy between (5.109), (4.195), and (2.61).

Finite-length Wiener filter In analogy to (5.102), the Wiener-Hopf equation for a finite-length multichannel filter $\mathbf{W}(z) = \sum_{n=a}^{b}\mathbf{W}_n z^{-n}$ is

$$\mathcal{P}_{a,b}\left(\mathcal{P}_{a,b}\left(\mathbf{W}(z)\right)\mathbf{R}_{\mathbf{xx}}(z)\right) = \mathcal{P}_{a,b}\left(\mathbf{R}_{\mathbf{sx}}(z)\right)\,. \tag{5.110}$$

Similar to the single-channel case, (5.110) is not as easily solvable as (5.102), since $x_m(z)$ are non-white, mutually correlated sequences and therefore $\mathbf{R}_{\mathbf{xx}}(z)$ does not consist of a single term. Thus, we have to setup a system of $b-a+1$ linear matrix equations to obtain the filter coefficients $\mathbf{W}_n$. These equations are obtained by evaluating (5.110) for every power z^{τ} for $a \le \tau \le b$. Doing the same steps as from (4.196) to (4.204) yields

$$\sum_{n=a}^{b}\mathbf{W}_n\,\mathbf{R}_{\mathbf{xx}_{\tau-n}} = \mathbf{R}_{\mathbf{sx}_\tau} \qquad (a \le \tau \le b)\,. \tag{5.111}$$

The equations in (5.111) can be written in matrix form

$$\begin{bmatrix}\mathbf{R}_{\mathbf{xx}_0} & \cdots & \mathbf{R}_{\mathbf{xx}_{a-b}}\\ \vdots & \ddots & \vdots\\ \mathbf{R}_{\mathbf{xx}_{b-a}} & \cdots & \mathbf{R}_{\mathbf{xx}_0}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{W}_a\\ \vdots\\ \mathbf{W}_b\end{bmatrix} = \begin{bmatrix}\mathbf{R}_{\mathbf{sx}_a}\\ \vdots\\ \mathbf{R}_{\mathbf{sx}_b}\end{bmatrix}\,. \tag{5.112}$$

Solving this system of linear equations, which involves a matrix inversion of dimension $(b-a+1)M \times (b-a+1)M$, yields the coefficients $\mathbf{W}_n$ of $\mathbf{W}^{\text{MMSE-s}}(z)$. Special cases are with $a=0$ and $b=N_{\mathrm{w}}$ (causal multichannel Wiener filter), or with $a=-N_{\mathrm{w}}$ and $b=N_{\mathrm{w}}$.

5.6 Performance measures

We use the following performance measures for multichannel identification and inverse modeling:

- Average MSE of $\mathbf{e}_{\mathrm{x}t}$ of block k:

$$J_{\text{MSE-x}}(k) = \frac{1}{2T_{\mathrm{x}}+1}\sum_{t=kL-T_{\mathrm{x}}}^{kL+T_{\mathrm{x}}}\|\mathbf{e}_{\mathrm{x}t}\|_F^2 = \frac{1}{2T_{\mathrm{x}}+1}\left\|\mathbf{e}_{\mathrm{x}k}(z)\right\|_{\mathcal{F}}^2\,. \tag{5.113}$$

- Average MSE of $\mathbf{e}_{\mathrm{s}t}$ of block k:

$$J_{\text{MSE-s}}(k) = \frac{1}{2T_{\mathrm{u}}+1}\sum_{t=kL-T_{\mathrm{u}}}^{kL+T_{\mathrm{u}}}\|\mathbf{e}_{\mathrm{s}t}\|_{\mathcal{F}}^2 = \frac{1}{2T_{\mathrm{u}}+1}\left\|\mathbf{e}_{\mathrm{s}k}(z)\right\|_{\mathcal{F}}^2\,. \tag{5.114}$$

- Average block interchannel interference $J_{\text{ICI}}(\mathbf{G}_k(z))$ of block k, where $J_{\text{ICI}}(.)$ is defined in (6.132) and $G_k(z) = \mathbf{W}_k(z)\mathbf{A}(z)$.

- Average block intersymbol interference $J_{\text{ISI}}(\mathbf{G}_k(z))$ of block k, where $J_{\text{ISI}}(.)$ is defined in (6.133).
- Average block multichannel intersymbol interference $J_{\text{MC-ISI}}(\mathbf{G}_k(z))$ of block k, where $J_{\text{MC-ISI}}(.)$ is defined in (6.134).

Alternative performance measures are the signal-to-interference ratio (SIR) or the signal-to-interference-plus-noise ratio (SINR).

5.7 Simulations

In this section two simulation examples are given to illustrate the performance behavior of some algorithms.

5.7.1 Multichannel system identification

The simulation setup is as follows: $M_{\text{s}} = 4$ source signals, $M = 4$ sensors, the source signals are white Gaussian distributed with $\sigma_{\text{s}} = 1$, the sensor noise is white Gaussian distributed with $\sigma_{\text{n}} = 0.01$. The elements of the unknown mixing matrix $\mathbf{A}(z)$ are chosen randomly, $a_{ij}(z)$ are causal filters with $N_{\text{a}} = 200$. The estimation matrix $\mathbf{H}(z)$ has the same number of coefficients as $\mathbf{A}(z)$, i.e. $N_{\text{h}} = 200$, where initially all elements are set to zero. This makes a total of $4 \cdot 4 \cdot 201 \approx 3200$ filter coefficients $h_{ij,n}$ to adapt. We use the LMS1-Hx algorithm given in Table E.6 with block length $L = 1601$ and FFT size $C = 2048$. The channel-wise performance curves are given in Fig. 5.1. $J_{\text{MSE-s}}(k)$ is evaluated with $\mathbf{W}_k(z) = \mathcal{P}_{-N_{\text{w}},N_{\text{w}}}\left(\mathbf{H}_k^{-1}(z)\right)$ with $N_{\text{w}} = 200$.

5.7.2 Multichannel inverse modeling

The simulation setup is as follows: $M_{\text{s}} = 4$ source signals, $M = 4$ sensors, the source signals are white Gaussian distributed with $\sigma_{\text{s}} = 1$, the sensor noise is white Gaussian distributed with $\sigma_{\text{n}} = 0.01$. We take the same 4×4 mixing matrix with $N_{\text{a}} = 2$ as in [70], which is ill conditioned and therefore difficult to separate and deconvolve in the blind case. The non-causal separation matrix $\mathbf{W}(z)$ has $N_{\text{w}} = 300$ and was initially set to $\mathbf{W}_0(z) = \mathbf{I}$. This makes a total of $4 \cdot 4 \cdot 601 \approx 9600$ filter coefficients $w_{ij,n}$ to adapt. We use the LMS1-Wx, RLS1-Wx, and RLS2-Ws algorithms given in Table E.10 with $T_{\text{x}} = 2000$, $T_{\text{u}} = 1700$, block length $L = 3401$ and FFT size $C = 4096$. The parameters of the algorithms are

LMS1-Wx	$\mu = 0.0001$
RLS1-Wx	$\lambda = 0.9$
RLS2-Ws	$\lambda = 0.9$

The channel-wise performance curves are given in Fig. 5.2 to 5.4. $J_{\text{MSE-x}}(k)$ is evaluated with $\mathbf{H}_k(z) = \mathcal{P}_{-N_{\text{h}},N_{\text{h}}}\left(\mathbf{W}_k^{-1}(z)\right)$ and $N_{\text{h}} = 200$. J_{ICI} and J_{ISI} are defined in (6.132) and (6.133), respectively.

5.8 Summary

Once the algorithms for the instantaneous mixing case and the single-channel convolutive case have been derived, the extension to the multichannel case is straightforward with the rules given in Section 5.1. Since we have focused on an implementation in the frequency domain, the algorithms are mainly of interest if we have to adapt filters with many coefficients. For filters with only a few coefficients, it might be worth remaining in the time domain to keep the complexity low.

The adaptation of a SIMO filter with M output signals can be treated as M SISO filters which share the same input signal. For an LMS algorithm, the adaptation of an MISO filter with M input signals can be handled as if there are M SISO filters which have the same desired output signal. The MIMO case is more difficult than the SIMO and MISO case.

In the next chapter, we derive the blind counterparts of the algorithms derived for single- and multichannel inverse modeling, which, in fact, perform the same task, but without knowledge of the source signals.

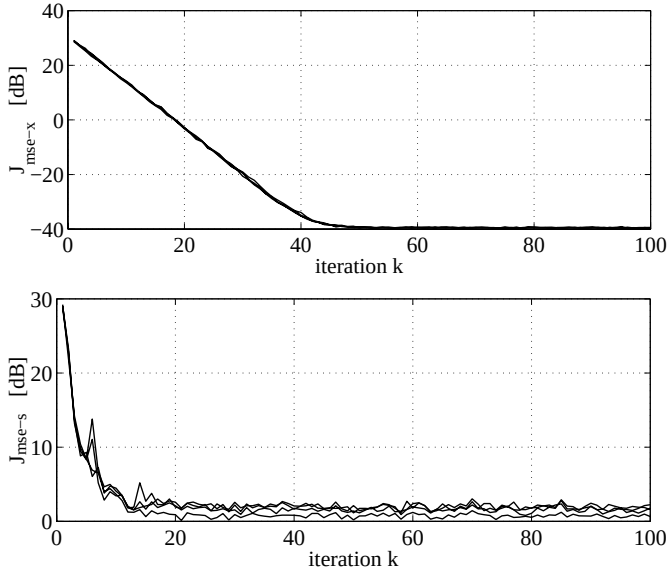

Figure 5.1: Channel-wise performance curves of LMS1-Hx in a multichannel system identification setup: (top) $J_{\text{MSE-x}}(k)$, (bottom) $J_{\text{MSE-s}}(k)$.

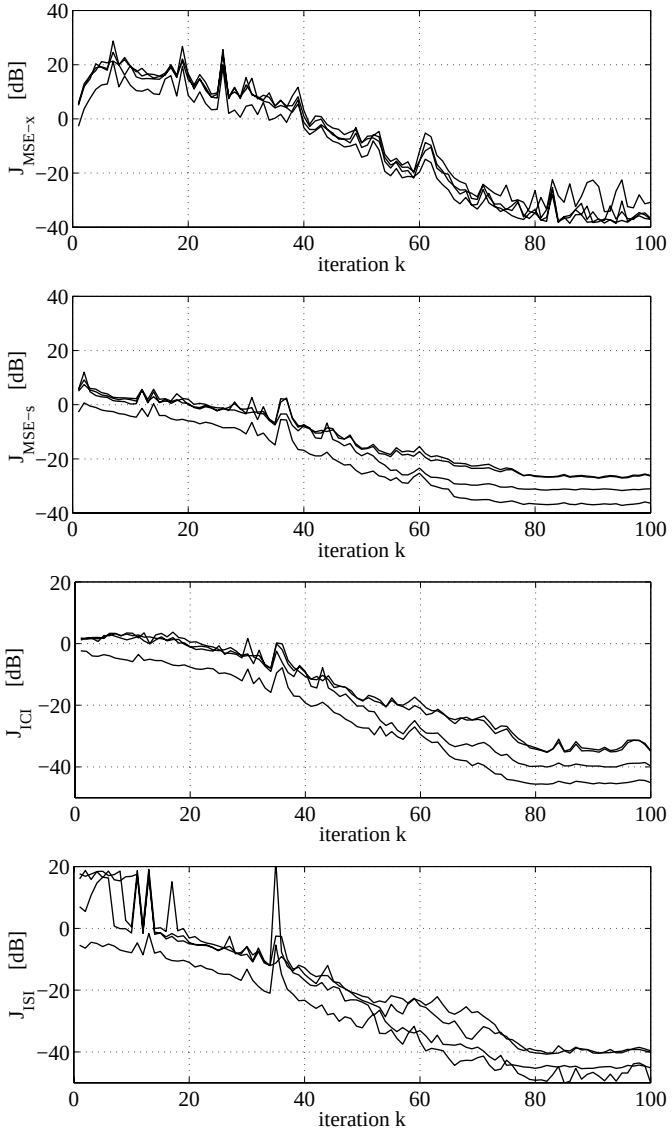

Figure 5.2: Channel-wise performance curves of LMS1-Wx in a multichannel inverse modeling setup: (from top) $J_{\text{MSE-x}}(k)$, $J_{\text{MSE-s}}(k)$, $J_{\text{ICI}}(k)$, and $J_{\text{ISI}}(k)$.

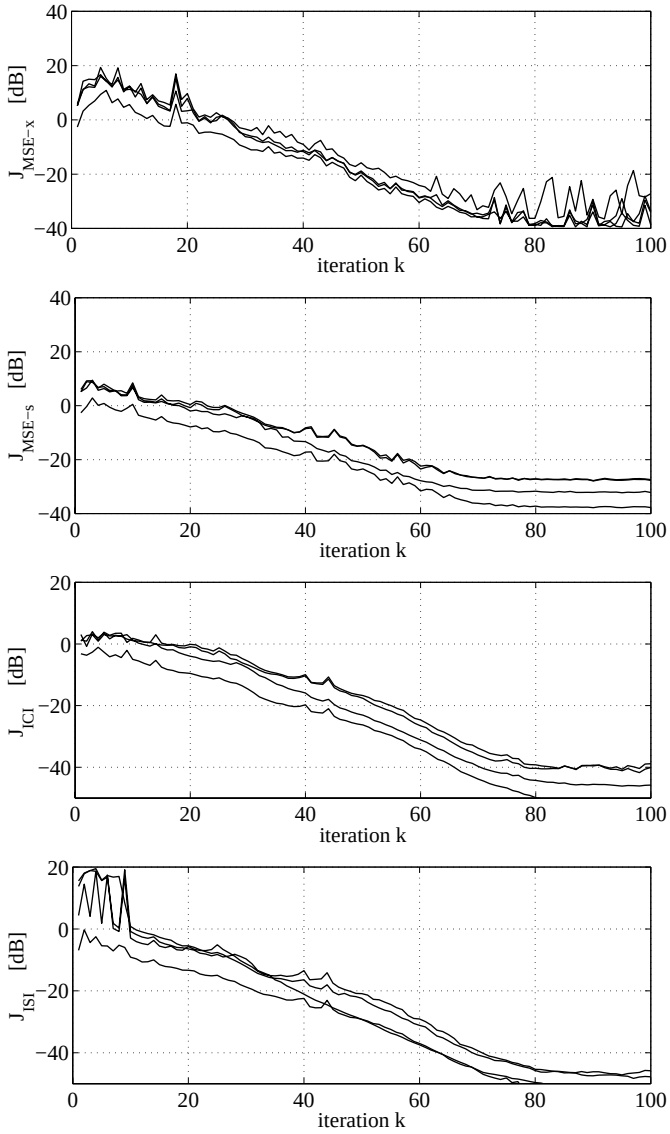

Figure 5.3: Channel-wise performance curves of RLS1-Wx in a multichannel inverse modeling setup: (from top) $J_{\text{MSE-x}}(k)$, $J_{\text{MSE-s}}(k)$, $J_{\text{ICI}}(k)$, and $J_{\text{ISI}}(k)$.

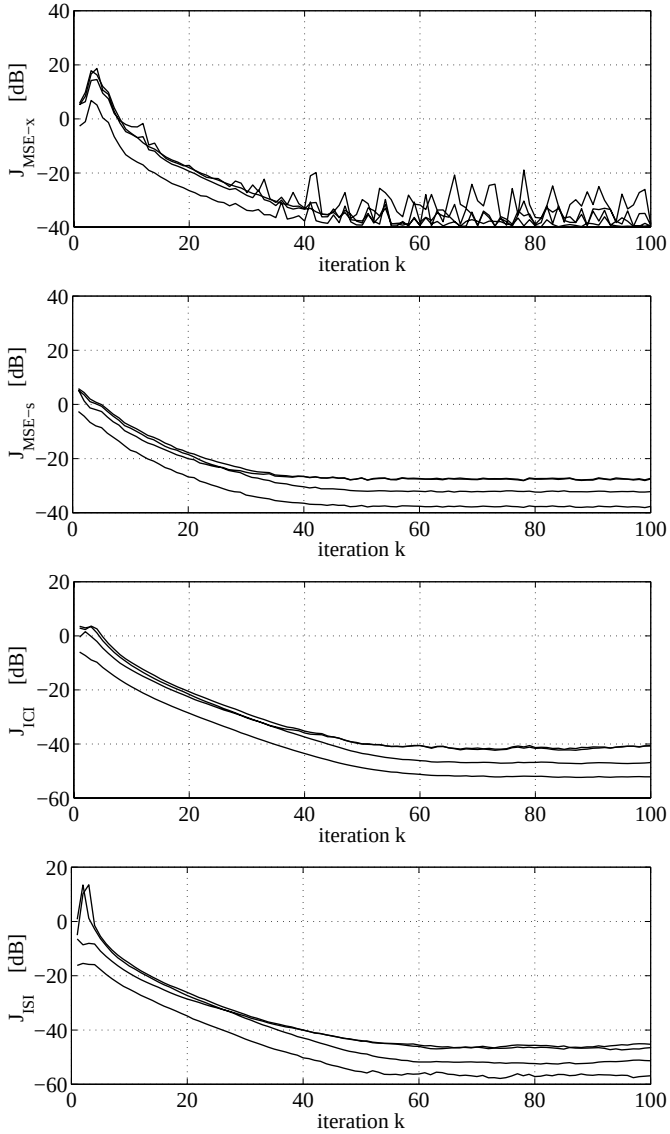

Figure 5.4: Channel-wise performance curves of RLS2-Ws in a multichannel inverse modeling setup: (from top) $J_{\text{MSE-x}}(k)$, $J_{\text{MSE-s}}(k)$, $J_{\text{ICI}}(k)$, and $J_{\text{ISI}}(k)$.

Chapter 6

Blind identification

In this chapter we develop blind algorithms to solve the blind source separation (BSS), blind deconvolution (BD) and multichannel blind deconvolution (MCBD) problem, which are the blind counterparts of the inverse modeling of an instantaneous-mixing system, single-channel convolutive-mixing system, and multichannel convolutive-mixing system, respectively. We therefore make the step from the non-blind algorithms given in Chapter 2, Chapter 4, and Chapter 5 to their blind counterpart by replacing the non-blind error criterion by a blind error criterion.

6.1 Central limit theorem

The *central limit theorem* (CLT) is one of the fundamental results important for the understanding of the blind identification problem. Loosely speaking, the CLT says that the pdf of a sufficiently large sum of independent random numbers converges towards a Gaussian distribution, regardless of the pdf of the individual random numbers. A precise definition of the central limit theorem is given in [85].

Since we assume that the source signals are independent and identically distributed (iid) random variables and mutually independent, the central limit theorem has a direct consequence to a signal mixture or a convolution situa-

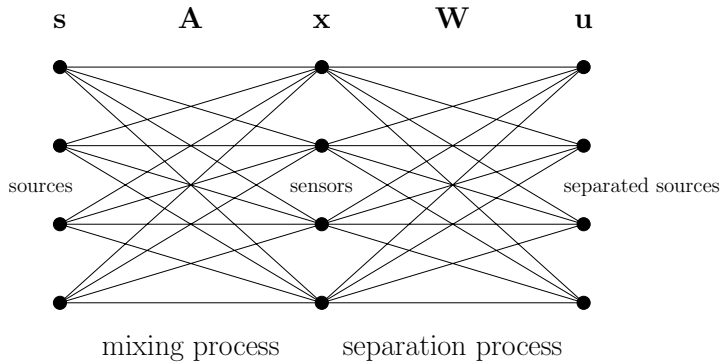

Figure 6.1: Mixing and separation.

tion. For the BSS problem it says that the pdf of the mixed signals (sensor signals) are always *closer* to a Gaussian distribution than the pdf of the individual source signals involved in the mixing process. The same is true for the BD problem, where the pdf of the convolved signal (sensor signal) is always closer to a Gaussian distribution than the pdf of the source signal. Of course the same statement holds for the MCBD problem. In fact, this problem is the hardest among the blind problems, because the output signals will always be very close to a Gaussian pdf, due to the numerous terms in $\mathbf{G}(z) = \mathbf{W}(z)\mathbf{A}(z)$, unless the convolutive separation matrix $\mathbf{W}(z)$ is really close to a true separation matrix, e.g. $\mathbf{A}^{-1}(z)$.

6.2 Assumptions in blind identification

For the BSS problem, if at most one source signal is Gaussian, then it is still possible to separate all source signals. For the BD problem, the source signal has to be non-Gaussian, otherwise no deconvolution is possible, only decorrelation. For the MCBD problem, all source signals have to be non-Gaussian, if separation and deconvolution of the signal mixture is aimed for. In case one source signal is Gaussian distributed, one can still separate all source signals, however, the Gaussian source signal cannot be deconvolved, only decorrelated.

6.3 Cost functions for blind identification

In a blind setup we have no access to the source signals s. This has the important consequence that neither for system identification nor for inverse modeling can we build the error signals required, i.e. e_{x} and e_{s}. We have to find a new *blind* error signal which is derived from the accessible signals of the system, i.e. x and u. We briefly list some of the techniques used in blind identification.

6.3.1 Blind source separation

Entropy

One can show that a uniform pdf has the highest *differential entropy*, a definition of which can be found in (A.1), among all bounded pdfs (compact support). If $p_S(s)$ is known, one can find a nonlinear mapping $y = g(s)$ such that $p_Y(.)$ becomes a uniform distribution $p_1(.)$. The choice of the nonlinearity $g(.)$ depends on $p_S(s)$. However, in a blind setup, we do not know the source signals s_m, but if we know $p_{S_m}(s_m)$, we can calculate $g_m(.)$ and then maximize the sum of the differential entropies of $y_m = g_m(u_m)$, i.e. maximize $\sum_m H(\, p_{Y_m}(g_m(u_m))\,)$, under the constraint that $\det \mathbf{W} \neq 0$. This concept was used in the derivation of the *Infomax* algorithm proposed by Bell and Sejnowski [7].

Mutual information

One possible blind error criterion is the mutual information of the output signals. Since the source signals are assumed to be mutually independent, we want to steer the coefficients of the separation matrix $\mathbf{W}$ such that the output signals become mutually independent again. We can measure the independence of random variables by using the Kullback-Leibler divergence

$$J_{\mathrm{MI}}(p_U(.)) = D\left(p_U(\mathbf{u}) \| \prod\nolimits_m p_{U_m}(u_m)\right) \geq 0\,. \tag{6.1}$$

$p_{U_m}(.)$ denotes the marginal probability density function of U_m. $D\,(.\|.)$ is defined in (A.3). For perfect separation, the cost function J_{MI} becomes zero.

Negentropy

Since the mixing process makes the sensor signals approach a Gaussian pdf, we can steer the separation process such that the output signals u_m are pushed away from a Gaussian distribution. To this end we define an error criterion, which measures the divergence of a pdf to the corresponding Gaussian distribution with the same first and second-order statistics. The *negentropy* is such a criterion and is defined as [40]

$$J_{\text{NE}}(p_U(.)) = D\left(p_U(\mathbf{u}) \| p_G(\mathbf{u})\right) \geq 0 \tag{6.2}$$

where $p_G(\mathbf{u})$ is the pdf of a multivariate Gaussian distribution with the same covariance matrix as $p_U(\mathbf{u})$. Maximizing J_{NE} corresponds to separating the output signals u_m.

Higher-order statistics

If X and Y are two random variables [85], we say that X and Y are *orthogonal* if

$$E\{XY\} = 0 \tag{6.3}$$

and X and Y are *uncorrelated* if

$$E\{(X - E\{X\})(Y - E\{Y\})\} = 0 \tag{6.4}$$

which is equal to

$$E\{XY\} = E\{X\}\,E\{Y\}\,. \tag{6.5}$$

For zero-mean random variables, (6.3) and (6.4) are the same. If X and Y are two random variables, we say that X and Y are *independent* if

$$p_{XY}(x,y) = p_X(x)\,p_Y(y)\,. \tag{6.6}$$

From (6.6) it follows that

$$E\{f(X)g(Y)\} = E\{f(X)\}\,E\{g(Y)\} \tag{6.7}$$

$$E\{X^m Y^n\} = E\{X^m\}\,E\{Y^n\} \qquad \forall m,n \tag{6.8}$$

where $f(.)$ and $g(.)$ are two arbitrary functions.

If two random variables are independent, they are also uncorrelated. However, uncorrelatedness does not necessarily imply independence, except when X and Y are jointly Gaussian distributed. Because (6.5) is a necessary but not sufficient condition for (6.7), second-order moments are not a sufficient statistic for blind identification.

Kurtosis The *kurtosis* is a measure based on the fourth-order statistics of a random variable. The kurtosis of a random variable X is defined as [1]

$$\kappa(X) = \frac{E\{X^4\}}{E\{X^2\}^2} - 3\,. \tag{6.9}$$

Using (6.9), we have $-2 \leq \kappa(X) \leq \infty$ and a Gaussian random variable X has $\kappa(X) = 0$. [67]. Depending on the sign of the kurtosis, we distinguish between *super-Gaussian* ($\kappa > 0$) and *sub-Gaussian* ($\kappa < 0$) distributions. Loosely speaking, a sub-Gaussian pdf looks more flat, e.g. uniform pdf, a super-Gaussian pdf more peaky, e.g. Laplacian pdf. The source signals used in data communications are usually sub-Gaussian, whereas in acoustics, e.g. speech, the source signals are normally super-Gaussian.

If we know for instance that all kurtoses of the source signals have the same sign, we can steer the blind algorithm such that the kurtoses of the output signals are either maximized or minimized.

A nice overview of different techniques for blind source separation is given in [72].

6.3.2 Blind deconvolution

Similar cost functions can be defined for blind deconvolution. Here we wish to remove the dependence in time of the output samples. This means that we wish to adapt the deconvolution or equalization filter $w(z)$ such that the output signal becomes white and non-Gaussian.

[1] Sometimes the -3 is omitted in the definition.

Bussgang property

The Bussgang property is another criterion commonly used in blind algorithms. A signal u is thereby passed through a nonlinearity $y = g(u)$. The autocorrelation of u must then be equal to the cross correlation between u and y. We say that a time-series $u(z)$ *is Bussgang* if $E\{g(u_t)u_{t-\tau}\} = E\{u_t u_{t-\tau}\}$ holds and u_t is independent identically distributed [71].

Memoryless estimator

Another concept is the one of a memoryless nonlinear estimator [8, 42]. Here the output signal u_t is passed through a nonlinearity $f(.)$ such that $f(u_t)$ is used as a nonlinear memoryless estimate of s_t. It is called memoryless, as the estimate does not use any time-delayed output values of u_t. If computable, the conditional mean estimator $\hat{s}_t = f(u_t) = E\{S_t|U_t = u_t\}$ is a suitable choice. Note, if $p_S(.)$ is a normal distribution, then $E\{S_t|U_t = u_t\} = u_t$, i.e., the exact shape of the nonlinear estimator degenerates to a linear estimator. Usually the exact shape of the nonlinearity $f(.)$ is not so crucial. Sometimes it is even enough if the sign of $f(u_t) - u_t$ is correct most of the time. Initially, it may be difficult to find a good estimator anyway.

6.4 Blind error signal

We now wish to find an error signal which can be used in the blind case. The non-blind error signal for inverse modeling is $e_{\mathrm{s}t} = s_t - u_t$. In the blind case, we do not have access to the time samples s_t and therefore have to estimate either s_t or $e_{\mathrm{s}t}$. A reasonable choice would be to use $\hat{s}_t = u_t$, as we have in fact $s_t = u_t$ at convergence in the noise-free case. However, the error signal $\hat{e}_{\mathrm{s}t} = \hat{s}_t - u_t$ would be zero all the time, which is of no help. Thus, if we have only the output signal u_t and the input signal x_t available for building an error signal, we have to introduce a nonlinearity. We follow the concept of a channel-wise memoryless estimator.

Conditional-mean estimator We consider the multichannel convolutive-mixing case. We apply a channel-wise *nonlinear memoryless estimator* such that $\hat{s}_{m,t} = f_m(u_{m,t})$. The error signal $\hat{e}_{\mathrm{s}m,t} = \hat{s}_{m,t} - u_{m,t}$ then becomes

$$\hat{e}_{\mathrm{s}m,t} = f_m(u_{m,t}) - u_{m,t}\,. \tag{6.10}$$

A reasonable estimator is given by the channel-wise evaluation of the *conditional mean*

$$\begin{aligned}
\hat{s}_{m,t} &= E\{S_m|U_m = u_{m,t}\} && (6.11)\\
&= \int s_{m,t} \cdot p_{S_m|U_m}(s_{m,t}|u_{m,t})\, ds_{m,t} && (6.12)\\
&= \int s_{m,t} \cdot \frac{p_{U_m|S_m}(u_{m,t}|s_{m,t}) \cdot p_{S_m}(s_{m,t})}{p_{U_m}(u_{m,t})}\, ds_{m,t}\,. && (6.13)
\end{aligned}$$

In this derivation we used Bayes' theorem. Near convergence, we can approximate $p_{U_m|S_m}(u_{m,t}|s_{m,t})$ by $p_{N_m}(u_{m,t} - s_{m,t})$, where $p_{N_m}(.)$ is a Gaussian distribution with variance $\bar{\sigma_{\mathrm{n}}}^2_m$, which models the multichannel convolutive noise [9], stemming from the interchannel and intersymbol interference due to the equalization mismatch $[\mathbf{I} - \mathbf{W}(z)\mathbf{A}(z)]$. We assume a sensor-noise-free model. We then have

$$\begin{aligned}
\hat{s}_{m,t} &= E\{S_m|u_{m,t}\} && (6.14)\\
&= \int s_{m,t} \cdot \frac{p_{N_m}(u_{m,t} - s_{m,t}) \cdot p_{S_m}(s_{m,t})}{p_{U_m}(u_{m,t})}\, ds_{m,t}\,. && (6.15)
\end{aligned}$$

However, this integral is not easy to solve analytically. Furthermore, we need an estimate of $\bar{\sigma_{\mathrm{n}}}^2_m$, the variance of $p_{N_m}(.)$. Near convergence, $\bar{\sigma_{\mathrm{n}}}^2_m$ will tend

towards zero. Note, that for a perfect separation and deconvolution $p_{U_m}(.) = p_{S_m}(.)$ and $p_{N_m}(.) = \delta(.)$. Eq. (6.14) then becomes $\hat{s}_{m,t} = u_{m,t}$, which concurs with our intuition.

An alternative blind error signal can be defined as

$$\hat{e}_{\mathrm{s}m,t} = s_{m,t} - \hat{u}_{m,t} \tag{6.16}$$
$$= s_{m,t} - g(s_{m,t})\,. \tag{6.17}$$

In fact, this error signal models the situation where $u_{m,t}$ is unknown but $s_{m,t}$ is known. In this case we can use $\hat{u}_{m,t} = g(s_{m,t}) = E\{U_m | S_m = s_{m,t}\}$.

Near convergence, where we have $s_{m,t} \approx u_{m,t}$, we can find a relationship between $f(.)$ and $g(.)$. To this end, we replace $s_{m,t}$ by $u_{m,t}$ in (6.17) and set equal the two error signals $\hat{e}_{\mathrm{s}t}$ from (6.10) and (6.17). We then end up with

$$f(u_{m,t}) - u_{m,t} = u_{m,t} - g(u_{m,t}) \tag{6.18}$$

which can be written as

$$f(u_{m,t}) = 2u_{m,t} - g(u_{m,t}) \tag{6.19}$$
$$g(u_{m,t}) = 2u_{m,t} - f(u_{m,t}) \tag{6.20}$$

or

$$\frac{f(u_{m,t}) + g(u_{m,t})}{2} = u_{m,t}\,. \tag{6.21}$$

Eq. (6.21) has the interpretation that if $f(u_{m,t}) > u_{m,t}$ then $g(u_{m,t}) < u_{m,t}$.

Score function The *score function* of a probability-density function $p_S(.)$ is defined as

$$\varphi_S(s) = -\frac{\partial \log p_S(s)}{\partial s} = \frac{-p'_S(s)}{p_S(s)}\,. \tag{6.22}$$

The score function typically appears in the update equations of many known gradient-based algorithms for blind identification if a maximum-likelihood cost function is used .

6.5 Transforming a non-blind into a blind algorithm

6.5.1 BRLS1

We use the RLS1-Wx as an example to show the steps for the transformation of a non-blind algorithm into a blind one. We assume the instantaneous mixing case. The RLS1-Wx with exponential forgetting is defined in Section 2.4.4 from (2.50) to (2.53), see also Table E.7. We make the usual assumption that the source signals are temporally white and mutually independent. Therefore $\mathbf{R_{ss}}$ is a diagonal matrix and if unknown, we substitute $\mathbf{R_{ss}} = \mathbf{I}$. Thus, there is no need to update $\hat{\mathbf{R}}^{-1}_{\mathbf{ss}_t}$ with (2.52) and (2.53). Under these assumptions, the RLS1-Wx simplifies to

$$\mu_t = \frac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{s}_t^H \mathbf{u}_t} \tag{6.23}$$
$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t\,(\mathbf{s}_t - \mathbf{u}_t)\,\mathbf{s}_t^H\,\mathbf{W}_t\,. \tag{6.24}$$

For the algorithm to work blindly, we somehow have to get rid of $\mathbf{s}_t$, as the source signals are not accessible.

We start with (6.23) which is, in fact, just a self-adjusting step-size control, with forgetting factor λ as a parameter. In fact, μ_t is not involved in the separation process, it merely controls the adaptation rate. Therefore we can replace $\mathbf{s}_t$ by $\mathbf{u}_t$, as $\mathbf{u}_t$ is certainly a reasonable estimate of $\mathbf{s}_t$ near convergence. Moreover, in doing so, we can keep the step size μ_t real valued, even though $\mathbf{u}_t$ might be complex. Otherwise μ_t might introduce an unwanted complex rotation in (6.24). Keeping the step size real valued is especially useful in the convolutive case, because μ_t becomes dependent on the frequency. In fact, $\mu_t(\omega)$ corresponds to a bin-wise step-size normalization, a well-known technique from adaptive filtering in the frequency domain to accelerate the convergence rate for non-white input signals [77]. A complex valued $\mu_t(\omega)$ would introduce an additional phase shift into the update equation. However, once the output signals are temporally white, $\mathbf{u}_t^H(z)\mathbf{u}_t(z)$, and hence $\mu_t(\omega) = \mu_t$, become frequency independent.

We can rewrite the update equation (6.24) as

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t\left(\mathbf{s}_t\mathbf{s}_t^H - \mathbf{u}_t\mathbf{s}_t^H\right)\mathbf{W}_t\,. \tag{6.25}$$

We now have several choices for replacing each $\mathbf{s}_t$ in (6.25) by either a linear estimate $\hat{\mathbf{s}}_t = \mathbf{u}_t$ or a nonlinear estimate $\hat{\mathbf{s}}_t = \mathbf{f}(\mathbf{u}_t)$. At least one $\mathbf{s}_t$ has to be replaced by a nonlinear estimate, otherwise the term in the bracket will be zero all time. In the derivation of stochastic-gradient-based algorithms, e.g., the LMS algorithm, a correlation matrix is sometimes replaced by an instantaneous estimate for two reasons. The first is because the correlation matrices are not known a priori and have to be estimated anyway, and the second is that a multiplication with an outer product (matrix with rank one) requires fewer element-wise multiplications than with a regular matrix. However, in our case it is just the other way round. We have an outer product $\mathbf{s}_t\mathbf{s}_t^H$ from which we know the expectation $E\left\{\mathbf{s}_t\mathbf{s}_t^H\right\} = \mathbf{R}_{\mathbf{ss}} = \mathbf{I}$. Thus, we have another possibility to transform (6.24) into a blind algorithm, namely

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t \left(\mathbf{I} - \mathbf{u}_t \mathbf{f}^H(\mathbf{u}_t)\right) \mathbf{W}_t \,. \tag{6.26}$$

If we now apply the *transpose property* [2, 109], which is based on a stability analysis of blind algorithms, we can replace $\mathbf{u}_t \mathbf{f}^H(\mathbf{u}_t)$ by $\mathbf{g}(\mathbf{u}_t)\,\mathbf{u}_t^H$ where the nonlinearities $g_m(.)$ have different characteristics than the $f_m(.)$. Loosely speaking, if $f_m(.)$ is a nonlinearity which can separate a sub-Gaussian signal, then $g_m(.)$ has to be a nonlinearity which can separate a super-Gaussian signal. In doing so, we end up with

$$\mu_t = \frac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{u}_t^H \mathbf{u}_t} \tag{6.27}$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t \left(\mathbf{I} - \mathbf{g}(\mathbf{u}_t)\,\mathbf{u}_t^H\right) \mathbf{W}_t \,. \tag{6.28}$$

We refer to (6.27) and (6.28) as BRLS1, where $g_m(.)$ is usually chosen as the score function $\varphi_{S_m}(.)$ defined in (6.22).

Surprisingly, the update equation (6.28) without (6.27) is the well-known *natural-gradient learning algorithm* proposed by Amari *et al.*, which is one of the most powerful algorithms known for blind source separation. Note, that (6.27) and (6.28) were derived here in a completely different manner than the one shown in [4].

The following comments can be made:

- The natural-gradient learning algorithm has the so-called *equivariant property*, which says that in the noiseless case the convergence behavior depends on the current global system $\mathbf{G}_t = \mathbf{W}_t\,\mathbf{A}$ and not only on $\mathbf{A}$.

- In fact, the RLS1-Wx is derived from the RLS1-Hx, which is an algorithm for system identification and therefore adapts $\mathbf{H} = \mathbf{W}^{-1}$ which is an estimate of $\mathbf{A}$. In the system identification problem, the convergence rate is independent of the mixing matrix $\mathbf{A}$ or $\mathbf{A}(z)$, as long as the source signals s_m are white and mutually uncorrelated. Hence, from Chapter 2 we know that RLS1-Wx has a fast convergence behavior which is robust against the conditioning of the mixing matrix $\mathbf{A}$. As we merely exchanged a non-blind error criterion with a blind one in the derivation of the BRLS1, but not the part which controls the direction of the gradient, it is not very surprisingly that (6.28), and therefore the natural-gradient learning algorithm, reveals the equivariant property.

- The RLS1-Wx as well as the blind counterpart BRLS1 belong to the class of so-called *serial-update algorithms* [17], as we can reformulate (6.28) recursively as

$$\mathbf{W}_{t+1} = \triangle\mathbf{W}_t \cdot \mathbf{W}_t = \left(\prod\nolimits_{\tau=0}^{t} \triangle\mathbf{W}_\tau\right) \cdot \mathbf{W}_0 \tag{6.29}$$

  with $\triangle\mathbf{W}_t = \mathbf{I} + \mu_t \left(\mathbf{I} - \mathbf{g}(\mathbf{u}_t)\,\mathbf{u}_t^H\right)$. From Chapter 2 and Table E.7 we see that this is not a property of blind algorithms alone. In fact, all algorithms for inverse modeling in Table E.7, which have their roots in system identification, see Table E.3, belong to the class of serial-update algorithms, and vice versa. They are originally designed to adapt an estimate $\mathbf{H} = \hat{\mathbf{A}}$ and from applying the matrix-inversion lemma, they become serial-update algorithms which adapt $\mathbf{W} = \hat{\mathbf{A}}^{-1}$. Recall from the simulation examples in Section 2.10, that the serial update algorithms for system identification showed slower convergence rates than the non-serial ones.

6.5.2 General rules for transforming a non-blind algorithm into a blind one

Nonlinearity To obtain not only mutually uncorrelated but mutually independent output signals, we have to introduce at least one nonlinearity into the update equation. The same is true for blind deconvolution to achieve not only temporally uncorrelated but temporally independent output signals.

If $f(.)$ and $g(.)$ are two nonlinearities, we denote the output of the nonlin-

earity as

$$y_t \triangleq f(u_t) \qquad \text{or} \qquad y_t \triangleq g(u_t)\,. \tag{6.30}$$

The corresponding time-series in the z-domain is

$$y(z) \triangleq \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} y_t z^{-t}\,. \tag{6.31}$$

We prefer to write $y(z) = \sum g(u_t) z^{-t}$ than $g(u(z))$, because the latter denotes a nonlinearity applied to a polynomial. The same is true for the frequency domain, we prefer $y(\omega)$ than $g(u(\omega))$, as we apply the nonlinearity in the time domain and not in the frequency domain.

In the multichannel case, we define the output of the multichannel nonlinearity as

$$\mathbf{y}_t \triangleq \mathbf{f}(\mathbf{u}_t) \qquad \text{or} \qquad \mathbf{y}_t \triangleq \mathbf{g}(\mathbf{u}_t) \tag{6.32}$$

$$\mathbf{y}_{m,t} \triangleq f_m(\mathbf{u}_{m,t}) \qquad \text{or} \qquad \mathbf{y}_{m,t} \triangleq f_m(\mathbf{u}_{m,t})\,. \tag{6.33}$$

The corresponding time-series in the z-domain is

$$\mathbf{y}(z) \triangleq \sum_{t=-T_\mathrm{u}}^{T_\mathrm{u}} \mathbf{y}_t z^{-t}\,. \tag{6.34}$$

Transformation As we have seen in Section 6.5.1, there is no unique transformation of a nonblind algorithm into a blind one. In Table 6.1 we list some possible replacements. However, there are often several possibilities and to see which one has the best performance, one has to carry out some simulations or try to analyze the stability conditions [2]. Note, that at least one nonlinearity must appear in a blind update equation, otherwise the output signals will only be uncorrelated, but not independent. The *transpose property* [109] is also a very useful tool to find a dual update equation. Finally, there is no guaranty that an algorithm is stable and converges towards a global minimum.

Transforming an algorithm for multichannel inverse modeling to a multichannel blind-deconvolution algorithm is straightforward. The update equations for BSS, BD, and MCBD are given in Table E.11, Table E.13, and Table E.14; they are the blind counterparts of those given in Table E.7, Table E.9,

non-blind	blind I	blind II
$\mathbf{e}_\mathrm{s} = \mathbf{s} - \mathbf{u}$	$\mathbf{e}_\mathrm{b} = \mathbf{y} - \mathbf{u}$	$\mathbf{e}_\mathrm{b} = \mathbf{u} - \mathbf{y}$
$\mathbf{y}$	$\mathbf{f}(\mathbf{u})$	$\mathbf{g}(\mathbf{u})$
$\mathbf{s}$	$\mathbf{y}$	$\mathbf{u}$
$\mathbf{u}$	$\mathbf{u}$	$\mathbf{y}$
$\mathbf{R}_{\mathbf{ss}}$	$\mathbf{R}_{\mathbf{yy}}$	$\mathbf{I}$
$\mathbf{R}_{\mathbf{uu}}$	$\mathbf{R}_{\mathbf{uu}}$	$\mathbf{I}$
$\mathbf{R}_{\mathbf{su}}$	$\mathbf{R}_{\mathbf{yu}}$	$\mathbf{R}_{\mathbf{uy}}$
$\mathbf{R}_{\mathbf{us}}$	$\mathbf{R}_{\mathbf{uy}}$	$\mathbf{R}_{\mathbf{yu}}$
$\mathbf{R}_{\mathbf{xx}}$	$\mathbf{W}^{-1}\mathbf{R}_{\mathbf{uu}}\mathbf{W}^{-H}$	or $\mathbf{W}^{-1}\mathbf{W}^{-H}$
$\mathbf{R}_{\mathbf{ux}}$	$\mathbf{R}_{\mathbf{uu}}\mathbf{W}^{-H}$	or $\mathbf{W}^{-H}$
$\mathbf{R}_{\mathbf{xu}}$	$\mathbf{W}^{-1}\mathbf{R}_{\mathbf{uu}}$	or $\mathbf{W}^{-1}$

Table 6.1: Mapping of variables to transform an algorithm for inverse modeling into a blind algorithm.

and Table E.10, respectively. Note, since the cost functions, and also the update equations are originally formulated in the time domain, these algorithms do not have a so-called *permutation problem*, which typically appears when the cost function is formulated solely in the frequency domain in a bin-wise manner. These algorithms presented here carry out the update in the frequency domain only for efficiency reasons.

6.6 Orthogonality principle and Bussgang property

In Section 5.5.2 we have derived the optimal infinite-length Wiener filter with the help of the orthogonality principle $\mathbf{R}_{\mathbf{e_s x}}(z) = \mathbf{0}$ [85]. If we replace the error signal $\mathbf{e}_\mathrm{s}$ by a blind error signal $\mathbf{e}_{\mathrm{b}t} = \mathbf{f}(\mathbf{u}_t) - \mathbf{u}_t$ or $\mathbf{e}_{\mathrm{b}t} = \mathbf{u}_t - \mathbf{g}(\mathbf{u}_t)$, and require again that the error-signal vector must be uncorrelated to the input-signal vector, the orthogonality principle becomes

$$\mathbf{R}_{\mathbf{e_b x}}(z) = \mathbf{0}\,. \tag{6.35}$$

For an infinite-length filter $\mathbf{W}(z) = \sum_{n=-\infty}^{\infty} \mathbf{W}_n z^{-n}$ and $\mathbf{e}_{\mathrm{b}t} = \mathbf{y}_t - \mathbf{u}_t$ we have

$$\mathbf{R}_{\mathbf{e_b x}}(z) = \mathbf{R}_{\mathbf{yx}}(z) - \mathbf{R}_{\mathbf{ux}}(z) = \mathbf{0} \tag{6.36}$$

or

$$\mathbf{R}_{\mathbf{yx}}(z) = \mathbf{R}_{\mathbf{ux}}(z)\,. \tag{6.37}$$

Postmultiplying both sides of (6.37) by $\mathbf{W}^H(z)$ gives

$$\mathbf{R}_{\mathbf{yu}}(z) = \mathbf{R}_{\mathbf{uu}}(z) \tag{6.38}$$

which is known as the *Bussgang property* [9]. Evaluating (6.38) for every power of z yields

$$\mathbf{R}_{\mathbf{yu}_\tau} = \mathbf{R}_{\mathbf{uu}_\tau} \tag{6.39}$$

$$E\left\{\mathbf{y}_{t+\tau}\mathbf{u}_t^H\right\} = E\left\{\mathbf{u}_{t+\tau}\mathbf{u}_t^H\right\}\,. \tag{6.40}$$

From (6.37) and (6.38) it follows that

$$\mathbf{R}_{\mathbf{yx}}(z) = \mathbf{W}(z)\mathbf{R}_{\mathbf{xx}}(z)\,. \tag{6.41}$$

In contrast to (5.108), (6.41) cannot be solved directly for $\mathbf{W}(z)$ because $\mathbf{y}(z)$ depends on $\mathbf{W}(z)$ in a nonlinear fashion.

The Bussgang property can be used to build a blind cost function

$$J_\mathrm{B} = \left\|\mathbf{R}_{\mathbf{uu}}(z) - \mathbf{R}_{\mathbf{yu}}(z)\right\|_\mathcal{F} \tag{6.42}$$

with the appropriate choice of the nonlinearity. A modified version of (6.42) is used in the natural-gradient learning algorithm, where $\mathbf{R}_{\mathbf{uu}}(z)$ is replaced by its expectation at convergence $\mathbf{R}_{\mathbf{ss}} = \mathbf{I}$.

6.7 Blind source separation (BSS)

Algorithm	Update equations
Infomax [7]	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\left(\mathbf{W}_t^{-H} - \mathbf{y}_t\mathbf{x}_t^H\right)$
Natural gradient [4]	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\left(\mathbf{I} - \mathbf{y}_t\mathbf{u}_t^H\right)\mathbf{W}_t$
EASI [17]	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\left(\mathbf{I} - \mathbf{u}_t\mathbf{u}_t^H + \mathbf{u}_t\mathbf{y}_t^H - \mathbf{y}_t\mathbf{u}_t^H\right)\mathbf{W}_t$

Table 6.2: Update equations for blind source separation.

This section is the "blind" counterpart of inverse modeling of an instantaneous-mixing system, decribed in Section 2.5.

The update equations are given in Table 6.2 and Table E.11.

6.8 Blind deconvolution (BD)

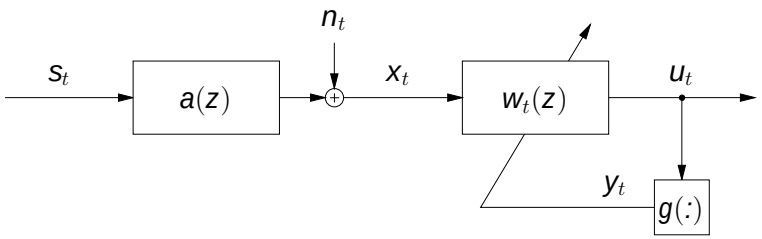

Figure 6.2: Single-channel blind deconvolution. The filter $w(z)$ is adapted such that $g(z) = w(z)a(z) \approx z^{-d}$. The delay d depends partially on the initial value $w_0(z)$.

This section is the "blind" counterpart of the single-channel inverse modeling problem decribed in Section 2.5. Blind deconvolution is also known as blind equalization, which is more common in data communications.

Batch learning algorithm for blind deconvolution

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_{\mathrm{x}} + 1 \geq 2(T_{\mathrm{u}} + N_{\mathrm{w}}) + 1 \tag{6.43}$$
$$L = 2\,T_{\mathrm{u}} + 1 \tag{6.44}$$
$$\tilde{\mathbf{x}} = (x_0, \ldots, x_{T_{\mathrm{x}}}, 0, \ldots, 0, x_{-T_{\mathrm{x}}}, \ldots, x_{-1})^T \tag{6.45}$$
$$\bar{\mathbf{X}} = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}) \tag{6.46}$$
$$\tilde{\mathbf{w}}_0 = (w_0, \ldots, w_{N_{\mathrm{w}}}, 0, \ldots, 0, w_{-N_{\mathrm{w}}}, \ldots, w_{-1})^T \tag{6.47}$$
$$\bar{\mathbf{W}}_0 = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{w}}_0) \tag{6.48}$$
$$\mathbf{P}_{\tilde{\mathbf{w}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\,\mathbf{F}^{-1} \tag{6.49}$$

For every iteration $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\bar{\mathbf{U}}_k = \bar{\mathbf{W}}_k\,\bar{\mathbf{X}} \tag{6.50}$$
$$\tilde{\mathbf{u}}_k = \tilde{\mathbf{P}}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\,\mathbf{F}^{-1}\,\mathrm{diag}\,(\bar{\mathbf{U}}_k) \tag{6.51}$$
$$= (u_0, \ldots, u_{T_{\mathrm{u}}}, 0, \ldots, 0, u_{-T_{\mathrm{u}}}, \ldots, u_{-1})^T \tag{6.52}$$

2. Adaptation error:

$$\tilde{\mathbf{y}}_k = g\,(\tilde{\mathbf{u}}_k) \tag{6.53}$$
$$= (y_0, \ldots, y_{T_{\mathrm{u}}}, 0, \ldots, 0, y_{-T_{\mathrm{u}}}, \ldots, y_{-1})^T \tag{6.54}$$
$$\bar{\mathbf{Y}}_k = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{y}}_k) \tag{6.55}$$
$$\tilde{\mathbf{e}}_{\mathrm{b}_k} = \tilde{\mathbf{u}}_k - \tilde{\mathbf{y}}_k \tag{6.56}$$
$$\bar{\mathbf{E}}_{\mathrm{b}_k} = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{b}_k}) \tag{6.57}$$

3. Update equations:

$$\bar{\mathbf{W}}'_{k+1} = \text{any update equation from Table 6.3 or Table E.13} \tag{6.58}$$
$$\bar{\mathbf{W}}_{k+1} = \mathrm{diag}\,(\mathbf{P}_{\tilde{\mathbf{w}}}\ \mathrm{diag}\,(\bar{\mathbf{W}}'_{k+1})) \tag{6.59}$$

Block-wise learning algorithm for blind deconvolution

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_{\mathrm{x}} + 1 \geq 2(T_{\mathrm{u}} + N_{\mathrm{w}}) + 1 \tag{6.60}$$
$$L = 2\,T_{\mathrm{u}} + 1 \tag{6.61}$$
$$\tilde{\mathbf{w}}_0 = (w_0, \ldots, w_{N_{\mathrm{w}}}, 0, \ldots, 0, w_{-N_{\mathrm{w}}}, \ldots, w_{-1})^T \tag{6.62}$$
$$\bar{\mathbf{W}}_0 = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{w}}_0) \tag{6.63}$$
$$\mathbf{P}_{\tilde{\mathbf{w}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\,\mathbf{F}^{-1} \tag{6.64}$$

For every block $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\tilde{\mathbf{x}}_k = (x_{kL}, \ldots, x_{kL+T_{\mathrm{x}}}, 0, \ldots, 0, x_{kL-T_{\mathrm{x}}}, \ldots, x_{kL-1})^T \tag{6.65}$$
$$\bar{\mathbf{X}}_k = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}_k) \tag{6.66}$$
$$\bar{\mathbf{U}}_k = \bar{\mathbf{W}}_k\,\bar{\mathbf{X}}_k \tag{6.67}$$
$$\tilde{\mathbf{u}}_k = \tilde{\mathbf{P}}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\,\mathbf{F}^{-1}\,\mathrm{diag}\,(\bar{\mathbf{U}}_k) \tag{6.68}$$
$$= (u_{kL}, \ldots, u_{kL+T_{\mathrm{u}}}, 0, \ldots, 0, u_{kL-T_{\mathrm{u}}}, \ldots, u_{kL-1})^T \tag{6.69}$$

2. Adaptation error:

$$\tilde{\mathbf{y}}_k = g\,(\tilde{\mathbf{u}}_k) \tag{6.70}$$
$$= (y_{kL}, \ldots, y_{kL+T_{\mathrm{u}}}, 0, \ldots, 0, y_{kL-T_{\mathrm{u}}}, \ldots, y_{kL-1})^T \tag{6.71}$$
$$\bar{\mathbf{Y}}_k = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{y}}_k) \tag{6.72}$$
$$\tilde{\mathbf{e}}_{\mathrm{b}_k} = \tilde{\mathbf{u}}_k - \tilde{\mathbf{y}}_k \tag{6.73}$$
$$\bar{\mathbf{E}}_{\mathrm{b}_k} = \mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{b}_k}) \tag{6.74}$$

3. Update equations:

$$\bar{\mathbf{W}}'_{k+1} = \text{any update equation from Table 6.3 or Table E.13} \tag{6.75}$$
$$\bar{\mathbf{W}}_{k+1} = \mathrm{diag}\,(\mathbf{P}_{\tilde{\mathbf{w}}}\ \mathrm{diag}\,(\bar{\mathbf{W}}'_{k+1})) \tag{6.76}$$

Algorithm	Update equations
Infomax	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu \left(\bar{\mathbf{W}}_k^{-H} - \bar{\mathbf{Y}}_k \bar{\mathbf{X}}_k^H \right)$
Natural gradient	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu \left(\mathbf{I} - \bar{\mathbf{Y}}_k \bar{\mathbf{U}}_k^H \right) \bar{\mathbf{W}}_k$
EASI	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu \left(\mathbf{I} - \bar{\mathbf{U}}_k \bar{\mathbf{U}}_k^H + \bar{\mathbf{U}}_k \bar{\mathbf{Y}}_k^H - \bar{\mathbf{Y}}_k \bar{\mathbf{U}}_k^H \right) \bar{\mathbf{W}}_k$

Table 6.3: Update equations for single-channel blind deconvolution.

6.8.1 Batch learning algorithm for blind deconvolution

This section is the "blind" counterpart to Section 4.4.2 and Section 4.5.2.

The whole batch learning algorithm for single-channel blind deconvolution is given on page 168 from (6.43) to (6.59). Since we have all data available, we adapt a non-causal filter.

The following comments can be made:

- The update equations are given in Table 6.3 and Table E.13.

6.8.2 Block-wise learning algorithm for blind deconvolution

This section is the "blind" counterpart to Section 4.4.3 and Section 4.5.3.

The whole block-wise learning algorithm for single-channel blind deconvolution is given on page 169 from (6.60) to (6.76). The update equations are given in Table 6.3 and Table E.13. We adapt a delayed non-causal filter, the origin is shifted by a delay, to cope with a nonminimum-phase system $a(z)$. The comments in Section 6.8.1 hold also for the block-wise learning algorithm. A MATLAB implementation of the algorithm is given in Appendix F. An alternative frequency-domain based blind deconvolution algorithm is presented by Douglas and Kung in [35].

Algorithm	Update equations
Infomax	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu \left(\overline{\mathbf{W}}_k^{-H} - \overline{\mathbf{Y}}_k \overline{\mathbf{X}}_k^H \right)$
Natural gradient	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu \left(\mathbf{I} - \overline{\mathbf{Y}}_k \overline{\mathbf{U}}_k^H \right) \overline{\mathbf{W}}_k$
EASI	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu \left(\mathbf{I} - \overline{\mathbf{U}}_k \overline{\mathbf{U}}_k^H + \overline{\mathbf{U}}_k \overline{\mathbf{Y}}_k^H - \overline{\mathbf{Y}}_k \overline{\mathbf{U}}_k^H \right) \overline{\mathbf{W}}_k$

Table 6.4: Update equations for multichannel blind deconvolution.

6.9 Multichannel blind deconvolution (MCBD)

This section is the "blind" counterpart of the multichannel inverse modeling problem decribed in Section 5.4.

6.9.1 Batch learning algorithm for multichannel blind deconvolution

This section is the "blind" counterpart to Section 5.4.1.

The whole batch learning algorithm for multichannel blind deconvolution is given on page 172 from (6.77) to (6.93). The update equations are given in Table 6.4 and Table E.14. Since we have all data available, we can adapt a non-causal filter.

6.9.2 Block-wise learning algorithm for multichannel blind deconvolution

This section is the "blind" counterpart to Section 5.4.2.

The whole block-wise learning algorithm for multichannel blind deconvolution is given on page 173 from (6.94) to (6.110). The update equations are given in Table 6.4 and Table E.14. We adapt a delayed non-causal filter, the origin is shifted by a delay, to cope with a nonminimum-phase system $a(z)$.

Batch learning algorithm for multichannel blind deconvolution

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_{\mathrm{x}} + 1 \geq 2\,(T_{\mathrm{u}} + N_{\mathrm{w}}) + 1 \tag{6.77}$$
$$L = 2\,T_{\mathrm{u}} + 1 \tag{6.78}$$
$$\tilde{\mathbf{x}}_m = (x_{m,0}, \ldots, x_{m,T_{\mathrm{x}}}, 0, \ldots, 0, x_{m,-T_{\mathrm{x}}}, \ldots, x_{m,-1})^T \tag{6.79}$$
$$\overline{\mathbf{X}} = \left[\bar{\mathbf{X}}_m\right] = [\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}_m)] \tag{6.80}$$
$$\tilde{\mathbf{w}}_{ij,0} = (w_{ij,0}, \ldots, w_{ij,N_{\mathrm{w}}}, 0, \ldots, 0, w_{ij,-N_{\mathrm{w}}}, \ldots, w_{ij,-1})^T \tag{6.81}$$
$$\overline{\mathbf{W}}_0 = \left[\bar{\mathbf{W}}_{ij,0}\right] = [\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{w}}_{ij,0})] \tag{6.82}$$
$$\mathbf{P}_{\tilde{\mathbf{w}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\,\mathbf{F}^{-1} \tag{6.83}$$

For every iteration $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\overline{\mathbf{U}}_k = \overline{\mathbf{W}}_k\,\overline{\mathbf{X}} \tag{6.84}$$
$$\tilde{\mathbf{u}}_{m,k} = \tilde{\mathbf{P}}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\,\mathbf{F}^{-1}\,\mathrm{diag}\left(\bar{\mathbf{U}}_{m,k}\right) \tag{6.85}$$
$$= (u_{m,0}, \ldots, u_{m,T_{\mathrm{u}}}, 0, \ldots, 0, u_{m,-T_{\mathrm{u}}}, \ldots, u_{m,-1})^T \tag{6.86}$$

2. Adaptation error:

$$\tilde{\mathbf{y}}_{m,k} = g_m\,(\tilde{\mathbf{u}}_{m,k}) \tag{6.87}$$
$$= (y_{m,0}, \ldots, y_{m,T_{\mathrm{u}}}, 0, \ldots, 0, y_{m,-T_{\mathrm{u}}}, \ldots, y_{m,-1})^T \tag{6.88}$$
$$\overline{\mathbf{Y}}_k = \left[\bar{\mathbf{Y}}_{m,k}\right] = [\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{y}}_{m,k})] \tag{6.89}$$
$$\tilde{\mathbf{e}}_{\mathrm{b}_{m,k}} = \tilde{\mathbf{u}}_{m,k} - \tilde{\mathbf{y}}_{m,k} \tag{6.90}$$
$$\overline{\mathbf{E}}_{\mathrm{b}_k} = \left[\bar{\mathbf{E}}_{\mathrm{b}_{m,k}}\right] = \left[\mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{b}_{m,k}}\right)\right] \tag{6.91}$$

3. Update equations:

$$\overline{\mathbf{W}}'_{k+1} = \text{any update equation from Table 6.4 or Table E.14} \tag{6.92}$$
$$\bar{\mathbf{W}}_{ij,k+1} = \mathrm{diag}\left(\mathbf{P}_{\tilde{\mathbf{w}}}\ \mathrm{diag}\left(\bar{\mathbf{W}}'_{ij,k+1}\right)\right) \tag{6.93}$$

Block-wise learning algorithm for multichannel blind deconvolution

Definitions and initialization $(k = 0)$:

$$C \geq 2\,T_{\mathrm{x}} + 1 \geq 2\,(T_{\mathrm{u}} + N_{\mathrm{w}}) + 1 \tag{6.94}$$
$$L = 2\,T_{\mathrm{u}} + 1 \tag{6.95}$$
$$\tilde{\mathbf{w}}_{ij,0} = (w_{ij,0}, \ldots, w_{ij,N_{\mathrm{w}}}, 0, \ldots, 0, w_{ij,-N_{\mathrm{w}}}, \ldots, w_{ij,-1})^T \tag{6.96}$$
$$\overline{\mathbf{W}}_0 = \left[\bar{\mathbf{W}}_{ij,0}\right] = [\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{w}}_{ij,0})] \tag{6.97}$$
$$\mathbf{P}_{\tilde{\mathbf{w}}} = \mathbf{F}\,\tilde{\mathbf{P}}_{-N_{\mathrm{w}},N_{\mathrm{w}}}\,\mathbf{F}^{-1} \tag{6.98}$$

For every block $k = 1, 2, 3, \ldots$:

1. Filtering:

$$\tilde{\mathbf{x}}_{m,k} = (x_{m,kL}, \ldots, x_{m,kL+T_{\mathrm{x}}}, 0, \ldots, 0, x_{m,kL-T_{\mathrm{x}}}, \ldots, x_{m,kL-1})^T \tag{6.99}$$
$$\overline{\mathbf{X}}_k = \left[\bar{\mathbf{X}}_{m,k}\right] = [\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{x}}_{m,k})] \tag{6.100}$$
$$\overline{\mathbf{U}}_k = \overline{\mathbf{W}}_k\,\overline{\mathbf{X}}_k \tag{6.101}$$
$$\tilde{\mathbf{u}}_{m,k} = \tilde{\mathbf{P}}_{-T_{\mathrm{u}},T_{\mathrm{u}}}\,\mathbf{F}^{-1}\,\mathrm{diag}\left(\bar{\mathbf{U}}_{m,k}\right) \tag{6.102}$$
$$= (u_{m,kL}, \ldots, u_{m,kL+T_{\mathrm{u}}}, 0, \ldots, 0, u_{m,kL-T_{\mathrm{u}}}, \ldots, u_{m,kL-1})^T \tag{6.103}$$

2. Adaptation error:

$$\tilde{\mathbf{y}}_{m,k} = g_m\,(\tilde{\mathbf{u}}_{m,k}) \tag{6.104}$$
$$= (y_{m,kL}, \ldots, y_{m,kL+T_{\mathrm{u}}}, 0, \ldots, 0, y_{m,kL-T_{\mathrm{u}}}, \ldots, y_{m,kL-1})^T \tag{6.105}$$
$$\overline{\mathbf{Y}}_k = \left[\bar{\mathbf{Y}}_{m,k}\right] = [\mathrm{diag}\,(\mathbf{F}\,\tilde{\mathbf{y}}_{m,k})] \tag{6.106}$$
$$\tilde{\mathbf{e}}_{\mathrm{b}_{m,k}} = \tilde{\mathbf{s}}_{m,k} - \tilde{\mathbf{u}}_{m,k} \tag{6.107}$$
$$\overline{\mathbf{E}}_{\mathrm{b}_k} = \left[\bar{\mathbf{E}}_{\mathrm{b}_{m,k}}\right] = \left[\mathrm{diag}\left(\mathbf{F}\,\tilde{\mathbf{e}}_{\mathrm{b}_{m,k}}\right)\right] \tag{6.108}$$

3. Update equations:

$$\overline{\mathbf{W}}'_{k+1} = \text{any update equation from Table 6.4 or Table E.14} \tag{6.109}$$
$$\bar{\mathbf{W}}_{ij,k+1} = \mathrm{diag}\left(\mathbf{P}_{\tilde{\mathbf{w}}}\ \mathrm{diag}\left(\bar{\mathbf{W}}'_{ij,k+1}\right)\right) \tag{6.110}$$

6.10 Blind decorrelation

In case we only wish to achieve decorrelated but not necessarily independent output signals, e.g., prewhitening of the input signals, we replace the nonlinearity $g(.)$ by a simple linearity. As a consequence we then have $y_{m,t}=u_{m,t}$. This substitution can be applied for all aforementioned blind algorithms: BSS, BD, and MCBD. Hence, we obtain either spatially uncorrelated output signals, and / or temporally uncorrelated output signals. For a detailed analysis of algorithms for blind decorrelation see [31].

6.11 Automatic gain control

The simplest blind algorithm is an automatic gain control, see also Fig. 1.5. If we deal with real valued signals and gains only, there is no need to use a nonlinearity for an AGC. Second-order statistics are sufficient here. However, in a complex-valued system, e.g., in the baseband representation of a communication system, the use of a nonlinearity can help not only to adjust the gain of the output signal, but also to control the phase of the output signal [8, 78]. The nonlinearity is split into a function of the real and imaginary part, and not only of the absolute value of u_t. The real and imaginary parts of the output signal can then be forced to be as independent as they can possibly get, resulting in the original constellation up to a rotation of a multiple of $\pi/2$. Algorithms for an AGC with phase control are given in Table 6.5 and Table E.12.

Algorithm	Update equations
Infomax	$w_{t+1} = w_t + \mu \left(w_t^{-*} - y_t x_t^* \right)$
Natural gradient	$w_{t+1} = w_t + \mu \left(1 - y_t u_t^* \right) w_t$
EASI	$w_{t+1} = w_t + \mu \left(1 - u_t u_t^* + u_t y_t^* - y_t u_t^* \right) w_t$

Table 6.5: Update equations for AGC.

6.12 Decomposition of the global system

6.12.1 BSS: Decomposition of the global-system matrix $\mathbf{G}$

We now wish to analyze the behavior of the interchannel interference of the global system. To this end, we decompose the global-system matrix $\mathbf{G} = \mathbf{WA}$ in different ways

$$\mathbf{G} = \mathbf{G}^{\mathrm{o}} + \tilde{\mathbf{G}} \tag{6.111}$$

$$= \mathbf{W}^{\mathrm{o}}\mathbf{A} + (\mathbf{W} - \mathbf{W}^{\mathrm{o}})\mathbf{A} = \mathbf{W}^{\mathrm{o}}\mathbf{A} + \tilde{\mathbf{W}}\mathbf{A} = (\mathbf{W}^{\mathrm{o}} + \tilde{\mathbf{W}})\mathbf{A} \tag{6.112}$$

$$= \mathbf{PD} + \tilde{\mathbf{P}}\,\mathbf{D} = (\mathbf{P} + \tilde{\mathbf{P}})\,\mathbf{D} \tag{6.113}$$

$$= \mathbf{PD} + \mathbf{P}\tilde{\mathbf{E}}\mathbf{D} = \mathbf{P}(\mathbf{I} + \tilde{\mathbf{E}})\mathbf{D} = \mathbf{PED} \tag{6.114}$$

where $\tilde{\mathbf{W}} = \mathbf{W} - \mathbf{W}^{\mathrm{o}}$, $\mathbf{W}^{\mathrm{o}}$ is a perfect separation matrix, $\mathbf{P}$ a permutation matrix, $\mathbf{D}$ a complex-valued diagonal matrix, $\mathbf{E}$ a matrix with unity in the main diagonal, and $\tilde{\mathbf{E}} = \mathbf{E} - \mathbf{I}$ a matrix with zeros in the main diagonal. The source signals propagate via $\mathbf{G}$ to the output $\mathbf{u}$, where $\mathbf{G}^{\mathrm{o}}$ and $\tilde{\mathbf{G}}$ describe the desired propagation and the interchannel interference, respectively. Since $\mathbf{G}^{\mathrm{o}} = \mathbf{PD}$, $\mathbf{G}^{\mathrm{o}}$ has to have full rank M_{s} and only one non-zero element per row and column. Furthermore, we constrain the M_{s} non-zero entries of $\mathbf{G}^{\mathrm{o}}$ to coincide with those of $\mathbf{G}$ with the same index. Hence, $\mathbf{G}^{\mathrm{o}} = \left[g_{ij}^{\mathrm{o}}\right]$ and $\tilde{\mathbf{G}} = [\tilde{g}_{ij}]$ can be written as

$$g_{ij}^{\mathrm{o}} = \begin{cases} g_{ij}\,, & j = \pi\,(i) \\ 0\,, & \text{otherwise} \end{cases} \tag{6.115}$$

and

$$\tilde{g}_{ij} = \begin{cases} 0\,, & j = \pi\,(i) \\ g_{ij}\,, & \text{otherwise} \end{cases} \tag{6.116}$$

where the permutation $\pi\,(i)$ of the set $\mathcal{M} = \{1, \ldots, M_{\mathrm{s}}\}$ is a one-to-one mapping of $\mathcal{M}$ onto itself [25].

Since we wish to minimize the interchannel interference at the output $\mathbf{u}$, the M_{s} non-zero entries in $\mathbf{G}^{\mathrm{o}}$ are chosen such as to minimize $\|\tilde{\mathbf{G}}\|_F$ for a given $\mathbf{G}$. Alternatively, the problem of finding $\mathbf{G}^{\mathrm{o}}$ which minimizes $\|\tilde{\mathbf{G}}\|_F$ can be reformulated as finding a permutation $\pi\,(i)$ out of the $M_{\mathrm{s}}!$ possible ones, which minimizes $\|\tilde{\mathbf{G}}\|_F$, i.e., if $\mathbf{G}$ happens to be diagonal dominant, then $\mathbf{G}^{\mathrm{o}}$ is a diagonal matrix containing the main diagonal of $\mathbf{G}$ as its own main diagonal,

i.e. $\pi(i) = i$. Furthermore, from (6.111) and (6.115) we have

$$\|\tilde{\mathbf{G}}\|_F^2 = \|\mathbf{G} - \mathbf{G}^{\mathrm{o}}\|_F^2 \tag{6.117}$$

$$= \sum_{\substack{i,j \\ i \neq \pi(i)}} |g_{ij}|^2 = \sum_{i,j} |g_{ij}|^2 - \sum_i |g_{i,\pi(i)}|^2$$

$$= \|\mathbf{G}\|_F^2 - \|\mathbf{G}^{\mathrm{o}}\|_F^2 . \tag{6.118}$$

This means that finding a permutation $\pi(i)$ that minimizes $\|\tilde{\mathbf{G}}\|_F$ is equivalent to finding a permutation $\pi(i)$ which maximizes $\|\mathbf{G}^{\mathrm{o}}\|_F$ under the given constraints (6.115) on the choice of $\mathbf{G}^{\mathrm{o}}$. Once $\mathbf{G}^{\mathrm{o}}$ and $\tilde{\mathbf{G}}$ are determined, the factorization in (6.114), $\mathbf{G}^{\mathrm{o}} = \mathbf{PD}$, is easily evaluated and $\tilde{\mathbf{E}} = \mathbf{P}^T \tilde{\mathbf{G}} \mathbf{D}^{-1}$.

In the case of perfect separation, $\tilde{\mathbf{G}}$ vanishes and therefore $\mathbf{G} = \mathbf{G}^{\mathrm{o}} = \mathbf{PD} = \mathbf{W}^{\mathrm{o}}\mathbf{A}$. The optimal solution for the separation matrix is then

$$\mathbf{W}^{\mathrm{o}} = \mathbf{PDA}^{-1} , \tag{6.119}$$

a row-wise scaled and permuted version of the inverse system. Since we are interested in waveform-preserving estimates of the source signals, scaling and permutation of the output signals does not affect an optimal solution with $\tilde{\mathbf{G}} = \mathbf{0}$. In the communications literature, such a solution would be described as zero-forcing [51]. A *zero-forcing* algorithm tries to force all elements of $\tilde{\mathbf{G}}$ to zero and therefore focuses on perfect separation rather than high output *signal-to-interference-plus-noise ratio* (SINR). The terminology of zero-forcing is more common in the field of blind deconvolution for the case where one merely wants to minimize intersymbol interference, regardless of any additive noise.

6.12.2 BD: Decomposition of the global-system filter $g(z)$

In a similar way to the decomposition of $\mathbf{G}$ in Section 6.12.1, we can decompose the global-system matrix $g(z) = w(z)a(z)$ in different ways

$$g(z) = g^{\mathrm{o}}(z) + \tilde{g}(z) \tag{6.120}$$

$$= w^{\mathrm{o}}(z)a(z) + (w(z) - w^{\mathrm{o}}(z))a(z) \tag{6.121}$$

$$= w^{\mathrm{o}}(z)a(z) + \tilde{w}(z)a(z) = (w^{\mathrm{o}}(z) + \tilde{w}(z))a(z) \tag{6.122}$$

$$= d(z) + \tilde{p}(z)\, d(z) = (1 + \tilde{p}(z))\, d(z) \tag{6.123}$$

$$= d(z) + \tilde{e}(z)d(z) = (1 + \tilde{e}(z))d(z) = e(z)d(z) \tag{6.124}$$

where $g^{\mathrm{o}}(z) = d(z) = d' z^{-\tau}$ consists of a single term. Note that in the single-channel case, we have no permutation indetermination of the output signals.

6.12.3 MCBD: Decomposition of the global-system matrix $\mathbf{G}(z)$

In a similar way to the decomposition of $\mathbf{G}$ in Section 6.12.1, we can decompose the global-system matrix $\mathbf{G}(z) = \mathbf{W}(z)\mathbf{A}(z)$ in different ways

$$\mathbf{G}(z) = \mathbf{G}^{\mathrm{o}}(z) + \tilde{\mathbf{G}}(z) \tag{6.125}$$

$$= \mathbf{W}^{\mathrm{o}}(z)\mathbf{A}(z) + (\mathbf{W}(z) - \mathbf{W}^{\mathrm{o}}(z))\mathbf{A}(z) \tag{6.126}$$

$$= \mathbf{W}^{\mathrm{o}}(z)\mathbf{A}(z) + \tilde{\mathbf{W}}(z)\mathbf{A}(z) = (\mathbf{W}^{\mathrm{o}}(z) + \tilde{\mathbf{W}}(z))\mathbf{A}(z) \tag{6.127}$$

$$= \mathbf{PD}(z) + \tilde{\mathbf{P}}(z)\, \mathbf{D}(z) = (\mathbf{P} + \tilde{\mathbf{P}}(z))\, \mathbf{D}(z) \tag{6.128}$$

$$= \mathbf{PD}(z) + \mathbf{P}\tilde{\mathbf{E}}(z)\mathbf{D}(z) = \mathbf{P}(\mathbf{I} + \tilde{\mathbf{E}}(z))\mathbf{D}(z) = \mathbf{PE}(z)\mathbf{D}(z) . \tag{6.129}$$

$\mathbf{P}$ is again a permutation matrix and $\mathbf{D}(z) = \mathrm{diag}\left[d'_1 z^{-\tau_1}, \ldots, d'_{M_{\mathrm{s}}} z^{-\tau_{M_{\mathrm{s}}}}\right]$. Furthermore, $\mathbf{G}^{\mathrm{o}}(z)$ has full rank and only one term per non-zero element.

6.13 Performance measures for blind identification

With the use of blind signal processing algorithms, we are more interested in wave-form preserving estimates of the signals, than in their exact amplitude or phase. We will therefore measure the performance of a blind algorithm on the residual *interchannel interference* (ICI) or the *intersymbol interference* (ISI). Alternatively, we could also use the *signal-to-interference ratio* (SIR), or in the noisy case the *signal-to-interference-plus-noise ratio* (SINR).

6.13.1 Blind source separation

Ideally, the global system matrix $\mathbf{G} = \mathbf{WA}$ in (6.111) would have one non-zero entry in each row i. $|g_{ij}|^2$ is the power transfer from source j to the output i. In each row i there will be one dominant entry, where $\max_j |g_{ij}|^2$ indicates the power transfer of the separated source j.

The other entries of the same row i of $\mathbf{G}$, which end up being non-zero in a realistic case, if squared and summed, reveal the power of other sources leaking through to the specific output i, if all sources have equal power and are mutually uncorrelated. Vice versa, by squaring the subdominant entries of the same column of $\mathbf{G}$, we get the power of one source leaking through to different outputs. Provided the same-source lock-on effect is not a problem for the algorithm under consideration [70], the row-wise observation of the permutation matrix is more meaningful than the column-wise observation, rendering the performance index into a measure of *interchannel interference* (ICI) [59]

$$J_{\text{ICI}}(\mathbf{G}) = \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{j=1}^{M}|g_{ij}|^2 - \max_j |g_{ij}|^2}{\max_j |g_{ij}|^2}\,. \tag{6.130}$$

Of course, $J_{\text{ICI}}(\mathbf{G})$ is available in a simulation environment only. In practical situations, the true matrix $\mathbf{A}$ and therefore the matrix $\mathbf{G}$ are unknown.

6.13.2 Single-channel blind deconvolution

In the BD case, the performance measure should reflect the deconvolution capability, hence indicating to what extent the deconvolved signal is influenced by adjacent samples of the same signal (convolutive noise). Recall that the global system response is defined as $g(z) = w(z)a(z) = \sum_{n=-\infty}^{\infty} g_n z^{-n}$. Similarly to (6.130), the *intersymbol interference* (ISI) can be defined as

$$J_{\text{ISI}}\left(g(z)\right) = \frac{\sum_{n=-\infty}^{\infty}|g_n|^2 - \max_n |g_n|^2}{\max_n |g_n|^2} = \frac{\sum_{n=-\infty}^{\infty}|g_n|^2}{\max_n |g_n|^2} - 1\,. \tag{6.131}$$

For finite impulse responses, the infinite sums in (6.131) are reduced to finite sums.

6.13.3 Multichannel blind deconvolution

For the more general MCBD case, we are interested in both measures (6.130) and (6.131). We denote $\mathbf{G}(z)$ as the matrix of filter polynomials in z of the global system response with entries $[\mathbf{G}(z)]_{ij} = g_{ij}(z) = \sum_{n=-\infty}^{\infty} g_{ij,n} z^{-n}$ being the global filter response between source j and output i. Then we can find the averaged ICI as

$$J_{\text{ICI}}(\mathbf{G}(z)) = \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{j=1}^{M}\sum_{n=-\infty}^{\infty}|g_{ij,n}|^2}{\max_j \sum_{n=-\infty}^{\infty}|g_{ij,n}|^2} - 1\,. \tag{6.132}$$

J_{ICI} is also a meaningful measure for algorithms which only separate the source signals but do not attempt to deconvolve them.
For the calculation of the averaged ISI, each row of the permutation matrix is searched for the entry containing the filter with the most energy. By calculating the ISI of these entries and averaging over all rows, we get

$$J_{\text{ISI}}(\mathbf{G}(z)) = \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{n=-\infty}^{\infty}|g_{i,(\arg\max_j \sum_n |g_{ij,n}|^2),\,n}|^2}{\max_n |g_{i,(\arg\max_j \sum_n |g_{ij,n}|^2),\,n}|^2} - 1. \tag{6.133}$$

An alternative measure reflecting both averaged ISI and ICI was defined in [70]

$$J_{\text{MC-ISI}}(\mathbf{G}(z)) = \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{j=1}^{M}\sum_{n=-\infty}^{\infty}|g_{ij,n}|^2}{\max_{j,\,n}|g_{ij,n}|^2} - 1. \tag{6.134}$$

If (6.132) and (6.133) are only calculated row-wise, rather than averaged over all rows, a simple relationship between them and (6.134) can be found

$$J_{\text{MC-ISI row}-i} = J_{\text{ICI row}-i} + J_{\text{ISI row}-i} + J_{\text{ICI row}-i}\cdot J_{\text{ISI row}-i}\,. \tag{6.135}$$

If $J_{\text{ICI row}-i}$ and $J_{\text{ISI row}-i}$ are small values (good separation and deconvolution), $J_{\text{MC-ISI row}-i}$ is roughly equal to the sum of these two measures. The corresponding results are obtained for a column-wise analysis.

6.14 Simulations

6.14.1 Blind source separation

The simulation setup is as follows: The mixing matrix has the condition number $\chi(\mathbf{A}) = 10$ and logarithmically distributed singular values. The $M_{\mathrm{s}} = 10$ source signals are Laplacian distributed with unity power. Sensor noise has $\sigma_{\mathrm{n}} = 0.01$ which equals -40 dB SNR. We used a block-wise update with block length $L = 100$. For the comparison we used the natural gradient, the BLMS2b, and the Infomax learning algorithm. The step sizes μ were 0.3, 0.3, and 0.7, respectively, and are chosen to achieve the fastest convergence behavior, without becoming unstable. The performance curves are shown in Table 6.3. Among the three algorithms, the Infomax algorithm has the slowest convergence rate. However, it is well known, that the Infomax has its best performance for unitary mixing matrices, i.e. $\chi(\mathbf{A}) = 0$.

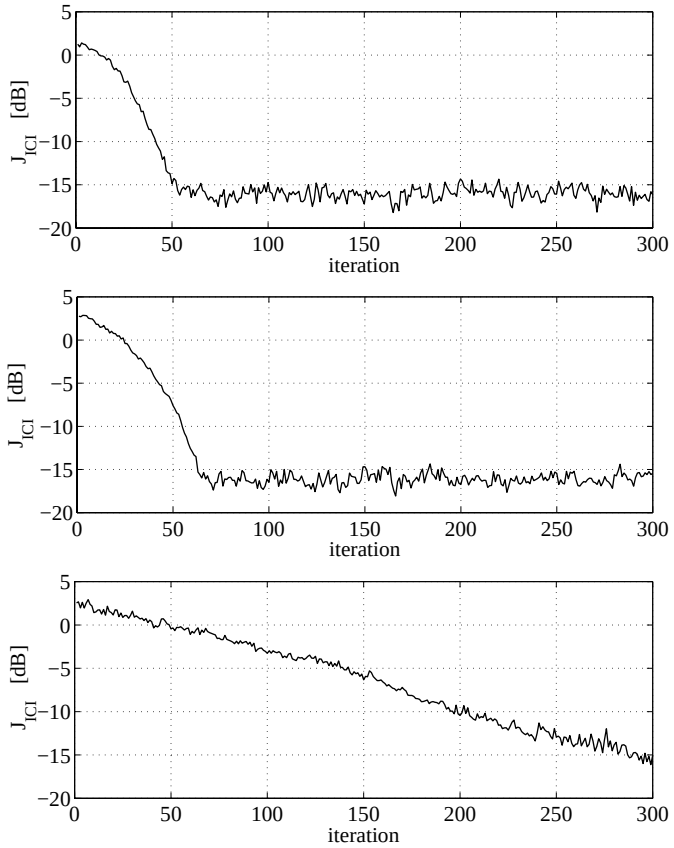

Figure 6.3: Performance curves of blind source separation: (from top) natural gradient, BLMS2b, and Infomax.

6.14.2 Blind deconvolution

The unknown system is the same as for the system identification example in Section 4.8.1 with $\|a(z)\|_{\mathcal{F}} = 1$. The simulation parameters are: The source signal is Laplacian distributed with $\sigma_{\mathrm{s}} = 1$, the sensor noise is white Gaussian distributed with $\sigma_{\mathrm{n}} = 0.01$ (-40 dB SNR). The non-causal deconvolution filter $w(z)$ has $N_{\mathrm{w}} = 300$, where the filter coefficients were initialized using a center-spike strategy, i.e. $w_0(z) = 1$. Further parameters are: $T_{\mathrm{x}} = 1000$, $N_{\mathrm{w}} = 300$, $T_{\mathrm{u}} = T_{\mathrm{x}} - N_{\mathrm{w}} = 700$, block size $L = 2T_{\mathrm{u}} + 1 = 1401$, and the FFT size $C = 2048$. The Bussgang nonlinearity is $g(u_t) = \sqrt{2}\,\mathrm{sign}(u_t)$. We use no prewhitening of the input signal $x(z)$.

The parameters of the algorithms are

BLMS1c	$\mu = 0.1$	
BLMS2a	$\mu = 0.1$	
BLMS2b	$\mu = 0.1$	
BLMS3	$\mu = 0.1$	almost no convergence
BLMS4	$\mu = 0.03$	almost no convergence (Infomax)
BRLS2	$\lambda = 0.8$	
BRLS1a	$\lambda = 0.8$	
BRLS1b	$\lambda = 0.8$	
BRLS1c	$\lambda = 0.8$	
Nat. grad.	$\mu = 0.05$	
EASI	$\mu = 0.05$	

The performance curves of J_{ISI} are shown in Fig. 6.4 to 6.6.

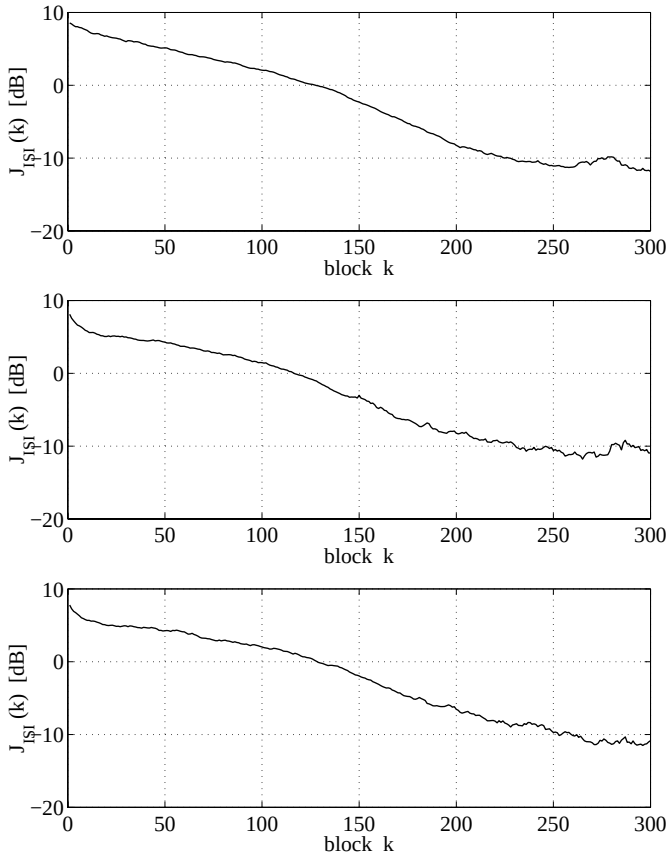

Figure 6.4: Performance curves of blind deconvolution: (from top) BLMS1c, BLMS2a, and BLMS2b.

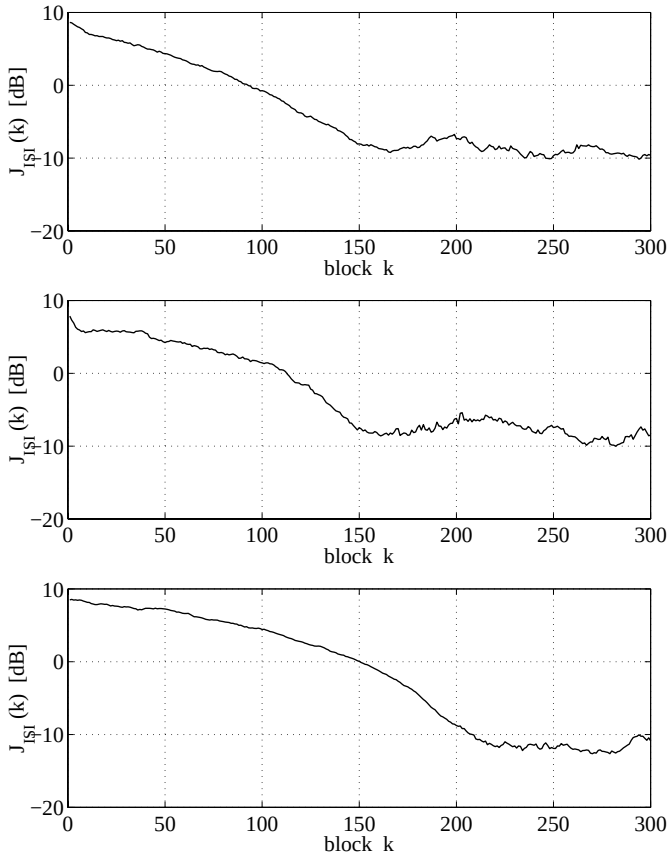

Figure 6.5: Performance curves of blind deconvolution: (from top) BRLS1a, BRLS1b, and BRLS2.

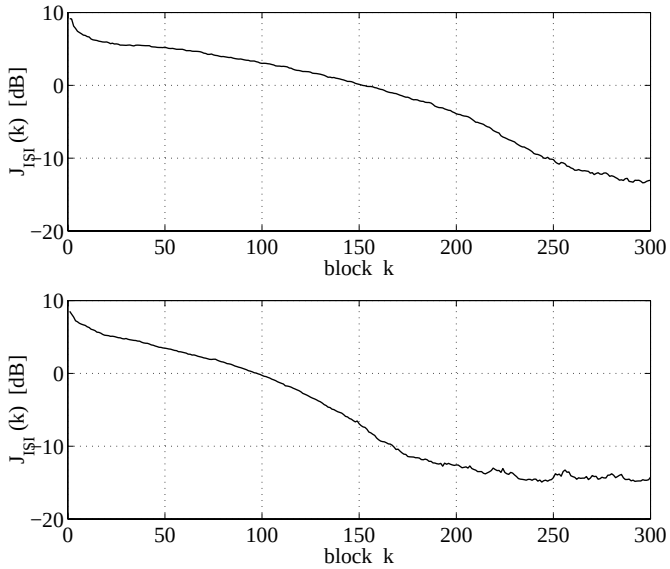

Figure 6.6: Performance curves of blind deconvolution: (top) Natural gradient, (bottom) EASI algorithm.

6.14.3 Multichannel blind deconvolution

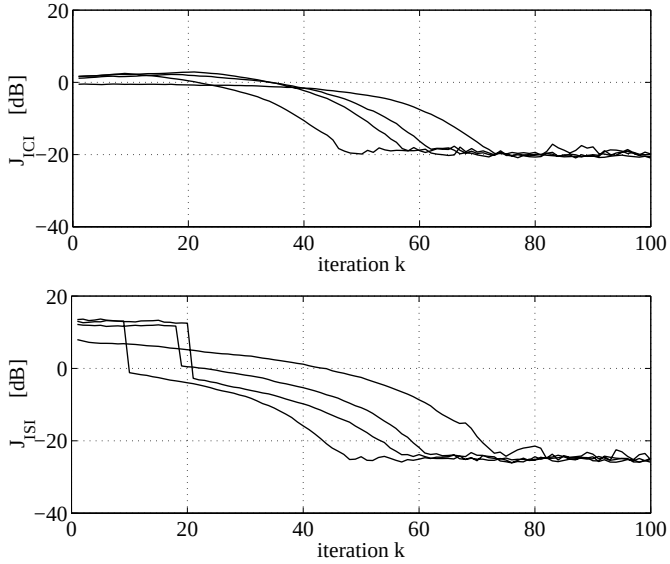

Figure 6.7: Channel-wise performance curves of BRLS2 in a multichannel blind deconvolution setup: (top) $J_{\text{ICI}}(k)$, (bottom) $J_{\text{ISI}}(k)$.

We use the same unknown convolutive mixing matrix as for the multichannel inverse modeling in Section 5.7.2. The simulation parameters are: $M_{\text{s}} = 4$ source signals which are Gamma distributed with $\sigma_{\text{s}} = 1$, $M = 4$ sensors with additive white Gaussian sensornoise with $\sigma_{\text{n}} = 0.01$, The non-causal separation matrix $\mathbf{W}(z)$ has $N_{\text{w}} = 100$ and was initially set to $\mathbf{W}_0(z) = \mathbf{I} + \mathbf{W}'_0(z)$, where the coefficients of $\mathbf{W}'(z)$ are small random numbers. This makes a total of $4 \cdot 4 \cdot 201 \approx 3200$ filter coefficients $w_{ij,n}$ to adapt. We use the BRLS2 algorithms given in Table E.14 with $T_{\text{x}} = 2000$, $T_{\text{u}} = 1900$, block length $L = 3801$ and FFT size $C = 4096$. The Bussgang nonlinearities are $g_m(u_t) = \sqrt{2}\,\text{sign}(u_{m,t})$. We use no prewhitening of the input signals $\mathbf{x}(z)$. The parameters of the algorithm are

BRLS2	$\lambda = 0.8$

The channel-wise performance curves are given in Fig. 6.7. J_{ICI} and J_{ISI} are defined in (6.132) and (6.133), respectively.

6.15 Summary

In this chapter we extended the non-blind algorithms of the previous chapters to work also in a blind environment, where the algorithm has no access to the source signals. Guidelines are given how to modify an algorithm when a non-blind error criterion is exchanged with a blind error criterion. In doing so, we have also shown an alternative derivation of the well-known natural-gradient learning algorithm.

Nonlinearity Simulations have shown that the separation capability of a blind algorithm is almost unaffected by small deviations of the nonlinearities $f(.)$ or $g(.)$ from their respective theoretical forms. In fact, the knowledge whether the pdf of a source signal is super-Gaussian (more peaky than a Gaussian pdf) or sub-Gaussian (flatter than a Gaussian pdf) is usually sufficient for a suitable choice of the nonlinearity.

Non-Gaussianity of the source signals The "closer" the sources are distributed to a Gaussian distribution, the "harder" the problem becomes, e.g., source signals which are Gamma distributed are easier to separate and deconvolve than signals which have a Laplacian distribution ($\kappa = 3$). One possible measure of the closeness to a Gaussian distributon is the kurtosis κ of a signal, defined in (6.9). In the extreme, when all source signals are Gaussian distributed ($\kappa = 0$), no separation or deconvolution is possible.

Simulations Simulation examples are presented, where many hundreds of filter coefficients are adapted, showing the performance behavior of some of the proposed algorithms. In fact, not all MCBD algorithms showed convergence. For some update equations, the difficulty of finding a stable equilibrium point is too high, due to the many coefficients which have to be adapted.

Same-source lock on One problem that sometimes comes up in blind source separation is that a source signal s_n appears at several outputs u_m. The so-called *same-source lock-on* problem can be detected by analyzing the row-wise *and* column-wise ICI of the global-system matrix $\mathbf{G}$ [70]. However, these measurements are available in a simulation environment only. In a real-world application, we have to monitor the determinant of $\mathbf{W}$. A small value of $\det \mathbf{W}$ is

an indication that a same-source lock-on problem is present. Some BSS algorithms are robust against the same-source lock-on problem, e.g. , the infomax algorithm, which contains $\mathbf{W}^{-1}$ explicitly in the update equation, the natural gradient algorithm, and also the EASI algorithm.

6.15.1 Further topics in blind identification

In the following, we mention some topics related to blind identification.

BD realization in the time domain

In many real-time applications in acoustics, a low processing latency is important, e.g. hands-free telephone. The block-wise processing of the data introduces a latency delay of $2L$ samples (2 blocks) for balanced processor load. One block is required for collecting the new samples and giving out the computed output samples, a second block delay is used for computing the filtering and adaptation. Thus, the overall group delay is $2L$ samples plus the group delay of the filter. To reduce the overall delay, one can use a small block size L or remain in the time domain. In fact, the algorithms proposed for BD and MCBD can also be realized in the time domain. The filtering and the adaptation are carried out at the sampling rate $L=1$ and not at the block rate. However, from simulation examples it seems that a block-wise processing of the data helps achieve a faster convergence of blind algorithms. An online learning algorithm for BD and MCBD which is based on the natural gradient and realized in the time-domain was given in [5, 6, 34]. See also the MATLAB example in [71].

Direct estimation of the mixing matrix

We can also transform an algorithm for blind source separation to directly estimate the mixing matrix $\mathbf{A}$ instead of the inverse mixing matrix $\mathbf{W}=\hat{\mathbf{A}}^{-1}$, see also [27]. As an example we take the natural-gradient learning algorithm shown in Table 6.2. Inverting both sides of the update equation and using the matrix-inversion lemma (A.5), with $\mathbf{A}'=\mathbf{W}_t \triangleq \mathbf{H}_t^{-1}$, $\mathbf{B}'=\mu\left(\mathbf{I}-\mathbf{y}_t\mathbf{u}_t^H\right)$, $\mathbf{C}'=\mathbf{I}$, and $\mathbf{D}'=\mathbf{W}_t$ gives

$$\mathbf{H}_{t+1}=\mathbf{H}_t-\mu_t\mathbf{H}_t\left(\mathbf{I}-\mathbf{y}_t\,\mathbf{u}_t^H\right)\left[\mathbf{I}+\mu_t\left(\mathbf{I}-\mathbf{y}_t\,\mathbf{u}_t^H\right)\right]^{-1}. \qquad (6.136)$$

Algorithm	Update equations
Infomax	$\mathbf{H}_{t+1}=\mathbf{H}_t-\mu\,\mathbf{H}_t\left(\mathbf{H}_t^{-H}-\mathbf{y}_t\mathbf{x}_t^H\right)\mathbf{H}_t$
Natural gradient	$\mathbf{H}_{t+1}=\mathbf{H}_t+\mu\,\mathbf{H}_t\left(\mathbf{I}-\mathbf{y}_t\mathbf{u}_t^H\right)$
EASI	$\mathbf{H}_{t+1}=\mathbf{H}_t+\mu\,\mathbf{H}_t\left(\mathbf{I}-\mathbf{u}_t\mathbf{u}_t^H+\mathbf{u}_t\mathbf{y}_t^H-\mathbf{y}_t\mathbf{u}_t^H\right)$

Table 6.6: Update equations for direct estimation of the mixing matrix where $\mathbf{H}=\hat{\mathbf{A}}$ and $\mathbf{x}_t=\mathbf{H}_t^{-1}\mathbf{u}_t$.

For small values of μ_t we can either expand the matrix inverse in (6.136) with (A.12) or replace it by the unity matrix in the latter case. We obtain

$$\mathbf{H}_{t+1}=\mathbf{H}_t-\mu_t\mathbf{H}_t\left(\mathbf{I}-\mathbf{y}_t\,\mathbf{u}_t^H\right) \qquad (6.137)$$

which is again a serial-update equation, but this time with multiplication from the right, i.e., $\mathbf{H}_{t+1}=\mathbf{H}_t\triangle\mathbf{H}_t$ with $\triangle\mathbf{H}_t=\mathbf{I}-\mu_t\left(\mathbf{I}-\mathbf{y}_t\,\mathbf{u}_t^H\right)$. Adapting $\mathbf{H}$ directly with (6.137) might be advantageous in the convolutive case. If the elements $a_{ij}(z)$ of the mixing matrix $\mathbf{A}(z)$ have only a few terms, then $\mathbf{H}(z)$ needs only a few terms for the adaptation as well. However, we still have to invert $\mathbf{H}(z)$ after every update step, because the output sequence is then $\mathbf{u}=\mathbf{H}^{-1}(z)\mathbf{x}(z)$.

Using the same steps we can also transform the other algorithms given in Table 6.2. The resulting adaptation equations are listed in Table 6.6.

Overdetermined blind source separation

Most algorithms for BSS and MCBD assume a fully determined system, i.e., the same number of sensors as source signals. We refer to the situation where more sensors than source signals are present as *overdetermined blind source separation* ($M > M_\mathrm{s}$). Basically there are two different approaches. One is to directly find an algorithm which copes with this situation. Another possibility is to first apply a preprocessing stage with M input and M_s output signals, and then build the input signals of a subsequent stage, e.g., an ordinary $M_\mathrm{s}\times M_\mathrm{s}$ BSS algorithm. Such a two-stage approach with a PCA (principle component

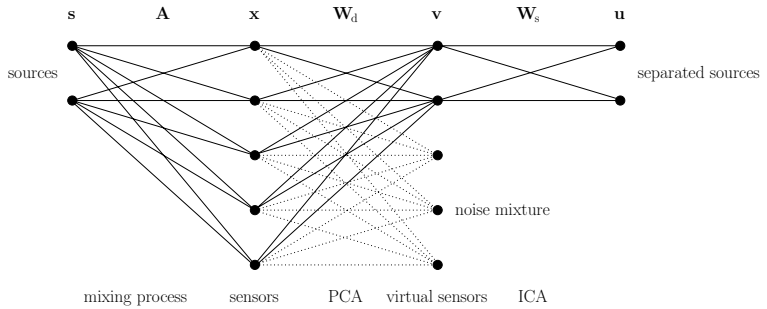

Figure 6.8: Two-stage approach for overdetermined blind source separation with $M_\mathrm{s} = 2$ sources and $M = 5$ sensors: First stage PCA, second stage ICA.

analysis) preprocessing stage is shown in Fig. 6.8, [60]. The PCA stage divides the M-dimensional input space into an M_s-dimensional signal-plus-noise space and an $(M - M_\mathrm{s})$-dimensional noise space. The first M_s *virtual sensors* are then used as the input signals of a subsequent ICA (independent component analysis) stage. Overdetermined blind source separation is also known as *undercomplete-bases problem*.

Underdetermined blind source separation

The case where fewer sensors than source signals are used ($M < M_\mathrm{s}$) is certainly one of the big challenges in the field of ICA. If the source signals have non-overlapping spectra, the task can easily be solved in the frequency domain. However, if they have overlapping spectra, then other methods have to be used. Algorithms for *underdetermined blind source separation* are described in [1, 73, 74]. Underdetermined blind source separation is also known as *overcomplete-bases problem*.

Mixture of sub- and super-Gaussian source signals

Most algorithms for BSS require prior knowledge, such as whether the source signals are sub- or super-Gaussian to select a proper nonlinearity in advance. If in the signal mixture the number of sub- and super-Gaussian source signals are known, often two different types of nonlinearities have to be used in the update equations. However, if the characteristics of the source signals is unknown, the blind algorithm has to estimate them online, e.g., by using parametric models of the source-signal pdfs. Algorithms which separate a mixture of sub- and super-Gaussian source signals are given in [32, 79].

Chapter 7

Concluding remarks

7.1 Conclusions

In acoustics, the transfer function between a source and a sensor is usually modeled by an FIR filter with hundreds or thousands of coefficients. This makes the proposed algorithms for the multichannel convolutive-mixing case to be suitable for acoustical applications, e.g., the combination of speaker separation and dereverberation of speech signals in a teleconferencing setup. However, in this work we have not discussed how to make the algorithms also operate successfully in a real environment. Many of the algorithms have been shown to perform well in a controlled simulation environment. In practice, several assumptions will certainly not be fulfilled and therefore influence the performance behavior of the algorithms, e.g., in stereophonic echo cancelling the source signals are strongly correlated. Thus, further research still has to be done to analyze the behavior with real-world signals.

7.2 Outlook and further directions

In this section, we list several topics related to blind and non-blind adaptive filtering that are subject of ongoing research:

7.2.1 Second-order statistics

Throughout this thesis we have assumed, that for blind algorithms the source signals are stationary but non-Gaussian. Because of the stationarity, second-order statistics are not capable of separating and deconvolving the source signals. However, second-order statistics can be sufficient for signal separation in the following cases:

- The source signals are nonstationary [63, 86–88, 107],
- the source signals are mutually independent but temporally correlated [10, 56, 80, 82],
- the source signals are cyclostationary [103, 104].

Whereas the first two situations appear in an acoustical environment, the last one is often true in data communications.

7.2.2 Filter partitioning

A further step towards reducing the computational burden of multichannel adaptive filtering can be to introduce filter-partitioning techniques [97, 99]. Similar to the overlap-save techniques, where the input sequence is partitioned into non-overlapping blocks, the filters can be partitioned as well. In doing so, the block size L can be reduced significantly without increasing the computational complexity very much. Moreover, a smaller FFT size can be chosen, which can even be smaller than the filter length. Filter partitioning is a technique often used in single-channel acoustical echo canceling [81]. Efficient methods for multichannel filter partitioning and cascading two or more systems in the partitioned frequency domain are given in [62].

7.2.3 Step-size control

Often adaptive algorithms reveal a poor performance behavior in a real environment, although they work well in simulation examples. The reason is that many underlying assumptions from theory are not fulfilled in reality, e.g., stationarity is probably not an adequate assumption for speech signals. In order to still obtain a satisfactory behavior of the adaptive algorithm, we have to perform online monitoring of certain signals for the proper adjustment of the step size [6]. For example, if we detect a change of the system, we increase the step-size to track the new situation. Or, if we know that the algorithm performs well, then we can reduce the step size. Note, that the Kalman filter has such a step-size control incorporated inherently in the update equations. Hence, knowing the current environment is of great value for controlling the adaptive algorithm.

7.2.4 Bootstrap

One of the big problems of blind algorithms is their relatively slow convergence rate, especially in the initial stage. Whereas non-blind adaptive algorithms reveal a steady decay of the performance curve from the beginning, blind algorithms tend to show a poor initial performance. It is like searching for a mouse hole on a soccer field: once you see the hole, you know in which direction you have to walk. Thus, if one can find a bootstrap technique which can accelerate the initial performance behavior, the overall convergence rate can be improved significantly.

A method for BD or MCBD which has shown to be useful in simulation examples is to use a time- or performance-dependent filter length $N_{\mathrm{w}}(t)$. Initially we start to adapt a short filter, then enlare the filter length steadily with time.

Another technique which we refer to as *data re-using*, is to carry out several update steps with the same data block. This is some sort of combination of a batch and a block-wise learning algorithm. The underlying idea comes from (6.41) which is in fact a nonlinear equation for the separation matrix $\mathbf{W}(z)$ and has to be solved iteratively.

Appendix A

General results

A.1 Differential entropy

The *differential entropy* of a probability-density function $p_U(\mathbf{u})$ is defined as

$$H(\,p_U(.)\,) \triangleq -\int p_U(\mathbf{u}) \log p_U(\mathbf{u})\, d\mathbf{u} \tag{A.1}$$

$$= -E\left\{\log p_U(\mathbf{u})\right\} \tag{A.2}$$

with $H(\,p_U(.)\,) \geq 0$.

A.2 Kullback-Leibler divergence

The *Kullback-Leibler divergence* or *relative entropy* between two probability-density functions $p_U(\mathbf{u})$ and $p_S(\mathbf{s})$ is defined as [23]

$$D\left(\,p_U(.) \,\|\, p_S(.)\,\right) \triangleq \int p_U(\mathbf{u}) \log \frac{p_U(\mathbf{u})}{p_S(\mathbf{u})}\, d\mathbf{u} \tag{A.3}$$

$$= E\left\{\log \frac{p_U(\mathbf{u})}{p_S(\mathbf{u})}\right\} \tag{A.4}$$

whenever the integral exists. Properties of $D\left(.\|.\right)$ are $D\left(\,p_U(.)\,\|\,p_S(.)\,\right) \geq 0$ with equality iff $p_U(.) = p_S(.)$ almost everywhere, and $D\left(\,p_U(.)\,\|\,p_S(.)\,\right) \neq D\left(\,p_S(.)\,\|\,p_U(.)\,\right)$.

A.3 Matrix-inversion lemma

If $\mathbf{A}'$ and $\mathbf{C}'$ are nonsingular $M \times M$ and $N \times N$ matrices, respectively, the following equality holds [50, 65, 75]:

$$\left[\mathbf{A}' + \mathbf{B}'\mathbf{C}'\mathbf{D}'\right]^{-1} = \mathbf{A}'^{-1} - \mathbf{A}'^{-1}\mathbf{B}'\left[\mathbf{C}'^{-1} + \mathbf{D}'\mathbf{A}'^{-1}\mathbf{B}'\right]^{-1}\mathbf{D}'\mathbf{A}'^{-1}. \tag{A.5}$$

Proof: Premultiply both sides of (A.5) with $\left[\mathbf{A}' + \mathbf{B}'\mathbf{C}'\mathbf{D}'\right]$. □

A.4 Inverse of a block matrix

If $\mathbf{A}^{-1}$ and $\mathbf{D}^{-1}$ exist, then we have [65]

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{C}\mathbf{A}^{-1} & \mathbf{D}^{-1} \end{bmatrix} \tag{A.6}$$

and

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{A}^{-1}\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{D}^{-1} \end{bmatrix}. \tag{A.7}$$

If $\mathbf{A}^{-1}$ exist, then we have

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{E}\boldsymbol{\Delta}^{-1}\mathbf{F} & -\mathbf{E}\boldsymbol{\Delta}^{-1} \\ -\boldsymbol{\Delta}^{-1}\mathbf{F} & \boldsymbol{\Delta}^{-1} \end{bmatrix} \tag{A.8}$$

with

$$\boldsymbol{\Delta} = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B} \tag{A.9}$$

$$\mathbf{E} = \mathbf{A}^{-1}\mathbf{B} \tag{A.10}$$

$$\mathbf{F} = \mathbf{C}\mathbf{A}^{-1}. \tag{A.11}$$

A.5 Matrix-inverse expansion

Under the assumption that the matrix $\mathcal{E}$ has a spectral norm smaller than 1 ($\sigma_1 < 1$), [51] the following matrix expansion holds

$$\left[\mathbf{I} - \mathcal{E}\right]^{-1} = \mathbf{I} + \sum_{k=1}^{\infty}\mathcal{E}^k = \sum_{k=0}^{\infty}\mathcal{E}^k \tag{A.12}$$

Proof: Premultiply both sides of (A.12) with $\left[\mathbf{I} - \mathcal{E}\right]$. □

A.6 Matrix inverse

The inverse of a nonsingular square matrix $\mathbf{A}$ can be derived using the following closed form expression [65]

$$\mathbf{A}^{-1} = \frac{\mathrm{adj}\,\mathbf{A}}{\det\mathbf{A}} \tag{A.13}$$

where $\mathrm{adj}\,\mathbf{A}$ is the *adjoint* or *adjugate* of $\mathbf{A}$ and $\det\mathbf{A}$ is the determinant of $\mathbf{A}$. The elements of the adjoint matrix are called *cofactors*

$$\left[\mathrm{adj}\,\mathbf{A}\right]_{mn} = (-1)^{m+n}\det\left(\mathbf{A}_{(nm)}\right). \tag{A.14}$$

The submatrix $\mathbf{A}_{(nm)}$ is obtained by deleting the nth row and the mth column of $\mathbf{A}$. Note, $\det(\mathbf{A}_{(nm)}) = \det((\mathbf{A}^T)_{(mn)})$ is called the nmth *minor* of $\mathbf{A}$.

Inverse of a polynomial matrix Likewise to ordinary matrices, the inverse of a polynomial matrix can be written as

$$\mathbf{A}^{-1}(z) = \frac{\mathrm{adj}\,\mathbf{A}(z)}{\det\mathbf{A}(z)} \tag{A.15}$$

with

$$\left[\mathrm{adj}\,\mathbf{A}(z)\right]_{mn} = (-1)^{m+n}\det\left(\mathbf{A}_{(nm)}(z)\right). \tag{A.16}$$

A.7 SVD — Singular Value Decomposition

SVD of a matrix One of the most powerful tools from Linear Algebra is the *singular value decomposition* (SVD) of a matrix. A matrix $\mathbf{A}^{m\times n}$ can be written as

$$\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^H = \mathbf{U}\begin{bmatrix} \widetilde{\mathbf{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\mathbf{V}^H \tag{A.17}$$

where $\mathbf{U}^{m\times m}$ and $\mathbf{V}^{n\times n}$ are unitary matrices, $\mathbf{\Sigma}^{m\times n}$ is a rectangular diagonal matrix, and $\widetilde{\mathbf{\Sigma}}^{k\times k}$ is a diagonal matrix of full rank. The diagonal elements of $\mathbf{\Sigma}$ and $\widetilde{\mathbf{\Sigma}}$ are the ordered singular values $\sigma_1 \geq \ldots \geq \sigma_k > \sigma_{k+1} = \ldots = \sigma_n = 0$ if $m \geq n \geq k$ and k is the *rank* of $\mathbf{A}$. The SVD is very helpful in analyzing transformations from one space to another, whereas the eigenvalue decomposition (EVD) is more useful in analyzing a transformation from a space to itself, e.g. , state-space model of a dynamic system.

SVD of a polynomial matrix The idea of a SVD of a scalar matrix can be extended to polynomial matrices.

$$\mathbf{A}(z) = \mathbf{U}(z)\mathbf{\Sigma}(z)\mathbf{V}^H(z) = \mathbf{U}(z)\begin{bmatrix} \widetilde{\mathbf{\Sigma}}(z) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\mathbf{V}^H(z) \tag{A.18}$$

where $\mathbf{U}(z)^{m\times m}$ and $\mathbf{V}(z)^{n\times n}$ are *paraunitary matrices*, $\mathbf{\Sigma}(z)^{m\times n}$ is a rectangular diagonal matrix, and $\widetilde{\mathbf{\Sigma}}(z)^{k\times k}$ is a diagonal matrix whose elements are nonzero. A paraunitary matrix has the property that

$$\mathbf{U}(z)\mathbf{U}^H(z) = \mathbf{U}^H(z)\mathbf{U}(z) = \mathbf{I}\,. \tag{A.19}$$

As a consequence, $\mathbf{U}^{-1}(z) = \mathbf{U}^H(z) = \mathbf{U}_*^T(z^{-1})$. In other words, the matrix $\mathbf{U}$ is transposed, the elements $u_{ij,k}$ are complex conjugated and the polynomials $u_{ij}(z)$ are time reversed, i.e., z is replaced by z^{-1}. Paraunitary matrices are the multidimensional extension of allpass filters.

The filters $\sigma_i(z)$ on the diagonal of $\mathbf{\Sigma}$ are symmetric, linear-phase filters, $\sigma_i(z) = \sigma_i(z^{-1})$. To make the matrix $\mathbf{\Sigma}(z)$ unique, a metric has to be specified, sorting the diagonal elements of $\mathbf{\Sigma}(z)$, e.g. by using $\|\sigma_i(z)\|_{\mathcal{F}}$.

A.8 Pseudoinverse

Pseudoinverse of a matrix Let the SVD of $\mathbf{A}$ be as in (A.17). The *Moore-Penrose pseudoinverse* of $\mathbf{A}$ is defined as

$$\mathbf{A}^{\#} \triangleq \mathbf{V}\,\mathbf{\Sigma}^{\#}\,\mathbf{U}^H \tag{A.20}$$

where $\mathbf{\Sigma}^{\#}$ is the transpose of $\mathbf{\Sigma}$ in which the positive singular values of $\mathbf{A}$ are replaced by their reciprocals. Note, that $\mathbf{A}^{-1} = \mathbf{V}\,\mathbf{\Sigma}^{-1}\,\mathbf{U}^H$.

Pseudoinverse of a polynomial matrix Let the SVD of $\mathbf{A}(z)$ be as in (A.18). We define the pseudoinverse of a polynomial matrix $\mathbf{A}(z)$ as

$$\mathbf{A}^{\#}(z) \triangleq \mathbf{V}(z)\,\mathbf{\Sigma}^{\#}(z)\,\mathbf{U}^H(z) \tag{A.21}$$

where $\mathbf{\Sigma}^{\#}(z)$ is the transpose of $\mathbf{\Sigma}$ in which the nonzero diagonal elements $\sigma_i(z)$ of $\mathbf{\Sigma}(z)$ are replaced by their inverse $\sigma_i^{-1}(z)$. The regular inverse is defined as $\mathbf{A}^{-1}(z) = \mathbf{V}(z)\,\mathbf{\Sigma}^{-1}(z)\,\mathbf{U}^H(z)$.

Appendix B

The trace function

B.1 Definitions

The Frobenius norm and the trace of a matrix are denoted by $\|.\|_F$ and $\mathrm{tr}\,(.)$, respectively. $\mathbf{a} = \mathrm{diag}\,(\mathbf{A})$ is a vector whose elements are the diagonal elements of $\mathbf{A}$ and $\mathrm{diag}\,(\mathbf{a})$ is a square diagonal matrix which contains the elements of $\mathbf{a}$. $\mathrm{ddiag}(\,\mathbf{A}\,)$ zeros the off-diagonal elements of $\mathbf{A}$ and

$$\mathrm{off}\,(\,\mathbf{A}\,) \triangleq \mathbf{A} - \mathrm{ddiag}(\,\mathbf{A}\,) \tag{B.1}$$

zeros the diagonal elements of $\mathbf{A}$. For a square matrix $\mathbf{A}$ we have $\mathrm{ddiag}(\,\mathbf{A}\,) = \mathrm{diag}\,(\mathrm{diag}\,(\mathbf{A}))$.

B.2 Basic properties of the trace function

Basic properties of the trace function:

$$\mathrm{tr}(c\mathbf{A}) = c\,\mathrm{tr}(\mathbf{A}) \tag{B.2}$$
$$\mathrm{tr}(\mathbf{A}^T) = \mathrm{tr}(\mathbf{A}) \tag{B.3}$$
$$\mathrm{tr}(\mathbf{A}^*) = (\mathrm{tr}(\mathbf{A}))^* \tag{B.4}$$
$$\mathrm{tr}(\mathbf{A}^H) = (\mathrm{tr}(\mathbf{A}))^* \tag{B.5}$$
$$\mathrm{tr}(\mathbf{A}) = \sum_m \lambda_m(\mathbf{A}) \tag{B.6}$$
$$\mathrm{tr}(\mathbf{A}^k) = \sum_m \lambda_m^k(\mathbf{A}) \tag{B.7}$$
$$\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA}) \tag{B.8}$$
$$\mathrm{tr}(\mathbf{ABC}) = \mathrm{tr}(\mathbf{CAB}) = \mathrm{tr}(\mathbf{BCA}) \tag{B.9}$$
$$\mathrm{tr}(\mathbf{A}+\mathbf{B}) = \mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B}) \tag{B.10}$$

Frobenius norm and trace function:

$$\|\mathbf{A}\|_F^2 \triangleq \mathrm{tr}\left(\mathbf{A}^H\mathbf{A}\right) \tag{B.11}$$
$$\|\mathrm{off}(\,\mathbf{A}\,)\|_F^2 = \|\mathbf{A}\|_F^2 - \|\mathrm{ddiag}(\,\mathbf{A}\,)\|_F^2 \tag{B.12}$$
$$\mathrm{tr}\,(\mathbf{A}\,\mathrm{ddiag}(\,\mathbf{B}\,)) = \mathrm{tr}\,(\mathrm{ddiag}(\,\mathbf{A}\,)\,\mathbf{B}) \tag{B.13}$$
$$= \mathrm{tr}\,(\mathrm{ddiag}(\,\mathbf{A}\,)\,\mathrm{ddiag}(\,\mathbf{B}\,)) \tag{B.14}$$
$$\mathrm{tr}\,(\mathrm{off}(\,\mathbf{A}\,)\,\mathrm{ddiag}(\,\mathbf{B}\,)) = 0\,. \tag{B.15}$$

B.3 Basic properties of the determinant

Determinant of a square matrix ($\mathbf{A}^{M\times M}, \mathbf{B}^{M\times M}$):

$$\det(\mathbf{A}) = \prod_m \lambda_m(\mathbf{A}) \tag{B.16}$$
$$\det(\mathbf{A}^k) = \prod_m \lambda_m^k(\mathbf{A}) \tag{B.17}$$
$$\det(\mathbf{A}^T) = \det(\mathbf{A}) \tag{B.18}$$
$$\det(\mathbf{A}^*) = (\det(\mathbf{A}))^* \tag{B.19}$$
$$\det(c\,\mathbf{A}) = c^M \det(\mathbf{A}) \tag{B.20}$$
$$\det(\mathbf{A}^k) = (\det(\mathbf{A}))^k \tag{B.21}$$
$$\det(\mathbf{AB}) = \det(\mathbf{BA}) = \det(\mathbf{A})\det(\mathbf{B})\,. \tag{B.22}$$

B.4 Derivatives with respect to a real matrix

The partial derivative of a real scalar function $J(\mathbf{X})$ with respect to a real matrix $\mathbf{X}$ is defined as

$$\left[\frac{\partial}{\partial \mathbf{X}} J(\mathbf{X})\right]_{mn} \triangleq \frac{\partial J(\mathbf{X})}{\partial x_{mn}}. \tag{B.23}$$

We have the following partial derivatives, see also [47]

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}) = \mathbf{I} \tag{B.24}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{AXB}) = \mathbf{A}^T\mathbf{B}^T \tag{B.25}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{AX}^T\mathbf{B}) = \mathbf{BA} \tag{B.26}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{XX}) = 2\mathbf{X}^T \tag{B.27}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^T\mathbf{X}) = 2\mathbf{X} \tag{B.28}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^k) = k\,(\mathbf{X}^T)^{k-1} \tag{B.29}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{XAXB}) = \mathbf{A}^T\mathbf{X}^T\mathbf{B}^T + \mathbf{B}^T\mathbf{X}^T\mathbf{A}^T \tag{B.30}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^T\mathbf{AXB}) = \mathbf{AXB} + \mathbf{A}^T\mathbf{XB}^T \tag{B.31}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^T\mathbf{AX}^T\mathbf{B}) = \mathbf{AX}^T\mathbf{B} + \mathbf{BX}^T\mathbf{A} \tag{B.32}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^{-1}) = -\left(\mathbf{X}^{-1}\mathbf{X}^{-1}\right)^T \tag{B.33}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{AX}^{-1}\mathbf{B}) = -\left(\mathbf{X}^{-1}\mathbf{BAX}^{-1}\right)^T \tag{B.34}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(e^{\mathbf{X}}) = e^{\mathbf{X}^T} \tag{B.35}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(e^{\mathbf{AXB}}) = \left(\mathbf{B}\,e^{\mathbf{AXB}}\mathbf{A}\right)^T \tag{B.36}$$

$$\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{X}) = \operatorname{adj}(\mathbf{X}^T) = \det(\mathbf{X})\,\mathbf{X}^{-T} \tag{B.37}$$

$$\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{AXB}) = \det(\mathbf{AXB})\,\mathbf{X}^{-T} \tag{B.38}$$

$$\frac{\partial}{\partial \mathbf{X}} \log(|\det(\mathbf{X})|) = \frac{1}{\det(\mathbf{X})}\frac{\partial}{\partial \mathbf{X}}\det(\mathbf{X}) = \mathbf{X}^{-T} \tag{B.39}$$

We can expand a real function $J(\mathbf{X})$ around $\mathbf{X}$ as

$$J(\mathbf{X} + d\mathbf{X}) = J(\mathbf{X}) + \langle \tfrac{\partial}{\partial \mathbf{X}} J(\mathbf{X}), d\mathbf{X} \rangle + \mathcal{O}\left(\|d\mathbf{X}\|_F^2\right) \tag{B.40}$$

with $\langle \mathbf{A},\mathbf{B} \rangle \triangleq \operatorname{tr}\left(\mathbf{AB}^T\right)$. For a real-valued matrix $\mathbf{X}$, the gradient is defined as

$$\nabla_{\mathbf{X}} J(\mathbf{X}) \triangleq \frac{\partial}{\partial \mathbf{X}} J(\mathbf{X}). \tag{B.41}$$

B.5 Derivatives with respect to a complex matrix[1]

After Brandwood [14], we can define the partial derivatives of a scalar function $J(\mathbf{X})$ with respect to a complex matrix $\mathbf{X}$ as

$$\left[\frac{\partial}{\partial \mathbf{X}} J(\mathbf{X})\right]_{mn} \triangleq \frac{1}{2}\left(\frac{\partial J(\mathbf{X})}{\partial x_{mn}^{\mathrm{re}}} - j\frac{\partial J(\mathbf{X})}{\partial x_{mn}^{\mathrm{im}}}\right) \tag{B.42}$$

$$\left[\frac{\partial}{\partial \mathbf{X}^*} J(\mathbf{X})\right]_{mn} \triangleq \frac{1}{2}\left(\frac{\partial J(\mathbf{X})}{\partial x_{mn}^{\mathrm{re}}} + j\frac{\partial J(\mathbf{X})}{\partial x_{mn}^{\mathrm{im}}}\right) \tag{B.43}$$

where $\mathbf{X} = [x_{mn}]$ with $x_{mn} \triangleq x_{mn}^{\mathrm{re}} + j\,x_{mn}^{\mathrm{im}}$.

We can expand a complex-valued function $J(\mathbf{X})$ around $\mathbf{X}$ as

$$J(\mathbf{X} + d\mathbf{X}) = J(\mathbf{X}) + \langle \tfrac{\partial}{\partial \mathbf{X}} J(\mathbf{X}), d\mathbf{X}^* \rangle + \langle \tfrac{\partial}{\partial \mathbf{X}^*} J(\mathbf{X}), d\mathbf{X} \rangle + \mathcal{O}\left(\|d\mathbf{X}\|_F^2\right) \tag{B.44}$$

with $\langle \mathbf{A},\mathbf{B} \rangle \triangleq \operatorname{tr}\left(\mathbf{AB}^H\right)$. After Haykin [51], the *complex gradient matrix* can be defined as

$$\nabla_{\mathbf{X}} J(\mathbf{X}) \triangleq 2\frac{\partial}{\partial \mathbf{X}^*} J(\mathbf{X}). \tag{B.45}$$

See also [102] for a nice treatment of this subject.

[1] Personal notes from Heinz Mathis

We have the following partial derivatives with respect to $\mathbf{X}$:

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}) = \mathbf{I} \tag{B.46}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^H) = \mathbf{0} \tag{B.47}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{A}^T\mathbf{B}^T \tag{B.48}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{A}\mathbf{X}^T\mathbf{B}) = \mathbf{B}\mathbf{A} \tag{B.49}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{A}\mathbf{X}^*\mathbf{B}) = \mathbf{0} \tag{B.50}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{A}\mathbf{X}^H\mathbf{B}) = \mathbf{0} \tag{B.51}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}\mathbf{X}) = 2\mathbf{X}^T \tag{B.52}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}\mathbf{X}^T) = 2\mathbf{X} \tag{B.53}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}\mathbf{X}^*) = \mathbf{X}^H \tag{B.54}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}\mathbf{X}^H) = \mathbf{X}^* \tag{B.55}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^k) = k\ \left(\mathbf{X}^T\right)^{k-1} \tag{B.56}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\left(\mathbf{X}^H\right)^k) = \mathbf{0} \tag{B.57}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{A}^T\mathbf{X}^T\mathbf{B}^T + \mathbf{B}^T\mathbf{X}^T\mathbf{A}^T \tag{B.58}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^H\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{A}^T\mathbf{X}^*\mathbf{B}^T \tag{B.59}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^H\mathbf{A}\mathbf{X}^H\mathbf{B}) = \mathbf{0} \tag{B.60}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{X}^{-1}) = -\left(\mathbf{X}^{-1}\mathbf{X}^{-1}\right)^T \tag{B.61}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(\mathbf{A}\mathbf{X}^{-1}\mathbf{B}) = -\left(\mathbf{X}^{-1}\mathbf{B}\mathbf{A}\mathbf{X}^{-1}\right)^T \tag{B.62}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(e^{\mathbf{X}}) = e^{\mathbf{X}^T} \tag{B.63}$$

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{tr}(e^{\mathbf{A}\mathbf{X}\mathbf{B}}) = \left(\mathbf{B}\,e^{\mathbf{A}\mathbf{X}\mathbf{B}}\mathbf{A}\right)^T \tag{B.64}$$

$$\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{X}) = \operatorname{adj}(\mathbf{X}^T) = \det(\mathbf{X})\,\mathbf{X}^{-T} \tag{B.65}$$

$$\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{A}\mathbf{X}\mathbf{B}) = \det(\mathbf{A}\mathbf{X}\mathbf{B})\,\mathbf{X}^{-T} \tag{B.66}$$

We have the following partial derivatives with respect to $\mathbf{X}^*$:

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}) = \mathbf{0} \tag{B.67}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}^H) = \mathbf{I} \tag{B.68}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{0} \tag{B.69}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{A}\mathbf{X}^T\mathbf{B}) = \mathbf{0} \tag{B.70}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{A}\mathbf{X}^*\mathbf{B}) = \mathbf{A}^T\mathbf{B}^T \tag{B.71}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{A}\mathbf{X}^H\mathbf{B}) = \mathbf{B}\mathbf{A} \tag{B.72}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}\mathbf{X}) = \mathbf{0} \tag{B.73}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}\mathbf{X}^T) = \mathbf{0} \tag{B.74}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}\mathbf{X}^*) = \mathbf{X}^T \tag{B.75}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}\mathbf{X}^H) = \mathbf{X} \tag{B.76}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}^k) = \mathbf{0} \tag{B.77}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\left(\mathbf{X}^H\right)^k) = k\ \left(\mathbf{X}^H\right)^{k-1} \tag{B.78}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{0} \tag{B.79}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}^H\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{A}\mathbf{X}\mathbf{B} \tag{B.80}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}^H\mathbf{A}\mathbf{X}^H\mathbf{B}) = \mathbf{A}\mathbf{X}^H\mathbf{B} + \mathbf{B}\mathbf{X}^H\mathbf{A} \tag{B.81}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{X}^{-1}) = \mathbf{0} \tag{B.82}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(\mathbf{A}\mathbf{X}^{-1}\mathbf{B}) = \mathbf{0} \tag{B.83}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(e^{\mathbf{X}}) = \mathbf{0} \tag{B.84}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \operatorname{tr}(e^{\mathbf{A}\mathbf{X}\mathbf{B}}) = \mathbf{0} \tag{B.85}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \det(\mathbf{X}) = \mathbf{0} \tag{B.86}$$

$$\frac{\partial}{\partial \mathbf{X}^*} \det(\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{0} \tag{B.87}$$

Appendix C

Norms

C.1 Frobenius norm

Let $\mathbb{M}$ be the inner product space of complex matrixes. Given two matrices $\mathbf{A}$ and $\mathbf{B}$ with $\mathbf{A}, \mathbf{B} \in \mathbb{M}$, we define the scalar product of two matrices as

$$\begin{aligned} \langle \mathbf{A}, \mathbf{B} \rangle &\triangleq \mathrm{tr}\left\{ \mathbf{A}\mathbf{B}^H \right\} && \text{(C.1)} \\ &= \sum_m \left[\mathbf{A}\mathbf{B}^H \right]_{mm} \\ &= \sum_m \sum_n a_{mn}\, b^*_{mn}\,. && \text{(C.2)} \end{aligned}$$

The induced norm is equivalent to the *Frobenius norm*, i.e.

$$\begin{aligned} \|\mathbf{A}\|_F &\triangleq \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle} \\ &= \sqrt{\sum_m \sum_n |a_{mn}|^2}\,. && \text{(C.3)} \end{aligned}$$

C.2 Inner product space of polynomial matrices

Let $\mathbb{P}$ be the inner product space of complex polynomial matrixes. Let $\mathbf{A}(z) = \sum_t \mathbf{A}_t z^{-t}$ and $\mathbf{B}(z)$ be two matrix polynomials or Laurent series.

Finite energy If $\mathbf{A}(z)$ or $\mathbf{B}(z)$ have *finite energy*, e.g., $\sum_t \|\mathbf{A}_t\|_F^2 < \infty$ or $\sum_t \|\mathbf{B}_t\|_F^2 < \infty$, we define the following *inner product*

$$\begin{aligned}\langle \mathbf{A}(z), \mathbf{B}(z) \rangle_{\mathcal{F}} &\triangleq \sum_t \langle \mathbf{A}_t, \mathbf{B}_t \rangle && \text{(C.4)}\\ &= \sum_t \operatorname{tr}\left\{ \mathbf{A}_t \mathbf{B}_t^H \right\} \\ &= \sum_t \sum_m \left[\mathbf{A}_t \mathbf{B}_t^H \right]_{mm} \\ &= \sum_t \sum_m \sum_n a_{mn,t}\, b^*_{mn,t} && \text{(C.5)}\\ &= \operatorname{tr}\left\{ \mathcal{P}_{0,0}\left(\mathbf{A}(z)\, \mathbf{B}^H(z) \right) \right\}. && \text{(C.6)}\end{aligned}$$

Finite power If $\mathbf{A}(z)$ and $\mathbf{B}(z)$ have *finite power*, e.g., $\lim_{T\to\infty} \frac{1}{2T+1} \sum_{t=-T}^{T} \|\mathbf{A}_t\|_F^2 < \infty$, we define the inner product as

$$\begin{aligned}\langle \mathbf{A}(z), \mathbf{B}(z) \rangle_{\mathcal{F}} &\triangleq \lim_{T\to\infty} \frac{1}{2T+1} \sum_{t=-T}^{T} \langle \mathbf{A}_t, \mathbf{B}_t \rangle && \text{(C.7)}\\ &= \lim_{T\to\infty} \frac{1}{2T+1} \sum_{t=-T}^{T} \sum_m \sum_n a_{mn,t}\, b^*_{mn,t} && \text{(C.8)}\\ &= \lim_{T\to\infty} \frac{1}{2T+1} \operatorname{tr}\left\{ \mathcal{P}_{0,0}\left(\mathbf{A}(z)\, \mathbf{B}^H(z) \right) \right\} && \text{(C.9)}\end{aligned}$$

for the *deterministic case* and as

$$\begin{aligned}\langle \mathbf{A}(z), \mathbf{B}(z) \rangle_{\mathcal{F}} &\triangleq E\left\{ \langle \mathbf{A}_t, \mathbf{B}_t \rangle \right\} && \text{(C.10)}\\ &= \sum_m \sum_n E\left\{ a_{mn,t}\, b^*_{mn,t} \right\} && \text{(C.11)}\end{aligned}$$

for the *stochastic case*.

Properties of inner products An inner product $\langle ., . \rangle$ has to fulfill the following properties [68]:

$$\begin{aligned}\langle \mathbf{A}(z) + \mathbf{B}(z), \mathbf{C}(z) \rangle_{\mathcal{F}} &= \langle \mathbf{A}(z), \mathbf{C}(z) \rangle_{\mathcal{F}} + \langle \mathbf{B}(z), \mathbf{C}(z) \rangle_{\mathcal{F}} && \text{(C.12)}\\ \langle \alpha \mathbf{A}(z), \mathbf{B}(z) \rangle_{\mathcal{F}} &= \alpha \langle \mathbf{A}(z), \mathbf{B}(z) \rangle_{\mathcal{F}} && \text{(C.13)}\\ \langle \mathbf{A}(z), \alpha \mathbf{B}(z) \rangle_{\mathcal{F}} &= \alpha^* \langle \mathbf{A}(z), \mathbf{B}(z) \rangle_{\mathcal{F}} && \text{(C.14)}\\ \langle \mathbf{A}(z), \mathbf{B}(z) \rangle_{\mathcal{F}} &= \langle \mathbf{B}(z), \mathbf{A}(z) \rangle^*_{\mathcal{F}} && \text{(C.15)}\\ \langle \mathbf{A}(z), \mathbf{A}(z) \rangle_{\mathcal{F}} &\geq 0 && \text{(C.16)}\\ \langle \mathbf{A}(z), \mathbf{A}(z) \rangle_{\mathcal{F}} = 0 &\iff \mathbf{A}(z) = \mathbf{0}. && \text{(C.17)}\end{aligned}$$

The proofs that $\langle ., . \rangle_{\mathcal{F}}$ fulfills these properties are easily obtained by using the definitions in (C.5), (C.8), and (C.11).

C.3 Norm space of polynomial matrices

The inner product $\langle ., . \rangle_{\mathcal{F}}$ defines an induced norm on $\mathbb{P}$ given by

$$\|\mathbf{A}(z)\|_{\mathcal{F}} \triangleq \sqrt{\langle \mathbf{A}(z), \mathbf{A}(z) \rangle_{\mathcal{F}}} \qquad \text{(C.18)}$$

and a *metric* on $\mathbb{P}$ induced by the norm

$$d\left(\mathbf{A}(z), \mathbf{B}(z) \right) \triangleq \|\mathbf{A}(z) - \mathbf{B}(z)\|_{\mathcal{F}}. \qquad \text{(C.19)}$$

Finite energy If $\mathbf{A}(z)$ has *finite energy*, the induced norm (C.18) becomes with (C.4)

$$\|\mathbf{A}(z)\|_{\mathcal{F}} = \sqrt{\sum_t \|\mathbf{A}_t\|_F^2} = \sqrt{\sum_t \sum_m \sum_n |a_{mn,t}|^2}. \qquad \text{(C.20)}$$

Finite power If $\mathbf{A}(z)$ has *finite power*, the induced norm (C.18) becomes with (C.7)

$$\|\mathbf{A}(z)\|_{\mathcal{F}} = \lim_{T\to\infty} \sqrt{\frac{1}{2T+1} \sum_{t=-T}^{T} \|\mathbf{A}_t\|_F^2} \qquad \text{(C.21)}$$

for the deterministic case and with (C.10)

$$\|\mathbf{A}(z)\|_{\mathcal{F}} = \sqrt{E\{\,\|\mathbf{A}_t\|_F^2\}} \tag{C.22}$$

for the stochastic case.

Properties of norms A norm $\|.\|$ has the following properties [54, 68]:

$$\|\mathbf{A}(z)\|_{\mathcal{F}} \geq 0 \tag{C.23}$$
$$\|\mathbf{A}(z)\|_{\mathcal{F}} = 0 \quad \Longleftrightarrow \quad \mathbf{A}(z) = \mathbf{0} \tag{C.24}$$
$$\|\alpha\mathbf{A}(z)\|_{\mathcal{F}} = |\alpha|\,\|\mathbf{A}(z)\|_{\mathcal{F}} \tag{C.25}$$
$$\|\mathbf{A}(z) + \mathbf{B}(z)\|_{\mathcal{F}} \leq \|\mathbf{A}(z)\|_{\mathcal{F}} + \|\mathbf{B}(z)\|_{\mathcal{F}} \tag{C.26}$$

where (C.26) is the *triangle inequality* where the equality sign holds if $\mathbf{B}(z) = \mathbf{0}$ or $\mathbf{A}(z) = c\mathbf{B}(z)$. Furthermore, an inner product and the corresponding norm satisfy the *Schwarz inequality*

$$|\,\langle \mathbf{A}(z), \mathbf{B}(z)\rangle_{\mathcal{F}}\,| \leq \|\mathbf{A}(z)\|_{\mathcal{F}}\,\|\mathbf{B}(z)\|_{\mathcal{F}} \tag{C.27}$$

with equality iff $\{\mathbf{A}(z), \mathbf{B}(z)\}$ is a linearly dependent set. Comparing (C.20) with (C.3), we see that $\|\mathbf{A}(z)\|_{\mathcal{F}}$ with $\mathbf{A}(z) \in \mathbb{P}$ is a natural extension of the Frobenius norm $\|\mathbf{A}\|_F$ for $\mathbf{A} \in \mathbb{M}$. We therefore refer to $\|.\|_{\mathcal{F}}$ as the *Frobenius norm for polynomial matrices*.

Appendix D

Projection operators

D.1 Generalized remainder $< . >_{a,b}$

In this section we define the *generalized remainder* $< . >_{a,b}$ and the *symmetric remainder* $< . >_C$. We will write $< t >_{0,C-1}$ to denote the standard remainder when t is divided by the non-zero integer C. The range of $< t >_{0,C-1}$ is $\{0, \cdots, C-1\}$. We introduce the following definitions:

$$< t >_{0,C-1} \triangleq t - \lfloor t/C \rfloor \cdot C = t \mod C \tag{D.1}$$
$$< t >_{a,b} \triangleq a + < t - a >_{0,b-a+1} \tag{D.2}$$
$$= a + (t - a \mod b - a + 1) \tag{D.3}$$
$$< t >_C \triangleq < t >_{-\lfloor C/2 \rfloor, \lfloor C/2 \rfloor} \tag{D.4}$$

With $< t >_{b,a}$ we denote the generalized remainder when t is divided by $b - a + 1$ where a, b are integers and $b > a$. For integer values of t, the domain of the remainder $< . >_{a,b}$ is $\{a, \cdots, b\}$. Finally, with $< t >_C$ we denote the symmetric remainder (C odd) when t is divided by C. The domain of $< t >_C$ is $\{-\lfloor C/2 \rfloor, \cdots, \lfloor C/2 \rfloor\}$. Fig. D.1 illustrates examples of the generalized remainder with different parameter sets. All of them have 7 elements in their domain.

Generalized and symmetric generalized remainders have the following properties:

$$< t >_C = - < -t >_C \tag{D.5}$$
$$< t + kC >_C = < t >_C \tag{D.6}$$
$$< t + kC >_{0,C-1} = < t >_{0,C-1} \tag{D.7}$$
$$< t + k(b-a+1) >_{a,b} = < t >_{a,b} \tag{D.8}$$

Eq.(D.5) means that $< t >_C$ is an odd function, i.e., symmetric w.r.t. the origin.

D.2 Polynomial projection operators

D.2.1 Polynomial projection operator $\mathcal{P}$

Let $x(z) = \sum_{-\infty}^{\infty} x_t z^{-t}$ be the double-sided z-transform (Laurent series) of the sequence x. We define the *polynomial projection operator* $\mathcal{P}$ as follows:

$$\mathcal{P}_{0,C-1}\left(x(z)\right) \triangleq \sum_{t=0}^{C-1} x_t\, z^{-t} = x_0 + \cdots + x_{C-1} z^{-C+1} \tag{D.9}$$

$$\mathcal{P}_{a,b}\left(x(z)\right) \triangleq \begin{cases} \sum_{t=a}^{b} x_t\, z^{-t} = x_a z^{-a} + \cdots + x_b z^{-b} & a \le b \\ 0 & a > b \end{cases} . \tag{D.10}$$

$\mathcal{P}_{a,b}\left(x(z)\right)$ with $b \ge a$ returns the sub-polynomial of $x(z)$ which contains the terms with the powers from z^{-a} to z^{-b}. Furthermore, we define the *symmetric polynomial projection operator*

$$\mathcal{P}_C\left(x(z)\right) \triangleq \mathcal{P}_{-\lfloor C/2 \rfloor, \lfloor C/2 \rfloor}\left(x(z)\right) = \sum_{t=-\lfloor C/2 \rfloor}^{\lfloor C/2 \rfloor} x_t\, z^{-t} \tag{D.11}$$

$$= x_{-\lfloor C/2 \rfloor} z^{+\lfloor C/2 \rfloor} + \cdots + x_{\lfloor C/2 \rfloor} z^{-\lfloor C/2 \rfloor} . \tag{D.12}$$

with $C \ge 0$ and odd. $\mathcal{P}_C\left(x(z)\right)$ returns the sub-polynomial of $x(z)$ which contains the terms with the powers from $z^{+\lfloor C/2 \rfloor}$ to $z^{-\lfloor C/2 \rfloor}$.

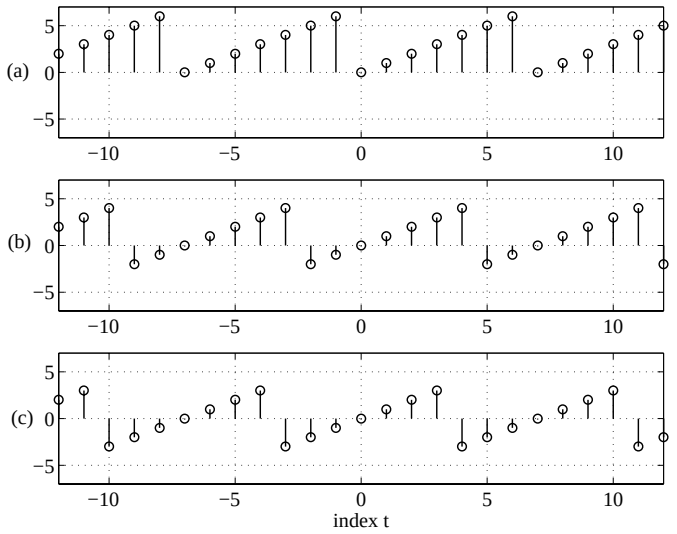

Figure D.1: Generalized remainder function:
(a) $< t >_{0,6}$: $t \longmapsto \{0, \cdots, 6\}$,
(b) $< t >_{-2,4}$: $t \longmapsto \{-2, \cdots, 4\}$,
(c) $< t >_{-3,3} = < t >_7$: $t \longmapsto \{-3, \cdots, 3\}$.

Properties of $\mathcal{P}$

Basic properties (α = complex value)

$$\mathcal{P}_{t,t}\left(x(z)\right) = x_t\, z^{-t} \tag{D.13}$$
$$\mathcal{P}_{a,b}\left(x(z)\right) = \sum\nolimits_{t=a}^{b} \mathcal{P}_{t,t}\left(x(z)\right) \tag{D.14}$$
$$\mathcal{P}_{a,b}\left(\alpha\, x(z)\right) = \alpha\, \mathcal{P}_{a,b}\left(x(z)\right) \tag{D.15}$$
$$\mathcal{P}^*_{a,b}\left(\alpha\, x(z)\right) = \alpha^*\, \mathcal{P}^*_{a,b}\left(x(z)\right) \tag{D.16}$$
$$\mathcal{P}_{a,b}\left(x(z) + y(z)\right) = \mathcal{P}_{a,b}\left(x(z)\right) + \mathcal{P}_{a,b}\left(y(z)\right) \tag{D.17}$$
$$\mathcal{P}_{a,b}\left(\sum\nolimits_k x_k(z)\right) = \sum\nolimits_k \mathcal{P}_{a,b}\left(x_k(z)\right) \tag{D.18}$$
$$\mathcal{P}_{a,c}\left(x(z)\right) = \mathcal{P}_{a,b}\left(x(z)\right) + \mathcal{P}_{b+1,c}\left(x(z)\right) \quad (a \le b < c) \tag{D.19}$$
$$\mathcal{P}_{a,b}\left(\mathcal{P}_{a,b}\left(x(z)\right)\right) = \mathcal{P}_{a,b}\left(x(z)\right) \tag{D.20}$$
$$\mathcal{P}_{a,b}\left(\mathcal{P}_{c,d}\left(x(z)\right)\right) = \mathcal{P}_{c,d}\left(\mathcal{P}_{a,b}\left(x(z)\right)\right) \tag{D.21}$$
$$= \mathcal{P}_{\max(a,c),\min(b,d)}\left(x(z)\right) . \tag{D.22}$$

Properties of time reversal, transposition, and complex conjugation

$$\mathcal{P}^*_{a,b}\left(x(z)\right) = \mathcal{P}_{-b,-a}\left(x^*(z)\right) \tag{D.23}$$
$$\mathcal{P}_{a,b}\left(x^*(z)\right) = \mathcal{P}_{a,b}\left(x_*(z^{-1})\right) = \mathcal{P}^*_{-b,-a}\left(x(z)\right) \tag{D.24}$$
$$\mathcal{P}_{a,b}\left(x(z^{-1})\right) = \mathcal{P}^*_{-b,-a}\left(x_*(z)\right) \tag{D.25}$$
$$\mathcal{P}_{a,b}\left(x_*(z)\right) = \mathcal{P}^*_{-b,-a}\left(x(z^{-1})\right) \tag{D.26}$$
$$\mathcal{P}_{C}\left(x^*(z)\right) = \mathcal{P}^*_{C}\left(x(z)\right) \tag{D.27}$$
$$\mathcal{P}_{C}\left(x(z^{-1})\right) = \mathcal{P}^*_{C}\left(x_*(z)\right) \tag{D.28}$$
$$\mathcal{P}_{a,b}\left(\mathbf{X}^T(z)\right) = \mathcal{P}^T_{a,b}\left(\mathbf{X}(z)\right) \tag{D.29}$$
$$\mathcal{P}_{a,b}\left(\mathbf{X}^H(z)\right) = \mathcal{P}_{a,b}\left(\mathbf{X}^T_*(z^{-1})\right) = \mathcal{P}^H_{-b,-a}\left(\mathbf{X}(z)\right) \tag{D.30}$$
$$\mathcal{P}^H_{a,b}\left(\mathbf{X}(z)\right) = \mathcal{P}_{-b,-a}\left(\mathbf{X}^H(z)\right)\,. \tag{D.31}$$

Properties of time shifting

$$\mathcal{P}_{t,t}\left(x(z)\right) = z^{-t}\,\mathcal{P}_{0,0}\left(z^t x(z)\right) \tag{D.32}$$
$$\mathcal{P}_{a,b}\left(x(z)\right) = z^{-a}\,\mathcal{P}_{0,b-a}\left(z^a x(z)\right) \tag{D.33}$$
$$\mathcal{P}_{a,b}\left(x(z)\right) = z^{d}\,\mathcal{P}_{a+d,b+d}\left(z^{-d}\,x(z)\right) \tag{D.34}$$
$$z^{-d}\,\mathcal{P}_{a,b}\left(x(z)\right) = \mathcal{P}_{a+d,b+d}\left(z^{-d}\,x(z)\right) \tag{D.35}$$
$$\mathcal{P}_{a,b}\left(z^{d}\,x(z)\right) = z^{d}\,\mathcal{P}_{a+d,b+d}\left(x(z)\right) \tag{D.36}$$
$$z^{-d}\,\mathcal{P}_{a,b}\left(z^{d}\,x(z)\right) = \mathcal{P}_{a+d,b+d}\left(x(z)\right) \tag{D.37}$$
$$\mathcal{P}_{t,t}\left(y(z)\right)\,\mathcal{P}_{a,b}\left(x(z)\right) = \mathcal{P}_{a+t,b+t}\left(\mathcal{P}_{t,t}\left(y(z)\right)\,x(z)\right) \tag{D.38}$$
$$\mathcal{P}^*_{t,t}\left(y(z)\right)\,\mathcal{P}_{a,b}\left(x(z)\right) = \mathcal{P}_{a-t,b-t}\left(\mathcal{P}^*_{t,t}\left(y(z)\right)\,x(z)\right)\,. \tag{D.39}$$

Properties of the inverse ($x(z)$ has at least two terms)

$$\mathcal{P}_{a,b}\left(x(z)\right)\mathcal{P}_{a,b}\left(x^{-1}(z)\right) \neq 1 \tag{D.40}$$
$$x(z)\,\mathcal{P}_{a,b}\left(x^{-1}(z)\right) \neq 1\,. \tag{D.41}$$

Most of these properties can easily be proved by applying (D.14) and (D.13). The task is then simplified to prove the properties with $a = b$.

We have some more involved properties ($a \leq b$, $c \leq d$)

$$\sum\nolimits_{t=a}^{b} \mathcal{P}_{t,t}\left(x(z)\right)\,\mathcal{P}^*_{t,t}\left(y(z)\right) = \sum\nolimits_{t=a}^{b} x_t\,y^*_t \tag{D.42}$$
$$= \mathcal{P}_{0,0}\left(\mathcal{P}_{a,b}\left(x(z)\right)y^*(z)\right) \tag{D.43}$$
$$= \mathcal{P}_{0,0}\left(x(z)\,\mathcal{P}^*_{a,b}\left(y(z)\right)\right) \tag{D.44}$$
$$= \mathcal{P}_{0,0}\left(\mathcal{P}_{a,b}\left(x(z)\right)\,\mathcal{P}^*_{a,b}\left(y(z)\right)\right) \tag{D.45}$$
$$\sum\nolimits_{t=a}^{b} \mathcal{P}_{t,t}\left(x(z)\right)\,\mathcal{P}^*_{t-d,t-d}\left(y(z)\right) = \sum\nolimits_{t=a}^{b} z^{-d}\,x_t\,y^*_{t-d} \tag{D.46}$$
$$= \mathcal{P}_{d,d}\left(\mathcal{P}_{a,b}\left(x(z)\right)y^*(z)\right) \tag{D.47}$$
$$= \mathcal{P}_{d,d}\left(x(z)\,\mathcal{P}^*_{a-d,b-d}\left(y(z)\right)\right) \tag{D.48}$$
$$= \mathcal{P}_{d,d}\left(\mathcal{P}_{a,b}\left(x(z)\right)\,\mathcal{P}^*_{a-d,b-d}\left(y(z)\right)\right) \tag{D.49}$$
$$\sum\nolimits_{t=a}^{b} \mathcal{P}_{t+d,t+d}\left(x(z)\right)\,\mathcal{P}^*_{t,t}\left(y(z)\right) = \sum\nolimits_{t=a}^{b} z^{-d}\,x_{t+d}\,y^*_t \tag{D.50}$$
$$= \mathcal{P}_{d,d}\left(\mathcal{P}_{a+d,b+d}\left(x(z)\right)y^*(z)\right) \tag{D.51}$$
$$= \mathcal{P}_{d,d}\left(x(z)\,\mathcal{P}^*_{a,b}\left(y(z)\right)\right) \tag{D.52}$$
$$= \mathcal{P}_{d,d}\left(\mathcal{P}_{a+d,b+d}\left(x(z)\right)\,\mathcal{P}^*_{a,b}\left(y(z)\right)\right)\,. \tag{D.53}$$

The more general case ($c \geq d$)

$$\sum\nolimits_{t=a}^{b} \mathcal{P}_{t,t}\left(x(z)\right)\,\mathcal{P}^*_{t-d,t-c}\left(y(z)\right) = \mathcal{P}_{c,d}\left(\mathcal{P}_{a,b}\left(x(z)\right)\,y^*(z)\right) \tag{D.54}$$
$$\sum\nolimits_{t=a}^{b} \mathcal{P}_{t+c,t+d}\left(x(z)\right)\,\mathcal{P}^*_{t,t}\left(y(z)\right) = \mathcal{P}_{c,d}\left(x(z)\,\mathcal{P}^*_{a,b}\left(y(z)\right)\right)\,. \tag{D.55}$$

Proof of (D.54): We start with the left side of (D.54) and apply (D.23), (D.38), (D.18), and (D.14)

$$\sum\nolimits_{t=a}^{b} \mathcal{P}_{t,t}\left(x(z)\right)\,\mathcal{P}^*_{t-d,t-c}\left(y(z)\right) = \sum\nolimits_{t=a}^{b} \mathcal{P}_{t,t}\left(x(z)\right)\mathcal{P}_{-t+c,-t+d}\left(y^*(z)\right)$$
$$= \sum\nolimits_{t=a}^{b} \mathcal{P}_{c,d}\left(\mathcal{P}_{t,t}\left(x(z)\right)\,y^*(z)\right)$$
$$= \mathcal{P}_{c,d}\left(\sum\nolimits_{t=a}^{b} \mathcal{P}_{t,t}\left(x(z)\right)\,y^*(z)\right)$$
$$= \mathcal{P}_{c,d}\left(\mathcal{P}_{a,b}\left(x(z)\right)\,y^*(z)\right)\,. \qquad \square$$

Alternative representation of $x(z)$

Alternatively, we can write $x(z) = \sum_{t=-\infty}^{\infty} x_t z^{-t}$ as

$$x(z) = \sum_{k=-\infty}^{\infty} \sum_{t=kC}^{kC+C-1} x_t \, z^{-t} \tag{D.56}$$

$$= \sum_{k=-\infty}^{\infty} \sum_{t=0}^{C-1} x_{t+kC} \, z^{-t-kC} \tag{D.57}$$

$$= \sum_{k=-\infty}^{\infty} z^{-kC} \sum_{t=0}^{C-1} x_{t+kC} \, z^{-t} = \sum_{k=-\infty}^{\infty} z^{-kC} x_k(z) \tag{D.58}$$

$$= \sum_{t=0}^{C-1} z^{-t} \sum_{k=-\infty}^{\infty} x_{t+kC} \, z^{-kC} = \sum_{t=0}^{C-1} z^{-t} \, x_t^{(p)}\left(z^C\right) . \tag{D.59}$$

Eq. (D.59) is called the *polyphase representation of* $x(z)$ [38] with $x_t^{(p)}(z) \triangleq \sum_{k=-\infty}^{\infty} x_{t+kC} \, z^{-k}$, whereas (D.58) represents $x(z)$ partitioned into consecutive non-overlapping blocks of length C. Using the definition in (D.10), we can reformulate (D.56) and (D.58), and obtain the following identities

$$x(z) = \sum_{k=-\infty}^{\infty} \mathcal{P}_{kC,kC+C-1}(x(z)) \tag{D.60}$$

$$= \sum_{k=-\infty}^{\infty} z^{-kC} \, \mathcal{P}_{0,C-1}\left(z^{kC} \, x(z)\right) = \sum_{k=-\infty}^{\infty} z^{-kC} \, x_k(z) \tag{D.61}$$

$$x_k(z) = z^{kC} \, \mathcal{P}_{kC,kC+C-1}(x(z)) = \mathcal{P}_{0,C-1}\left(z^{kC} \, x(z)\right) . \tag{D.62}$$

By doing similar calculations we obtain the general identities $(C = b - a + 1)$

$$x(z) = \sum_{k=-\infty}^{\infty} \mathcal{P}_{a+kC,b+kC}(x(z)) \tag{D.63}$$

$$= \sum_{k=-\infty}^{\infty} z^{-kC} \, \mathcal{P}_{a,b}\left(z^{kC} \, x(z)\right) = \sum_{k=-\infty}^{\infty} z^{-kC} \, x_k(z) \tag{D.64}$$

$$x_k(z) = z^{kC} \, \mathcal{P}_{a+kC,b+kC}(x(z)) = \mathcal{P}_{a,b}\left(z^{kC} \, x(z)\right) \tag{D.65}$$

and by using the symmetric polynomial projection operator

$$x(z) = \sum_{k=-\infty}^{\infty} z^{-kC} \, \mathcal{P}_C\left(z^{kC} \, x(z)\right) = \sum_{k=-\infty}^{\infty} z^{-kC} \, x_k(z) \tag{D.66}$$

$$x_k(z) = \mathcal{P}_C\left(z^{kC} \, x(z)\right) . \tag{D.67}$$

D.2.2 Circular polynomial projection operator $\tilde{\mathcal{P}}$

Let $x(z) = \sum_{t=-\infty}^{\infty} x_t z^{-t}$. We define the *circular polynomial projection operator* $\tilde{\mathcal{P}}$ as

$$\tilde{\mathcal{P}}_{0,C-1}(x(z)) = \sum_{t=-\infty}^{\infty} x_t \, z^{-<t>_{0,C-1}} \tag{D.68}$$

$$\tilde{\mathcal{P}}_{a,b}(x(z)) = \sum_{t=-\infty}^{\infty} x_t \, z^{-<t>_{a,b}} \tag{D.69}$$

$$\tilde{\mathcal{P}}_C(x(z)) = \sum_{t=-\infty}^{\infty} x_t \, z^{-<t>_C} \tag{D.70}$$

where $< t >_{a,b}$ denotes the generalized remainder defined in (D.2). $\tilde{\mathcal{P}}_C(.)$ is the *symmetric circular polynomial projection operator*. Similar to $\mathcal{P}_C(.)$, we require C to be odd and $C \geq 1$ for $\tilde{\mathcal{P}}_C(.)$.

Alternatively, we can write (D.68) as

$$\tilde{\mathcal{P}}_{0,C-1}(x(z)) = x(z) \mod z^{-C} - 1 \tag{D.71}$$

where $x(z) \mod z^{-C} - 1$ denotes the remainder of the division of $x(z)$ by $z^{-C} - 1$ over the field of polynomials [13].

We have the following useful relations ($C = b - a + 1$)

$$\tilde{\mathcal{P}}_{0,C-1}\left(x(z)\right) \triangleq \sum_{k=-\infty}^{\infty} \mathcal{P}_{0,C-1}\left(z^{kC}\, x(z)\right) \tag{D.72}$$

$$= \sum_{k=-\infty}^{\infty} z^{kC}\, \mathcal{P}_{kC,kC+C-1}\left(x(z)\right) \tag{D.73}$$

$$= \sum_{k=-\infty}^{\infty} \sum_{t=0}^{C-1} x_{t+kC}\, z^{-t} = \sum_{t=0}^{C-1} z^{-t} \sum_{k=-\infty}^{\infty} x_{t+kC} \tag{D.74}$$

$$= \tilde{x}_0 + \cdots + \tilde{x}_{C-1} z^{-C+1} \tag{D.75}$$

$$\tilde{\mathcal{P}}_{a,b}\left(x(z)\right) \triangleq \sum_{k=-\infty}^{\infty} \mathcal{P}_{a,b}\left(z^{kC}\, x(z)\right) \tag{D.76}$$

$$= \sum_{k=-\infty}^{\infty} z^{kC}\, \mathcal{P}_{a+kC,b+kC}\left(x(z)\right) \tag{D.77}$$

$$= \sum_{k=-\infty}^{\infty} \sum_{t=a}^{b} x_{t+kC}\, z^{-t} \tag{D.78}$$

$$= \tilde{x}_a z^{-a} + \cdots + \tilde{x}_b z^{-b} \tag{D.79}$$

$$\tilde{\mathcal{P}}_{C}\left(x(z)\right) \triangleq \sum_{k=-\infty}^{\infty} \mathcal{P}_{C}\left(z^{kC}\, x(z)\right) \tag{D.80}$$

$$= \sum_{k=-\infty}^{\infty} z^{kC}\, \mathcal{P}_{-\lfloor C/2 \rfloor + kC, \lfloor C/2 \rfloor + kC}\left(x(z)\right) \tag{D.81}$$

$$= \sum_{k=-\infty}^{\infty} \sum_{t=-\lfloor C/2 \rfloor}^{\lfloor C/2 \rfloor} x_{t+kL}\, z^{-t} \tag{D.82}$$

$$= \tilde{x}_{-\lfloor C/2 \rfloor} z^{+\lfloor C/2 \rfloor} + \cdots + \tilde{x}_{\lfloor C/2 \rfloor} z^{-\lfloor C/2 \rfloor} \triangleq \tilde{x}(z)\,. \tag{D.83}$$

Properties of $\tilde{\mathcal{P}}$

Basic properties (α = complex value)

$$\tilde{\mathcal{P}}_{0,0}\left(x(z)\right) = \sum_{k=-\infty}^{\infty} x_k \tag{D.84}$$

$$\tilde{\mathcal{P}}_{t,t}\left(x(z)\right) = z^{-t}\, \tilde{\mathcal{P}}_{0,0}\left(x(z)\right) \tag{D.85}$$

$$\tilde{\mathcal{P}}_{a,b}\left(\alpha\, x(z)\right) = \alpha\, \tilde{\mathcal{P}}_{a,b}\left(x(z)\right) \tag{D.86}$$

$$\tilde{\mathcal{P}}^{*}_{a,b}\left(\alpha\, x(z)\right) = \alpha^{*}\, \tilde{\mathcal{P}}^{*}_{a,b}\left(x(z)\right) \tag{D.87}$$

$$\tilde{\mathcal{P}}_{a,b}\left(x(z) + y(z)\right) = \tilde{\mathcal{P}}_{a,b}\left(x(z)\right) + \tilde{\mathcal{P}}_{a,b}\left(y(z)\right) \tag{D.88}$$

$$\tilde{\mathcal{P}}_{a,b}\left(\sum\nolimits_k x_k(z)\right) = \sum\nolimits_k \tilde{\mathcal{P}}_{a,b}\left(x_k(z)\right) \tag{D.89}$$

$$\tilde{\mathcal{P}}_{a,b}\left(\tilde{\mathcal{P}}_{a,b}\left(x(z)\right)\right) = \tilde{\mathcal{P}}_{a,b}\left(x(z)\right)\,. \tag{D.90}$$

Properties of complex conjugation, time reversal, and complex conjugation

$$\tilde{\mathcal{P}}^{*}_{a,b}\left(x(z)\right) = \tilde{\mathcal{P}}_{-b,-a}\left(x^{*}(z)\right) \tag{D.91}$$

$$\tilde{\mathcal{P}}_{a,b}\left(x^{*}(z)\right) = \tilde{\mathcal{P}}_{a,b}\left(x_{*}(z^{-1})\right) = \tilde{\mathcal{P}}^{*}_{-b,-a}\left(x(z)\right) \tag{D.92}$$

$$\tilde{\mathcal{P}}_{a,b}\left(x(z^{-1})\right) = \tilde{\mathcal{P}}^{*}_{-b,-a}\left(x_{*}(z)\right) \tag{D.93}$$

$$\tilde{\mathcal{P}}_{a,b}\left(x_{*}(z)\right) = \tilde{\mathcal{P}}^{*}_{-b,-a}\left(x(z^{-1})\right) \tag{D.94}$$

$$\tilde{\mathcal{P}}_{C}\left(x^{*}(z)\right) = \tilde{\mathcal{P}}^{*}_{C}\left(x(z)\right) \tag{D.95}$$

$$\tilde{\mathcal{P}}_{C}\left(x(z^{-1})\right) = \tilde{\mathcal{P}}^{*}_{C}\left(x_{*}(z)\right) \tag{D.96}$$

$$\tilde{\mathcal{P}}_{a,b}\left(\mathbf{X}^{T}(z)\right) = \tilde{\mathcal{P}}^{T}_{a,b}\left(\mathbf{X}(z)\right) \tag{D.97}$$

$$\tilde{\mathcal{P}}_{a,b}\left(\mathbf{X}^{H}(z)\right) = \tilde{\mathcal{P}}_{a,b}\left(\mathbf{X}^{T}_{*}(z^{-1})\right) = \tilde{\mathcal{P}}^{H}_{-b,-a}\left(\mathbf{X}(z)\right) \tag{D.98}$$

$$\tilde{\mathcal{P}}^{H}_{a,b}\left(\mathbf{X}(z)\right) = \tilde{\mathcal{P}}_{-b,-a}\left(\mathbf{X}^{H}(z)\right)\,. \tag{D.99}$$

Properties of time shifting

$$\tilde{\mathcal{P}}_{a,b}\left(x(z)\right) = z^{-a}\, \tilde{\mathcal{P}}_{0,b-a}\left(z^{a} x(z)\right) \tag{D.100}$$

$$\tilde{\mathcal{P}}_{a,b}\left(x(z)\right) = z^{d}\, \tilde{\mathcal{P}}_{a+d,b+d}\left(z^{-d}\, x(z)\right) \tag{D.101}$$

$$z^{-d}\, \tilde{\mathcal{P}}_{a,b}\left(x(z)\right) = \tilde{\mathcal{P}}_{a+d,b+d}\left(z^{-d}\, x(z)\right) \tag{D.102}$$

$$\tilde{\mathcal{P}}_{a,b}\left(z^{d}\, x(z)\right) = z^{d}\, \tilde{\mathcal{P}}_{a+d,b+d}\left(x(z)\right) \tag{D.103}$$

$$z^{-d}\, \tilde{\mathcal{P}}_{a,b}\left(z^{d}\, x(z)\right) = \tilde{\mathcal{P}}_{a+d,b+d}\left(x(z)\right)\,. \tag{D.104}$$

Furthermore, $\tilde{\mathcal{P}}$ also has the following properties ($k \in \mathbb{Z}$)

$$\tilde{\mathcal{P}}_{0,C-1}\left(x(z)\right) = \tilde{\mathcal{P}}_{0,C-1}\left(z^{kC}\, x(z)\right) \tag{D.105}$$

$$\tilde{\mathcal{P}}_{a,b}\left(x(z)\right) = \tilde{\mathcal{P}}_{a,b}\left(z^{k(b-a+1)}\, x(z)\right) \tag{D.106}$$

$$\tilde{\mathcal{P}}_{C}\left(x(z)\right) = \tilde{\mathcal{P}}_{C}\left(z^{kC}\, x(z)\right) \tag{D.107}$$

$$\tilde{\mathcal{P}}_{kC,kC+C-1}\left(x(z)\right) = z^{-kC}\, \tilde{\mathcal{P}}_{0,C-1}\left(x(z)\right) \tag{D.108}$$

$$\tilde{\mathcal{P}}_{a+k(b-a+1),b+k(b-a+1)}\left(x(z)\right) = z^{-k(b-a+1)}\, \tilde{\mathcal{P}}_{a,b}\left(x(z)\right)\,. \tag{D.109}$$

Properties of products

$$\tilde{\mathcal{P}}_{a,b}\left(x(z)\, y(z)\right) = \tilde{\mathcal{P}}_{a,b}\left(x(z)\, \tilde{\mathcal{P}}_{a,b}\left(y(z)\right)\right) \tag{D.110}$$

$$= \tilde{\mathcal{P}}_{a,b}\left(\tilde{\mathcal{P}}_{a,b}\left(x(z)\right) \tilde{\mathcal{P}}_{a,b}\left(y(z)\right)\right)\,. \tag{D.111}$$

Properties of the inverse (use (D.110) and (D.111) with $y(z) = x^{-1}(z)$)

$$\tilde{\mathcal{P}}_{a,b}\left(\tilde{\mathcal{P}}_{a,b}\left(x(z)\right) \tilde{\mathcal{P}}_{a,b}\left(x^{-1}(z)\right)\right) = 1 \tag{D.112}$$

$$\tilde{\mathcal{P}}_{C}\left(\tilde{\mathcal{P}}_{C}\left(x(z)\right) \tilde{\mathcal{P}}_{C}\left(x^{-1}(z)\right)\right) = 1 \tag{D.113}$$

$$\tilde{\mathcal{P}}_{a,b}\left(x^{-1}(z)\right) = \tilde{\mathcal{P}}_{a,b}\left(\left(\tilde{\mathcal{P}}_{a,b}\left(x(z)\right)\right)^{-1}\right) \tag{D.114}$$

$$\tilde{\mathcal{P}}_{C}\left(x^{-1}(z)\right) = \tilde{\mathcal{P}}_{C}\left(\left(\tilde{\mathcal{P}}_{C}\left(x(z)\right)\right)^{-1}\right)\,. \tag{D.115}$$

Combined properties of $\mathcal{P}$ and $\tilde{\mathcal{P}}$

$$\mathcal{P}_{a,b}\left(\tilde{\mathcal{P}}_{a,b}\left(x(z)\right)\right) = \tilde{\mathcal{P}}_{a,b}\left(x(z)\right) \tag{D.116}$$

$$\tilde{\mathcal{P}}_{a,b}\left(\mathcal{P}_{a,b}\left(x(z)\right)\right) = \mathcal{P}_{a,b}\left(x(z)\right) \tag{D.117}$$

but in general $\mathcal{P}_{a,b}\left(\tilde{\mathcal{P}}_{a,b}\left(x(z)\right)\right) \neq \tilde{\mathcal{P}}_{a,b}\left(\mathcal{P}_{a,b}\left(x(z)\right)\right)$.

Examples

$$\mathcal{P}_{-T_{\mathrm{x}},T_{\mathrm{x}}}\left(x(z)\right) = \mathcal{P}_{2T_{\mathrm{x}}+1}\left(x(z)\right) = \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} x_t\, z^{-t} \tag{D.118}$$

$$\tilde{\mathcal{P}}_{-T_{\mathrm{x}},T_{\mathrm{x}}}\left(x(z)\right) = \tilde{\mathcal{P}}_{2T_{\mathrm{x}}+1}\left(x(z)\right) = \sum_{t=-T_{\mathrm{x}}}^{T_{\mathrm{x}}} \sum_{k=-\infty}^{\infty} x_{t+k(2T_{\mathrm{x}}+1)}\, z^{-t} \tag{D.119}$$

Polynomial matrices or matrix polynomial

The projection operators $\mathcal{P}$ and $\tilde{\mathcal{P}}$ applied on polynomial matrices or a matrix polynomial is straightforward, i.e. $\mathcal{P}_{a,b}\left(\mathbf{A}(z)\right) = \sum_{n=a}^{b} \mathbf{A}_n z^{-n}$.

Sequences

The projection operators $\mathcal{P}$ and $\tilde{\mathcal{P}}$ applied on sequences are defined similarly as for polynomials, i.e.
$\mathcal{P}_{a,b}\left(\{\ldots, x_a, \ldots, x_b, \ldots\}\right) = \{\ldots, 0, x_a, \ldots, x_b, 0, \ldots\}$.

Appendix E

Update equations

In this Appendix we summarize the update equations of the algorithms for system identification, inverse modeling, and blind identification. The derivation of the non-blind algorithms is given in Chapter 2,4, and 5, those of the blind algorithms in Chapter 6.

cost function	$\hat{\mathbf{A}}$		$\hat{\mathbf{A}}^{-1}$
$J_{\text{MSE-x}}$	$\mathbf{H}_t$	$\longrightarrow$	$\mathbf{W}_t = \mathbf{H}_t^{-1}$
	RLS1-Hx	$\longrightarrow$	RLS1-Wx
	LMS1-Hx	$\longrightarrow$	LMS1a-Wx, LMS1b-Wx
	LMS2-Hx	$\longrightarrow$	LMS2a-Wx, LMS2b-Wx
$J_{\text{MSE-s}}$	$\mathbf{H}_t = \mathbf{W}_t^{-1}$	$\longleftarrow$	$\mathbf{W}_t$
	RLS2-Hs	$\longleftarrow$	RLS2-Ws
	LMS3a-Hs, LMS3b-Hs	$\longleftarrow$	LMS3-Ws
	LMS4a-Hs, LMS4b-Hs	$\longleftarrow$	LMS4-Ws

Table E.1: Relationships between update equations. We can choose between $J_{\text{MSE-x}}$ or $J_{\text{MSE-s}}$ for the desired cost functions and can either update $\mathbf{H}_t$ or $\mathbf{W}_t$.

non-blind algorithm		blind algorithm
RLS1-Wx	$\longrightarrow$	BRLS1a, BRLS1b, BRLS1c
RLS2-Ws	$\longrightarrow$	BRLS2
LMS1a-Wx	$\longrightarrow$	BLMS1a
LMS1b-Wx	$\longrightarrow$	BLMS1b
LMS2a-Wx	$\longrightarrow$	BLMS2a
LMS2b-Wx	$\longrightarrow$	BLMS2b

Table E.2: Relationships between non-blind and blind update equations.

Algorithm	Update equations
RLS1-Hx	$\breve{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \mathbf{s}_t}$ $\mathbf{H}_{t+1} = \mathbf{H}_t + \breve{\mu}_t\, \mathbf{e}_{\mathrm{x}t}\, \mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}$ $\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} - \breve{\mu}_t \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \mathbf{s}_t \mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\right)$
RLS2-Hs	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} \hat{\mathbf{x}}_t}$ $\mathbf{H}_{t+1} = \mathbf{H}_t + \mu_t\, \mathbf{e}_{\mathrm{x}t} \mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} \mathbf{H}_t$ $\breve{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} \mathbf{x}_t}$ $\hat{\mathbf{R}}_{\mathbf{xx}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} - \breve{\mu}_t \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} \mathbf{x}_t \mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1}\right)$
LMS1-Hx	$\mathbf{H}_{t+1} = \mathbf{H}_t + \mu\, \mathbf{e}_{\mathrm{x}t} \mathbf{s}_t^H$
LMS2-Hx	$\mathbf{H}_{t+1} = (1-\mu)\, \mathbf{H}_t + \mu \mathbf{x}_t \mathbf{s}_t^H$
LMS3a-Hs LMS4a-Hs	$\mathbf{H}_{t+1} = \mathbf{H}_t + \frac{\mu}{1-\mu \mathbf{x}_t^H \mathbf{e}_{\mathrm{x}t}}\, \mathbf{e}_{\mathrm{x}t} \mathbf{x}_t^H \mathbf{H}_t$
LMS3b-Hs LMS4b-Hs	$\mathbf{H}_{t+1} = \mathbf{H}_t + \mu\, \mathbf{e}_{\mathrm{x}t} \mathbf{x}_t^H \left[\mathbf{I} - \mu\, \mathbf{e}_{\mathrm{x}t} \mathbf{x}_t^H\right]^{-1} \mathbf{H}_t$

Table E.3: Update equations for **system identification** of an instantaneous mixing system. $(\mathbf{e}_{\mathrm{x}t} \triangleq \mathbf{x}_t - \hat{\mathbf{x}}_t)$

Algorithm	Update equations
RLS1-Hx	$\breve{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, s_t^* \hat{r}^{-1}_{ss\,t-1} s_t}$ $h_{t+1} = h_t + \breve{\mu}_t\, e_{\mathrm{x}t} s_t^* \hat{r}^{-1}_{ss\,t-1}$ $\hat{r}^{-1}_{ss\,t} = \frac{1}{\lambda}\left(\hat{r}^{-1}_{ss\,t-1} - \breve{\mu}_t \hat{r}^{-1}_{ss\,t-1} s_t s_t^* \hat{r}^{-1}_{ss\,t-1}\right)$
RLS2-Hs	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, x_t^* \hat{r}^{-1}_{xx\,t-1} \hat{x}_t}$ $h_{t+1} = h_t + \mu_t\, e_{\mathrm{x}t} x_t^* \hat{r}^{-1}_{xx\,t-1} h_t$ $\breve{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, x_t^* \hat{r}^{-1}_{xx\,t-1} x_t}$ $\hat{r}^{-1}_{xx\,t} = \frac{1}{\lambda}\left(\hat{r}^{-1}_{xx\,t-1} - \breve{\mu}_t \hat{r}^{-1}_{xx\,t-1} x_t x_t^* \hat{r}^{-1}_{xx\,t-1}\right)$
LMS1-Hx	$h_{t+1} = h_t + \mu\, e_{\mathrm{x}t} s_t^*$
LMS2-Hx	$h_{t+1} = (1-\mu)\, h_t + \mu x_t s_t^*$
LMS3a-Hs LMS4a-Hs	$h_{t+1} = h_t + \frac{\mu}{1-\mu\, x_t^* e_{\mathrm{x}t}}\, e_{\mathrm{x}t} x_t^* h_t$
LMS3b-Hs LMS4b-Hs	$h_{t+1} = h_t + \mu\, e_{\mathrm{x}t} x_t^* \left[1 - \mu\, e_{\mathrm{x}t} x_t^*\right]^{-1} h_t$

Table E.4: Update equations for **gain identification**. ($e_{\mathrm{x}t} \triangleq x_t - \hat{x}_t$)

Algorithm	Update equations
RLS1-Hx	$\bar{\bar{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda \mathbf{I} + (1-\lambda)\, \bar{\mathbf{S}}_k^H \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{ss}_{k-1}} \bar{\mathbf{S}}_k\right]^{-1}$ $\bar{\mathbf{H}}_{k+1} = \bar{\mathbf{H}}_k + \bar{\bar{\boldsymbol{\mu}}}_k \bar{\mathbf{E}}_{\mathrm{x}_k} \bar{\mathbf{S}}_k^H \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{ss}_{k-1}}$ $\bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{ss}_k} = \frac{1}{\lambda}\left(\bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{ss}_{k-1}} - \bar{\bar{\boldsymbol{\mu}}}_k \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{ss}_{k-1}} \bar{\mathbf{S}}_k \bar{\mathbf{S}}_k^H \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{ss}_{k-1}}\right)$
RLS2-Hs	$\bar{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\, \mathbf{I} + (1-\lambda)\, \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{xx}_{k-1}} \bar{\hat{\mathbf{X}}}_k\right]^{-1}$ $\bar{\mathbf{H}}_{k+1} = \bar{\mathbf{H}}_k + \bar{\boldsymbol{\mu}}_k \bar{\mathbf{E}}_{\mathrm{x}_k} \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{xx}_{k-1}} \bar{\mathbf{H}}_k$ $\bar{\bar{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda\, \mathbf{I} + (1-\lambda)\, \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{xx}_{k-1}} \bar{\mathbf{X}}_k\right]^{-1}$ $\bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{xx}_k} = \frac{1}{\lambda}\left(\bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{xx}_{k-1}} - \bar{\bar{\boldsymbol{\mu}}}_k \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{xx}_{k-1}} \bar{\mathbf{X}}_k \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}^{-1}_{\mathbf{xx}_{k-1}}\right)$
LMS1-Hx	$\bar{\mathbf{H}}_{k+1} = \bar{\mathbf{H}}_k + \mu \bar{\mathbf{E}}_{\mathrm{x}_k} \bar{\mathbf{S}}_k^H$
LMS2-Hx	$\bar{\mathbf{H}}_{k+1} = (1-\mu)\, \bar{\mathbf{H}}_k + \mu \bar{\mathbf{X}}_k \bar{\mathbf{S}}_k^H$
LMS3a-Hs LMS4a-Hs	$\bar{\mathbf{H}}_{k+1} = \bar{\mathbf{H}}_k + \mu \left[\mathbf{I} - \mu \bar{\mathbf{X}}^H \bar{\mathbf{E}}_{\mathrm{x}_k}\right]^{-1} \bar{\mathbf{E}}_{\mathrm{x}_k} \bar{\mathbf{X}}_k^H \bar{\mathbf{H}}_k$
LMS3b-Hs LMS4b-Hs	$\bar{\mathbf{H}}_{k+1} = \bar{\mathbf{H}}_k + \mu \bar{\mathbf{E}}_{\mathrm{x}_k} \bar{\mathbf{X}}_k^H \left[\mathbf{I} - \mu \bar{\mathbf{E}}_{\mathrm{x}_k} \bar{\mathbf{X}}_k^H\right]^{-1} \bar{\mathbf{H}}_k$

Table E.5: Update equations for **single-channel system identification**. By extending the diagonal matrices to block diagonal matrices, e.g. $\bar{\mathbf{H}}_k \rightarrow \overline{\mathbf{H}}_k$, etc., we obtain the update equations for MIMO system identification. ($\bar{\mathbf{E}}_{\mathrm{x}_k} \triangleq \bar{\mathbf{X}}_k - \bar{\hat{\mathbf{X}}}_k$)

Algorithm	Update equations
RLS1-Hx	$\overline{\breve{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda\mathbf{I} + (1-\lambda)\,\overline{\mathbf{S}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \overline{\mathbf{S}}_k\right]^{-1}$ $\overline{\mathbf{H}}_{k+1} = \overline{\mathbf{H}}_k + \overline{\boldsymbol{\mu}}_k \overline{\mathbf{E}}_{\mathrm{x}_k} \overline{\mathbf{S}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1}$ $\overline{\hat{\mathbf{R}}}_{\mathbf{ss}_k}^{-1} = \frac{1}{\lambda}\left(\overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} - \overline{\boldsymbol{\mu}}_k \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \overline{\mathbf{S}}_k \overline{\mathbf{S}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1}\right)$
RLS2-Hs	$\overline{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \overline{\hat{\mathbf{X}}}_k\right]^{-1}$ $\overline{\mathbf{H}}_{k+1} = \overline{\mathbf{H}}_k + \overline{\boldsymbol{\mu}}_k \overline{\mathbf{E}}_{\mathrm{x}_k} \overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \overline{\mathbf{H}}_k$ $\overline{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \overline{\mathbf{X}}_k\right]^{-1}$ $\overline{\hat{\mathbf{R}}}_{\mathbf{xx}_k}^{-1} = \frac{1}{\lambda}\left(\overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} - \overline{\boldsymbol{\mu}}_k \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \overline{\mathbf{X}}_k \overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}\right)$
LMS1-Hx	$\overline{\mathbf{H}}_{k+1} = \overline{\mathbf{H}}_k + \mu \overline{\mathbf{E}}_{\mathrm{x}_k} \overline{\mathbf{S}}_k^H$
LMS2-Hx	$\overline{\mathbf{H}}_{k+1} = (1-\mu)\,\overline{\mathbf{H}}_k + \mu \overline{\mathbf{X}}_k \overline{\mathbf{S}}_k^H$
LMS3a-Hs LMS4a-Hs	$\overline{\mathbf{H}}_{k+1} = \overline{\mathbf{H}}_k + \mu\left[\mathbf{I} - \mu \overline{\mathbf{X}}^H \overline{\mathbf{E}}_{\mathrm{x}_k}\right]^{-1} \overline{\mathbf{E}}_{\mathrm{x}_k} \overline{\mathbf{X}}_k^H \overline{\mathbf{H}}_k$
LMS3b-Hs LMS4b-Hs	$\overline{\mathbf{H}}_{k+1} = \overline{\mathbf{H}}_k + \mu \overline{\mathbf{E}}_{\mathrm{x}_k} \overline{\mathbf{X}}_k^H \left[\mathbf{I} - \mu \overline{\mathbf{E}}_{\mathrm{x}_k} \overline{\mathbf{X}}_k^H\right]^{-1} \overline{\mathbf{H}}_k$

Table E.6: Update equations for **multichannel system identification**. By substituting the block diagonal matrices to diagonal matrices, e.g. $\overline{\mathbf{H}}_k \rightarrow \breve{\mathbf{H}}_k$, etc., we obtain the update equations for SISO system identification. ($\overline{\mathbf{E}}_{\mathrm{x}_k} \triangleq \overline{\mathbf{X}}_k - \overline{\hat{\mathbf{X}}}_k$)

Algorithm	Update equations
RLS1-Wx	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \mathbf{u}_t}$ $\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t\, \mathbf{e}_{\mathrm{s}t} \mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \mathbf{W}_t$ $\breve{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \mathbf{s}_t}$ $\hat{\mathbf{R}}_{\mathbf{ss}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} - \breve{\mu}_t \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \mathbf{s}_t \mathbf{s}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1}\right)$
RLS2-Ws	$\breve{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} \mathbf{x}_t}$ $\mathbf{W}_{t+1} = \mathbf{W}_t + \breve{\mu}_t\, \mathbf{e}_{\mathrm{s}t} \mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1}$ $\hat{\mathbf{R}}_{\mathbf{xx}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} - \breve{\mu}_t \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} \mathbf{x}_t \mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1}\right)$
LMS1a-Wx	$\mathbf{W}_{t+1} = \mathbf{W}_t + \frac{\mu}{1-\mu \mathbf{s}_t^H \mathbf{e}_{\mathrm{s}t}}\, \mathbf{e}_{\mathrm{s}t} \mathbf{s}_t^H \mathbf{W}_t$
LMS1b-Wx	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\, \mathbf{e}_{\mathrm{s}t} \mathbf{s}_t^H \left[\mathbf{I} - \mu\, \mathbf{e}_{\mathrm{s}t} \mathbf{s}_t^H\right]^{-1} \mathbf{W}_t$
LMS2a-Wx	$\mathbf{W}_{t+1} = \frac{1}{1-\mu}\left(\mathbf{I} - \frac{\mu}{1-\mu+\mu \mathbf{s}_t^H \mathbf{u}_t} \mathbf{u}_t \mathbf{s}_t^H\right) \mathbf{W}_t$
LMS2b-Wx	$\mathbf{W}_{t+1} = \left[(1-\mu)\mathbf{I} + \mu \mathbf{u}_t \mathbf{s}_t^H\right]^{-1} \mathbf{W}_t$
LMS3-Ws	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\, \mathbf{e}_{\mathrm{s}t} \mathbf{x}_t^H$
LMS4-Ws	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\left(\mathbf{W}_t^{-H} - \mathbf{u}_t \mathbf{x}_t^H\right)$

Table E.7: Update equations for **inverse modeling** of an instantaneous mixing system. ($\mathbf{e}_{\mathrm{s}t} \triangleq \mathbf{s}_t - \mathbf{u}_t$)

Algorithm	Update equations
RLS1-Wx	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, s_t^* \hat{r}_{ss_{t-1}}^{-1} u_t}$ $w_{t+1} = w_t + \mu_t\, e_{\mathrm{s}t} s_t^* \hat{r}_{ss_{t-1}}^{-1} w_t$ $\check{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, s_t^* \hat{r}_{ss_{t-1}}^{-1} s_t}$ $\hat{r}_{ss_t}^{-1} = \frac{1}{\lambda}\left(\hat{r}_{ss_{t-1}}^{-1} - \check{\mu}_t \hat{r}_{ss_{t-1}}^{-1} s_t s_t^* \hat{r}_{ss_{t-1}}^{-1}\right)$
RLS2-Ws	$\check{\mu}_t = \frac{1-\lambda}{\lambda + (1-\lambda) x_t^* \hat{r}_{xx_{t-1}}^{-1} x_t}$ $w_{t+1} = w_t + \check{\mu}_t\, e_{\mathrm{s}t} x_t^* \hat{r}_{xx_{t-1}}^{-1}$ $\hat{r}_{xx_t}^{-1} = \frac{1}{\lambda}\left(\hat{r}_{xx_{t-1}}^{-1} - \check{\mu}_t \hat{r}_{xx_{t-1}}^{-1} x_t x_t^* \hat{r}_{xx_{t-1}}^{-1}\right)$
LMS1a-Wx	$w_{t+1} = w_t + \frac{\mu}{1-\mu\, s_t^* e_{\mathrm{s}t}}\, e_{\mathrm{s}t} s_t^* w_t$
LMS1b-Wx	$w_{t+1} = w_t + \mu\, e_{\mathrm{s}t} s_t^* \left[1 - \mu\, e_{\mathrm{s}t} s_t^*\right]^{-1} w_t$
LMS2a-Wx	$w_{t+1} = \frac{1}{1-\mu}\left(1 - \frac{\mu}{1-\mu+\mu s_t^* u_t} u_t s_t^*\right) w_t$
LMS2b-Wx	$w_{t+1} = \left[(1-\mu) + \mu u_t s_t^*\right]^{-1} w_t$
LMS3-Ws	$w_{t+1} = w_t + \mu\, e_{\mathrm{s}t} x_t^*$
LMS4-Ws	$w_{t+1} = w_t + \mu\left(w_t^{-*} - u_t x_t^*\right)$

Table E.8: Update equations for **inverse-gain identification**. $(e_{\mathrm{s}t} \triangleq s_t - u_t)$

Algorithm	Update equations
RLS1-Wx	$\bar{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda \mathbf{I} + (1-\lambda)\, \bar{\mathbf{S}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \bar{\mathbf{U}}_k\right]^{-1}$ $\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \bar{\boldsymbol{\mu}}_k \bar{\mathbf{E}}_{\mathrm{s}_k} \bar{\mathbf{S}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \bar{\mathbf{W}}_k$ $\bar{\check{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda \mathbf{I} + (1-\lambda)\, \bar{\mathbf{S}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \bar{\mathbf{S}}_k\right]^{-1}$ $\bar{\hat{\mathbf{R}}}_{\mathbf{ss}_k}^{-1} = \frac{1}{\lambda}\left(\bar{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} - \bar{\check{\boldsymbol{\mu}}}_k \bar{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \bar{\mathbf{S}}_k \bar{\mathbf{S}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1}\right)$
RLS2-Ws	$\bar{\check{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda \mathbf{I} + (1-\lambda)\, \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \bar{\mathbf{X}}_k\right]^{-1}$ $\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \bar{\check{\boldsymbol{\mu}}}_k \bar{\mathbf{E}}_{\mathrm{s}_k} \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}$ $\bar{\hat{\mathbf{R}}}_{\mathbf{xx}_k}^{-1} = \frac{1}{\lambda}\left(\bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} - \bar{\check{\boldsymbol{\mu}}}_k \bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \bar{\mathbf{X}}_k \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}\right)$
LMS1a-Wx	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu\left[\mathbf{I} - \mu \bar{\mathbf{S}}_k^H \bar{\mathbf{E}}_{\mathrm{s}_k}\right]^{-1} \bar{\mathbf{E}}_{\mathrm{s}_k} \bar{\mathbf{S}}_k^H \bar{\mathbf{W}}_k$
LMS1b-Wx	$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu \bar{\mathbf{E}}_{\mathrm{s}_k} \bar{\mathbf{S}}_k^H \left[\mathbf{I} - \mu \bar{\mathbf{E}}_{\mathrm{s}_k} \bar{\mathbf{S}}_k^H\right]^{-1} \mathbf{W}_k$
LMS2a-Wx	$\bar{\mathbf{W}}_{k+1} = \frac{1}{1-\mu}\left(\mathbf{I} - \mu\left[(1-\mu)\mathbf{I} + \mu \bar{\mathbf{S}}_k^H \bar{\mathbf{U}}_k\right]^{-1} \bar{\mathbf{U}}_k \bar{\mathbf{S}}_k^H\right) \bar{\mathbf{W}}_k$
LMS2b-Wx	$\bar{\mathbf{W}}_{k+1} = \left[(1-\mu)\mathbf{I} + \mu \bar{\mathbf{U}}_k \bar{\mathbf{S}}_k^H\right]^{-1} \bar{\mathbf{W}}_k$
LMS3-Ws	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu \bar{\mathbf{E}}_{\mathrm{s}_k} \bar{\mathbf{X}}_k^H$
LMS4-Ws	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu\left(\bar{\mathbf{W}}_k^{-H} - \bar{\mathbf{U}}_k \bar{\mathbf{X}}_k^H\right)$

Table E.9: Update equations for **single-channel inverse modelling**. By extending the diagonal matrices to block diagonal matrices, e.g. $\bar{\mathbf{W}}_k \rightarrow \overline{\mathbf{W}}_k$, etc., we obtain the update equations for MIMO inverse modelling. $(\bar{\mathbf{E}}_{\mathrm{s}_k} \triangleq \bar{\mathbf{S}}_k - \bar{\mathbf{U}}_k)$

Algorithm	Update equations
RLS1-Wx	$\overline{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\mathbf{I} + (1-\lambda)\,\overline{\mathbf{S}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \overline{\mathbf{U}}_k\right]^{-1}$ $\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \overline{\boldsymbol{\mu}}_k \overline{\mathbf{E}}_{\mathrm{s}_k} \overline{\mathbf{S}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \overline{\mathbf{W}}_k$ $\overline{\bar{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda\mathbf{I} + (1-\lambda)\,\overline{\mathbf{S}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \overline{\mathbf{S}}_k\right]^{-1}$ $\overline{\hat{\mathbf{R}}}_{\mathbf{ss}_k}^{-1} = \frac{1}{\lambda}\left(\overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} - \overline{\bar{\boldsymbol{\mu}}}_k \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \overline{\mathbf{S}}_k \overline{\mathbf{S}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1}\right)$
RLS2-Ws	$\overline{\bar{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda\mathbf{I} + (1-\lambda)\,\overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \overline{\mathbf{X}}_k\right]^{-1}$ $\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \overline{\bar{\boldsymbol{\mu}}}_k \overline{\mathbf{E}}_{\mathrm{s}_k} \overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}$ $\overline{\hat{\mathbf{R}}}_{\mathbf{xx}_k}^{-1} = \frac{1}{\lambda}\left(\overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} - \overline{\bar{\boldsymbol{\mu}}}_k \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \overline{\mathbf{X}}_k \overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}\right)$
LMS1a-Wx	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu\left[\mathbf{I} - \mu\overline{\mathbf{S}}_k^H \overline{\mathbf{E}}_{\mathrm{s}_k}\right]^{-1} \overline{\mathbf{E}}_{\mathrm{s}_k} \overline{\mathbf{S}}_k^H \overline{\mathbf{W}}_k$
LMS1b-Wx	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu\overline{\mathbf{E}}_{\mathrm{s}_k} \overline{\mathbf{S}}_k^H \left[\mathbf{I} - \mu\overline{\mathbf{E}}_{\mathrm{s}_k} \overline{\mathbf{S}}_k^H\right]^{-1} \overline{\mathbf{W}}_k$
LMS2a-Wx	$\overline{\mathbf{W}}_{k+1} = \frac{1}{1-\mu}\left(\mathbf{I} - \mu\left[(1-\mu)\mathbf{I} + \mu\overline{\mathbf{S}}_k^H \overline{\mathbf{U}}_k\right]^{-1} \overline{\mathbf{U}}_k \overline{\mathbf{S}}_k^H\right) \overline{\mathbf{W}}_k$
LMS2b-Wx	$\overline{\mathbf{W}}_{k+1} = \left[(1-\mu)\mathbf{I} + \mu\overline{\mathbf{U}}_k \overline{\mathbf{S}}_k^H\right]^{-1} \overline{\mathbf{W}}_k$
LMS3-Ws	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu\overline{\mathbf{E}}_{\mathrm{s}_k} \overline{\mathbf{X}}_k^H$
LMS4-Ws	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu\left(\overline{\mathbf{W}}_k^{-H} - \overline{\mathbf{U}}_k \overline{\mathbf{X}}_k^H\right)$

Table E.10: Update equations for **multichannel inverse modelling**. By substituting the block diagonal matrices to diagonal matrices, e.g. $\overline{\mathbf{W}}_k \to \tilde{\mathbf{W}}_k$, etc., we obtain the update equations for SISO system identification. $(\overline{\mathbf{E}}_{\mathrm{s}_k} \triangleq \overline{\mathbf{S}}_k - \overline{\mathbf{U}}_k)$

Algorithm	Update equations
BRLS1a	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{u}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \mathbf{u}_t}$ $\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t\,\mathbf{e}_{\mathrm{b}t}\mathbf{u}_t^H \hat{\mathbf{R}}_{\mathbf{ss}_{t-1}}^{-1} \mathbf{W}_t$
BRLS1b	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{u}_t^H \hat{\mathbf{R}}_{\mathbf{ss}}^{-1} \mathbf{u}_t}$ $\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t\left(\mathbf{I} - \mathbf{y}_t\mathbf{u}_t^H\right) \hat{\mathbf{R}}_{\mathbf{ss}}^{-1} \mathbf{W}_t$
BRLS1c	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{u}_t^H \mathbf{u}_t}$ $\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_t\left(\mathbf{I} - \mathbf{y}_t\mathbf{u}_t^H\right) \mathbf{W}_t$
BRLS2	$\breve{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\,\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} \mathbf{x}_t}$ $\mathbf{W}_{t+1} = \mathbf{W}_t + \breve{\mu}_t\,\mathbf{e}_{\mathrm{b}t}\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1}$ $\hat{\mathbf{R}}_{\mathbf{xx}_t}^{-1} = \frac{1}{\lambda}\left(\hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} - \breve{\mu}_t \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1} \mathbf{x}_t\mathbf{x}_t^H \hat{\mathbf{R}}_{\mathbf{xx}_{t-1}}^{-1}\right)$
BLMS1a	$\mathbf{W}_{t+1} = \mathbf{W}_t + \frac{\mu}{1-\mu\,\mathbf{u}_t^H \mathbf{e}_{\mathrm{b}t}}\,\mathbf{e}_{\mathrm{b}t}\mathbf{u}_t^H \mathbf{W}_t$
BLMS1b	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\,\mathbf{e}_{\mathrm{b}t}\mathbf{u}_t^H \left[\mathbf{I} - \mu\,\mathbf{e}_{\mathrm{b}t}\mathbf{u}_t^H\right]^{-1} \mathbf{W}_t$
BLMS2a	$\mathbf{W}_{t+1} = \frac{1}{1-\mu}\left(\mathbf{I} - \frac{\mu}{1-\mu+\mu\mathbf{u}_t^H \mathbf{u}_t}\mathbf{y}_t\mathbf{u}_t^H\right) \mathbf{W}_t$
BLMS2b	$\mathbf{W}_{t+1} = \left[(1-\mu)\mathbf{I} + \mu\mathbf{y}_t\mathbf{u}_t^H\right]^{-1} \mathbf{W}_t$
BLMS3	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\,\mathbf{e}_{\mathrm{b}t}\mathbf{x}_t^H$
BLMS4	$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu\left(\mathbf{W}_t^{-H} - \mathbf{y}_t\mathbf{x}_t^H\right)$

Table E.11: Update equations for **blind source separation**. $(\mathbf{e}_{\mathrm{b}t} \triangleq \mathbf{u}_t - \mathbf{y}_t)$

Algorithm	Update equations
BRLS1a	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, u_t^* \hat{r}_{ss_{t-1}}^{-1} u_t}$ $w_{t+1} = w_t + \mu_t\, e_{\mathrm{b}t} u_t^* \hat{r}_{ss_{t-1}}^{-1} w_t$
BRLS1b	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, u_t^* \hat{r}_{ss}^{-1} u_t}$ $w_{t+1} = w_t + \mu_t\, (1 - y_t u_t^*)\, \hat{r}_{ss}^{-1} w_t$
BRLS1c	$\mu_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, u_t^* u_t}$ $w_{t+1} = w_t + \mu_t\, (1 - y_t u_t^*)\, w_t$
BRLS2	$\check{\mu}_t = \dfrac{1-\lambda}{\lambda + (1-\lambda)\, x_t^* \hat{r}_{xx_{t-1}}^{-1} x_t}$ $w_{t+1} = w_t + \check{\mu}_t\, e_{\mathrm{b}t} x_t^* \hat{r}_{xx_{t-1}}^{-1}$ $\hat{r}_{xx_t}^{-1} = \frac{1}{\lambda}\left(\hat{r}_{xx_{t-1}}^{-1} - \check{\mu}_t \hat{r}_{xx_{t-1}}^{-1} x_t x_t^* \hat{r}_{xx_{t-1}}^{-1}\right)$
BLMS1a	$w_{t+1} = w_t + \frac{\mu}{1-\mu\, u_t^*\, e_{\mathrm{b}t}}\, e_{\mathrm{b}t} u_t^* w_t$
BLMS1b	$w_{t+1} = w_t + \mu\, e_{\mathrm{b}t} u_t^* \left[1 - \mu\, e_{\mathrm{b}t} u_t^*\right]^{-1} w_t$
BLMS2a	$w_{t+1} = \frac{1}{1-\mu}\left(1 - \frac{\mu}{1-\mu+\mu u_t^* u_t} y_t u_t^*\right) w_t$
BLMS2b	$w_{t+1} = \left[1 - \mu + \mu y_t u_t^*\right]^{-1} w_t$
BLMS3	$w_{t+1} = w_t + \mu\, e_{\mathrm{b}t} x_t^*$
BLMS4	$w_{t+1} = w_t + \mu\left(w_t^{-*} - y_t x_t^*\right)$

Table E.12: Update equations for **AGC**. ($e_{\mathrm{b}t} \triangleq u_t - y_t$)

Algorithm	Update equations
BRLS1a	$\bar{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\bar{\mathbf{U}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \bar{\mathbf{U}}_k\right]^{-1}$ $\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \bar{\boldsymbol{\mu}}_k \bar{\mathbf{E}}_{\mathrm{b}_k} \bar{\mathbf{U}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \bar{\mathbf{W}}_k$
BRLS1b	$\bar{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\bar{\mathbf{U}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{ss}}^{-1} \bar{\mathbf{U}}_k\right]^{-1}$ $\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \bar{\boldsymbol{\mu}}_k \left(\mathbf{I} - \bar{\mathbf{Y}}_k \bar{\mathbf{U}}_k^H\right) \bar{\hat{\mathbf{R}}}_{\mathbf{ss}}^{-1} \bar{\mathbf{W}}_k$
BRLS1c	$\bar{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\bar{\mathbf{U}}_k^H \bar{\mathbf{U}}_k\right]^{-1}$ $\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \bar{\boldsymbol{\mu}}_k \left(\mathbf{I} - \bar{\mathbf{Y}}_k \bar{\mathbf{U}}_k^H\right) \bar{\mathbf{W}}_k$
BRLS2	$\bar{\check{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \bar{\mathbf{X}}_k\right]^{-1}$ $\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \bar{\check{\boldsymbol{\mu}}}_k \bar{\mathbf{E}}_{\mathrm{b}_k} \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}$ $\bar{\hat{\mathbf{R}}}_{\mathbf{xx}_k}^{-1} = \frac{1}{\lambda}\left(\bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} - \bar{\check{\boldsymbol{\mu}}}_k \bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \bar{\mathbf{X}}_k \bar{\mathbf{X}}_k^H \bar{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}\right)$
BLMS1a	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu\left[\mathbf{I} - \mu \bar{\mathbf{U}}_k^H \bar{\mathbf{E}}_{\mathrm{b}_k}\right]^{-1} \bar{\mathbf{E}}_{\mathrm{b}_k} \bar{\mathbf{U}}_k^H \bar{\mathbf{W}}_k$
BLMS1b	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu \bar{\mathbf{E}}_{\mathrm{b}_k} \bar{\mathbf{U}}_k^H \left[\mathbf{I} - \mu \bar{\mathbf{E}}_{\mathrm{b}_k} \bar{\mathbf{U}}_k^H\right]^{-1} \bar{\mathbf{W}}_k$
BLMS2a	$\bar{\mathbf{W}}_{k+1} = \frac{1}{1-\mu}\left(\mathbf{I} - \mu\left[(1-\mu)\mathbf{I} + \mu \bar{\mathbf{U}}_k^H \bar{\mathbf{U}}_k\right]^{-1} \bar{\mathbf{Y}}_k \bar{\mathbf{U}}_k^H\right) \bar{\mathbf{W}}_k$
BLMS2b	$\bar{\mathbf{W}}_{k+1} = \left[(1-\mu)\mathbf{I} + \mu \bar{\mathbf{Y}}_k \bar{\mathbf{U}}_k^H\right]^{-1} \bar{\mathbf{W}}_k$
BLMS3	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu \bar{\mathbf{E}}_{\mathrm{b}_k} \bar{\mathbf{X}}_k^H$
BLMS4	$\bar{\mathbf{W}}_{k+1} = \bar{\mathbf{W}}_k + \mu\left(\bar{\mathbf{W}}_k^{-H} - \bar{\mathbf{Y}}_k \bar{\mathbf{X}}_k^H\right)$

Table E.13: Update equations for **single-channel blind deconvolution**. ($\bar{\mathbf{E}}_{\mathrm{b}_k} \triangleq \bar{\mathbf{U}}_k - \bar{\mathbf{Y}}_k$)

Algorithm	Update equations
BRLS1a	$\overline{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\overline{\mathbf{U}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \overline{\mathbf{U}}_k\right]^{-1}$ $\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \overline{\boldsymbol{\mu}}_k \overline{\mathbf{E}}_{\mathrm{b}_k} \overline{\mathbf{U}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}_{k-1}}^{-1} \overline{\mathbf{W}}_k$
BRLS1b	$\overline{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\overline{\mathbf{U}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{ss}}^{-1} \overline{\mathbf{U}}_k\right]^{-1}$ $\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \overline{\boldsymbol{\mu}}_k \left(\mathbf{I} - \overline{\mathbf{Y}}_k \overline{\mathbf{U}}_k^H\right) \overline{\hat{\mathbf{R}}}_{\mathbf{ss}}^{-1} \overline{\mathbf{W}}_k$
BRLS1c	$\overline{\boldsymbol{\mu}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\overline{\mathbf{U}}_k^H \overline{\mathbf{U}}_k\right]^{-1}$ $\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \overline{\boldsymbol{\mu}}_k \left(\mathbf{I} - \overline{\mathbf{Y}}_k \overline{\mathbf{U}}_k^H\right) \overline{\mathbf{W}}_k$
BRLS2	$\overline{\bar{\boldsymbol{\mu}}}_k = (1-\lambda)\left[\lambda\,\mathbf{I} + (1-\lambda)\,\overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \overline{\mathbf{X}}_k\right]^{-1}$ $\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \overline{\bar{\boldsymbol{\mu}}}_k \overline{\mathbf{E}}_{\mathrm{b}_k} \overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}$ $\overline{\hat{\mathbf{R}}}_{\mathbf{xx}_k}^{-1} = \frac{1}{\lambda}\left(\overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} - \overline{\bar{\boldsymbol{\mu}}}_k \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1} \overline{\mathbf{X}}_k \overline{\mathbf{X}}_k^H \overline{\hat{\mathbf{R}}}_{\mathbf{xx}_{k-1}}^{-1}\right)$
BLMS1a	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu\left[\mathbf{I} - \mu \overline{\mathbf{U}}_k^H \overline{\mathbf{E}}_{\mathrm{b}_k}\right]^{-1} \overline{\mathbf{E}}_{\mathrm{b}_k} \overline{\mathbf{U}}_k^H \overline{\mathbf{W}}_k$
BLMS1b	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu \overline{\mathbf{E}}_{\mathrm{b}_k} \overline{\mathbf{U}}_k^H \left[\mathbf{I} - \mu \overline{\mathbf{E}}_{\mathrm{b}_k} \overline{\mathbf{U}}_k^H\right]^{-1} \overline{\mathbf{W}}_k$
BLMS2a	$\overline{\mathbf{W}}_{k+1} = \frac{1}{1-\mu}\left(\mathbf{I} - \mu\left[(1-\mu)\mathbf{I} + \mu \overline{\mathbf{U}}_k^H \overline{\mathbf{U}}_k\right]^{-1} \overline{\mathbf{Y}}_k \overline{\mathbf{U}}_k^H\right) \overline{\mathbf{W}}_k$
BLMS2b	$\overline{\mathbf{W}}_{k+1} = \left[(1-\mu)\mathbf{I} + \mu \overline{\mathbf{Y}}_k \overline{\mathbf{U}}_k^H\right]^{-1} \overline{\mathbf{W}}_k$
BLMS3	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu \overline{\mathbf{E}}_{\mathrm{b}_k} \overline{\mathbf{X}}_k^H$
BLMS4	$\overline{\mathbf{W}}_{k+1} = \overline{\mathbf{W}}_k + \mu\left(\overline{\mathbf{W}}_k^{-H} - \overline{\mathbf{Y}}_k \overline{\mathbf{X}}_k^H\right)$

Table E.14: Update equations for **multichannel blind deconvolution**. ($\overline{\mathbf{E}}_{\mathrm{b}_k} \triangleq \overline{\mathbf{U}}_k - \overline{\mathbf{Y}}_k$)

Appendix F

Implementation of a single-channel blind deconvolution algorithm

A frequency-domain blind deconvolution algorithm was presented recently by Douglas and Kung in [35]. In this Appendix we present an alternative implementation in MATLAB of a single-channel frequency-domain blind deconvolution algorithm as described in Section 6.8.2. Several update equations can be chosen in `FDBDeconv.m`, such as the natural gradient algorithm, the EASI algorithm, or the Infomax algorithm. The algorithm can handle complex-valued signals and coefficients. Depending on the choice of the nonlinearity $g(.)$, the source signal can be either sub-Gaussian or super-Gaussian. As an example, the non-causal filter and update equations of the natural gradient learning algorithm in the time domain at block k are [5, 6]:

$$u_t = \sum\nolimits_{n=-N_\mathrm{w}}^{N_\mathrm{w}} w_{n,k}\, x_{t-n} \tag{F.1}$$

$$v_t = \sum\nolimits_{n=-N_\mathrm{w}}^{N_\mathrm{w}} w_{-n,k}^*\, u_{t-n} \tag{F.2}$$

$$w_{n,k+1} = w_{n,k} + \frac{\mu}{2T_\mathrm{y}+1} \sum\nolimits_{t=-T_\mathrm{y}}^{T_\mathrm{y}} w_{n,k} - g(u_t)\, v_{t-n}^*\,. \tag{F.3}$$

The constraints of the algorithm to avoid boundary or circular wrap-around effects of the convolutions are $T_\mathrm{v} \geq T_\mathrm{y} + N_\mathrm{w}$, $T_\mathrm{u} \geq T_\mathrm{v} + N_\mathrm{w}$, $T_\mathrm{x} \geq T_\mathrm{u} + N_\mathrm{w}$, and $C \geq 2T_\mathrm{x} + 1$. The algorithm can be made causal by delaying the filtering and update.

F.1 FDBDeconv.m

```
%FDBDeconv  Frequency-domain blind deconvolution.
%
%   Marcel Joho, 24.11.2000, (joho@isi.ee.ethz.ch)
%

clear
%%% parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C=512; Nw=50; Tx=250; iter=500; plot_isi=20;
Tu=Tx-Nw; Ty=Tx-3*Nw; L=2*Tu+1; L2=2*Ty+1;

%%% initial conditions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
len_s=iter*L+L;
x=zeros(C,1); u=x; y=u; u_out=zeros(L*iter,1);

Unit=fft([1;zeros(C-1,1)],C); W=Unit;

%%% a(z) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a=[ -1-4i 1-5i -11-2i -17-11i -1+20i]/10;       % a(z)

Na=length(a); a=a(:)/sqrt(sum(a.*conj(a)));
a=[a;zeros(C-Na,1)]; A=fft(a,C);

%%% projection matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ng=Na+Nw-1;
gg=[(C-Ng+1:C) (1:Ng+1)]';
ww=[(C-Nw+1:C) (1:Nw+1)]'; Pw=zeros(C,1); Pw(ww)=1;
xx=[(C-Tx+1:C) (1:Tx+1)]';
uu=[(C-Tu+1:C) (1:Tu+1)]'; Pu=zeros(C,1); Pu(uu)=1;
yy=[(C-Ty+1:C) (1:Ty+1)]'; Py=zeros(C,1); Py(yy)=1;

%%% optimal deconvolution filter w(z) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
W_opt=1./A; w_opt=ifft(W_opt);
figure(1); plot((-Nw:Nw),[real(w_opt(ww)) imag(w_opt(ww))]);
ylabel('w_{opt}(z)'); xlabel('tap n'); drawnow ; pause(1);

%%% source signal %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%s=randn(len_s,2);s=sign(s).*s.^2;s_seq=s*[1;i]/sqrt(6);  % super-G
s_seq=(sign(randn(len_s,1))+i*sign(randn(len_s,1)))/sqrt(2);% sub-G

n_seq=0.032*(randn(len_s,1) + i*randn(len_s,1)) / sqrt(2);
x_seq=filter(a(1:Na),[1],s_seq) + n_seq;
```



```
%%% algorithm %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1:iter
  t=k*L; k

  x(xx)=x_seq(t-Tx:t+Tx); X=fft(x);
  U=W.*X; u=Pu.*ifft(U); u_out((k-1)*L+1:k*L) = u(uu);

  %%% nonlinearity y=g(u) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  ur=real(u); ui=imag(u);
% y=sign(ur) + i*sign(ui);              % super-G (re/im)
% y=sign(u);                            % super-G (abs)
% y=ur.*abs(ur).^2 + i*ui.*abs(ui).^2;  %   sub-G (re/im)
  y=u.*abs(u).^2;                       %   sub-G (abs)

  eb=Pu.*(u-y); Eb=fft(eb);             % blind error
  y=Py.*y; Y=fft(y);

  %%% update equations %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                          % EASI
% W  = W + 0.05/L2 * (L2*Unit-U.*conj(U)+U.*conj(Y)-Y.*conj(U)).*W;
  W  = W + 0.05 * (Unit - Y.*conj(U)/L2).*W;             % nat.grad.I
% W  = W + 0.05 * (W - Y.*conj(U).*W/L2);                % nat.grad.II
% W  = W + 0.05/L2 * (L2*conj(W).^(-1) - Y.*conj(X));    % Infomax

% mu = 0.3/L; mu_2 = mu./(Unit-mu*conj(U).*Eb);          % BLMS-1Wx
% W  = W + mu_2.*Eb.*conj(U).*W;                         % BLMS-1Wx
% W  = W + 0.3/L*Eb.*conj(U).*W;                         % BLMS-1Wx-c
% mu = 0.05; mu_2 = mu./((1-mu)*Unit+mu/L*conj(U).*U);   % BLMS-2Wx-a
% W  = 1/(1-mu) * (Unit - mu_2/L2.*Y.*conj(U)).*W;       % BLMS-2Wx-a
% mu = 0.05; W = 1./((1-mu)*Unit + mu/L*Y.*conj(U)).*W;  % BLMS-2Wx-b
% W  = W + 0.05/L * Eb.*conj(X);                         % BLMS-3Ws
% W  = W + 0.05/L2 * (L2*conj(W).^(-1) - Y.*conj(X));    % BLMS-4Ws

  %%% filter projection operation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  w=Pw.*ifft(W); W=fft(w);

  %%% performance %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  G=W.*A; g=ifft(G);
  Jisi(k)=sum(g.*conj(g))/max(g.*conj(g))-1;

  %%% plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  if rem(k,plot_isi)==0,
    figure(1); plot(10*log10(Jisi));
               ylabel('J_{ISI}  [dB]');xlabel('k')
    figure(2); plot((-Nw:Nw),[real(w(ww)) imag(w(ww))]);
               ylabel('w(z)');xlabel('tap n')
    figure(3); plot((-Ng:Ng),[real(g(gg)) imag(g(gg))]);
               ylabel('g(z)');xlabel('tap n')
    figure(4); plot(real(u(uu)),imag(u(uu)),'.');
    drawnow
  end
end
```

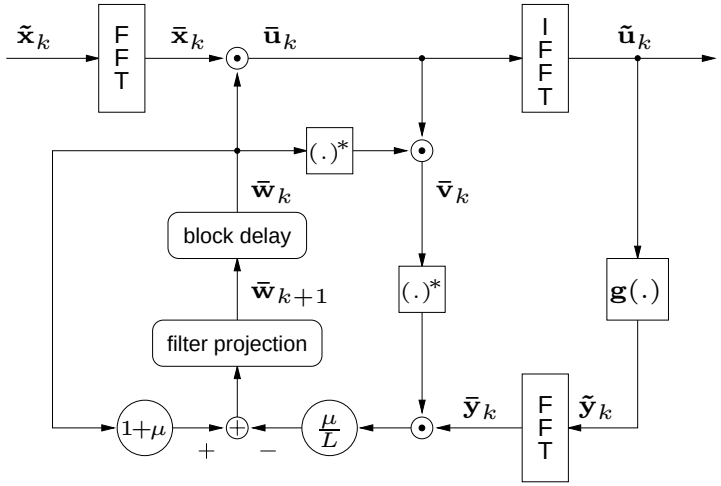

Figure F.1: Block diagram of the frequency-domain realization using the natural gradient learning algorithm.

F.2 Performance plots

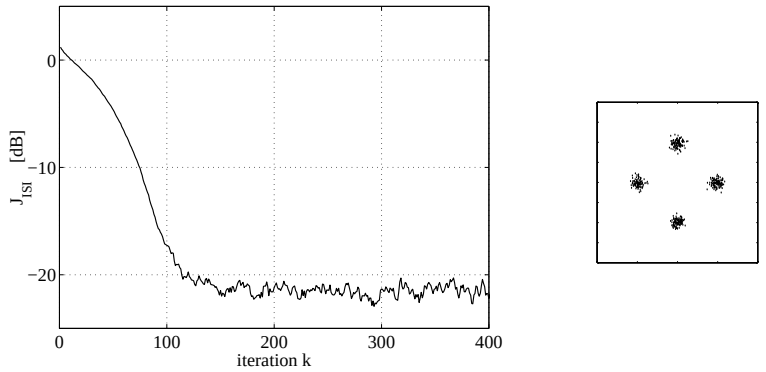

Figure F.2: Performance curve and constellation diagram of the natural gradient learning algorithm with a QPSK source signal and $g(u) = u|u|^2$.

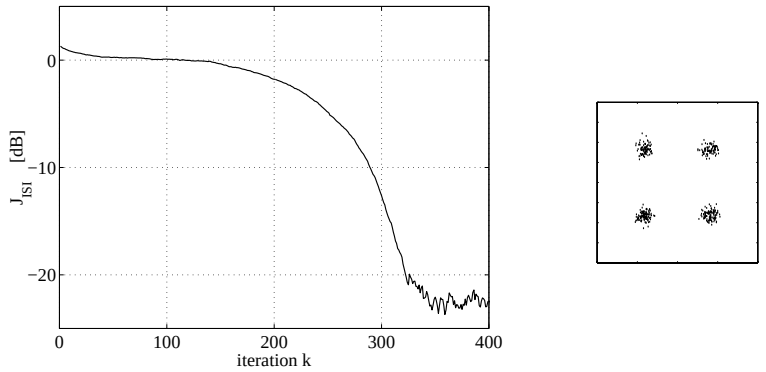

Figure F.3: Performance curve and constellation diagram of the natural gradient learning algorithm with a QPSK source signal and $g(u_{\mathrm{re}} + ju_{\mathrm{im}}) = u_{\mathrm{re}}|u_{\mathrm{re}}|^2 + ju_{\mathrm{im}}|u_{\mathrm{im}}|^2$.

List of Abbreviations

AGC	Automatic gain control
BD	Blind deconvolution
BIBO	Bounded input – bounded output
BLMS	Blind LMS
BRLS	Blind RLS
BSS	Blind source separation
CLT	Central limit theorem
CMA	Constant modulus algorithm
DFT	Discrete Fourier transform
ETH	Eidgenössische Technische Hochschule (Swiss Federal Institute of Technology)
FDAF	Frequency-domain adaptive filter
FDLMS	Frequency-domain LMS
FFT	Fast Fourier transform
FIR	Finite impulse response
FLMS	Fast LMS
HOS	Higher-order statistics
ICA	Independent component analysis
ICI	Interchannel interference
IDFT	Inverse discrete Fourier transform
iff	if and only if
IFFT	Inverse fast Fourier transform

iid	independent identical distributed
IIR	Infinite impulse response
ISI	Intersymbol interference, Signal and Information Processing Laboratory
LMS	Least-mean square
MAP	Maximum *a posteriori*
MCBD	Multichannel blind deconvolution
MIL	Matrix-inversion lemma (A.5)
MIMO	Multiple input – multiple output
ML	Maximum likelihood
MMSE	Minimum mean-square error
MSE	Mean-squared error
NLMS	Normalized LMS
OLA	Overlap add
OLS	Overlap save
PCA	Principal component analysis
pdf	probability density function
QAM	Quadrature amplitude modulation
QPSK	Quaternary phase shift keying
RLS	Recursive least squares
SINR	Signal-to-interference-and-noise ratio
SIR	Signal-to-interference ratio
SISO	Single input – single output
SNR	Signal-to-noise ratio
SOS	Second-order statistics
SVD	Singular value decomposition
WHE	Wiener-Hopf equation
ZF	Zero forcing

List of Symbols

Scalars

κ	Kurtosis (6.9)
λ, λ_m	Forgetting factor, eigenvalue
$\mu, \tilde{\mu}$	Stepsize
ω	Radian frequency ($2\pi f$)
σ, σ_1	Singular value, largest singular value
σ_{n}^2	Power of sensor noise
σ_{s}^2	Power of source signals
C	DFT/FFT size
f	Frequency [Hz]
f_s	Sampling frequency [Hz]
j	$\sqrt{-1}$
J_{ICI}	Interchannel interference cost function (6.130), (6.132)
J_{ISI}	Intersymbol interference cost function (6.131), (6.133)
$J_{\text{MC-ISI}}$	Multichannel intersymbol interference cost function (6.134)
$J_{\text{MSE-s}}$	Cost function for inverse modeling (2.57), (4.210), (5.114)
$J_{\text{MSE-x}}$	Cost function for system identification (2.4), (4.209), (5.113)
k	Iteration, block index $t \triangleq kL$
L	Block length
M	No. of sensors
M_{s}	No. of source signals

N	Filter length
t	Time sample index, discrete time $t_c = t\,T_s$
t_c	Continuous time
T_s	Sampling period
z	z-transform operator

Vectors and Matrices

$\mathbf{n}$	Sensor-noise vector
$\mathbf{s}$	Source-signal vector
$\mathbf{u}$	De-mixing system output vector
$\mathbf{x}$	Sensor-signal vector
$\hat{\mathbf{x}}$	Estimation of $\mathbf{x}$
$\mathbf{y}$	$\mathbf{g}(\mathbf{u})$
$\mathbf{A}, \mathbf{A}(z)$	Mixing system
$\mathbf{H}, \mathbf{H}(z)$	Estimation of mixing system $\mathbf{A}$
$\mathbf{H}^{\text{MMSE-s}}$	$\triangleq [\mathbf{W}^{\text{MMSE-s}}]^{-1}$
$\mathbf{H}^{\text{MMSE-x}}$	(2.8)
$\mathbf{W}, \mathbf{W}(z)$	De-mixing system, estimation of $\mathbf{A}^{-1}$
$\mathbf{W}^{\text{MMSE-s}}$	(2.61)
$\mathbf{W}^{\text{MMSE-x}}$	$\triangleq [\mathbf{H}^{\text{MMSE-x}}]^{-1}$
$\mathbf{0}$	Vector or matrix containing zeros
$\mathbf{1}$	Vector or matrix containing ones
$\mathbf{F}$	DFT matrix (3.1)
$\mathbf{I}$	Identity matrix
$\mathbf{J}$	Exchange matrix (3.9)
$\tilde{\mathbf{J}}$	Circulant permutation matrix (3.26)
$\mathbf{J}_{\text{c}}$	Circulant-time-reversal matrix (3.6)
$\mathbf{P}$	Projection matrix
$\tilde{\mathbf{P}}$	Projection matrix (3.12)
$\mathbf{T}$	Block DFT matrix (3.7)

Mathematical Operators

$\bar{\mathbf{A}}$	Diagonal matrix (Section 3.1.4)
$\overline{\mathbf{A}}$	Block diagonal matrix (Section 3.1.7)
$\tilde{\mathbf{A}}$	Circulant matrix (Section 3.1.5)
$\widetilde{\mathbf{A}}$	Block circulant matrix (Section 3.1.8)
$\mathbf{a}^M$	Vector length
$\mathbf{A}^{M \times N}$	Matrix dimensions
$[\mathbf{a}]_m$	mth vector element (a_m)
$[\mathbf{A}]_{mn}$	mnth matrix element (a_{mn})
$[a_{mn}]$	Define a matrix $\mathbf{A}$ by it's mnth element
$\mathbf{A}_{(m,n)}$	Delete mth row and nth column of a matrix
$\mathbf{A}^*$	Complex conjugation $[a^*_{mn}]$
$\mathbf{A}^T$	Transposition $[a_{nm}]$
$\mathbf{A}^H$	Hermitian transposition, conjugate transposition $[a^*_{nm}]$
$\mathbf{A}^{-1}$	Matrix inverse (A.13)
$\mathbf{A}^{\#}$	Moore-Penrose pseudoinverse (A.20)
$\mathbf{A}^{((-1))}$	Elementwise inversion of a matrix $[a^{-1}_{mn}]$
$\mathcal{C}(\mathbf{a})$	Generate circulant matrix (3.18)
$\mathcal{C}^{-1}(\tilde{\mathbf{A}})$	First column vector of a circulan matrix $\tilde{\mathbf{A}}$ (3.19)
$\chi\{\mathbf{A}\}$	Matrix condition number, $\chi\{\mathbf{A}\} \triangleq \|\mathbf{A}\|_2 / \|\mathbf{A}^{-1}\|_2$
$\mathrm{adj}\,(\mathbf{A})$	Adjoint of a matrix (A.14)
$\mathrm{ddiag}\,(\mathbf{A})$	Set off-diagonal elements of $\mathbf{A}$ to zero (B.1)
$\det(\mathbf{A})$	Determinant of a matrix
$\mathrm{diag}\,(\mathbf{A})$	Vector with diagonal elements of matrix
$\mathrm{diag}\,[\mathbf{a}]$	Generate diagonal matrix
$\mathrm{off}\,(\mathbf{A})$	Set diagonal elements of $\mathbf{A}$ to zero (B.1)
$\mathrm{rank}\,(\mathbf{A})$	Rank of a matrix (Appendix A.7)
$\mathrm{tr}\,(\mathbf{A})$	Trace of a matrix (Appendix B)
$\max(.)$	Maximum
$\min(.)$	Minimum

$\lceil . \rceil$	Round to next higher integer
$\lfloor . \rfloor$	Round to next lower integer
$< . >_{a,b}$	Generalized remainder (D.2)
$< . >_{C}$	Symmetric remainder (D.4)
$\mathrm{sign}\,(.)$	Sign funktion
$\lvert . \rvert$	Absolute value
$\langle .,. \rangle$	Inner product, scalar product (C.1)
$\langle .,. \rangle_{\mathcal{F}}$	Inner product of polynomial matrices (C.4), (C.7), (C.10)
$\lVert . \rVert$	Euklidian norm (vector), $\lVert . \rVert_2 = \sigma_1$ (matrix)
$\lVert . \rVert_p$	p-norm
$\lVert . \rVert_F$	Frobenius norm (C.3)
$\lVert . \rVert_{\mathcal{F}}$	Frobenius norm of polynomial matrices (C.18), (C.20), (C.21), (C.22)
$*$	Linear convolution (Section 3.2.1, 3.5.1)
$\circledast$	Circular convolution (Section 3.2.2, 3.5.1)
$\otimes$	Kronecker product [15, 46, 101]
$\oplus$	Direct sum [15]
$\odot$	Elementwise matrix multiplication (Hadamard product)
$\mathcal{P}_{a,b}\,(a(z))$	Polynomial projection operator (D.10)
$\mathcal{P}_{C}\,(a(z))$	Symmetric polynomial projection operator (D.11)
$\tilde{\mathcal{P}}_{a,b}\,(a(z))$	Circular polynomial projection operator (D.69)
$\tilde{\mathcal{P}}_{C}\,(a(z))$	Symmetric circular polynomial projection operator (D.70)
$D\,(.\lVert .)$	Kullback-Leibler divergence (A.3)
$E\,\{.\}$	Expectation
$\delta(.)$	Dirac impulse
$\delta[.]$	Kronecker delta
$p_{\mathbf{s}}(.)$	Probability density function (pdf) of $\mathbf{s}$

Bibliography

[1] S.-I. Amari. Natural gradient learning for over- and under-complete bases in ICA. *Neural Computation*, 11(8):1875–1883, November 1999.

[2] S.-I. Amari, T.-P. Chen, and A. Cichocki. Stability analysis of adaptive blind source separation. *Neural Networks*, 10(8):1345–1351, August 1997.

[3] S.-I. Amari and A. Cichocki. Adaptive blind signal processing—neural network approaches. *Proceedings of the IEEE*, 86(10):2026–2048, October 1998.

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

[5] S.-I. Amari, S. C. Douglas, A. Cichocki, and H. H. Yang. Multichannel blind deconvolution and equalization using the natural gradient. In *IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications*, pages 101–104, Paris, France, April 16–18 1997.

[6] S.-I. Amari, S. C. Douglas, A. Cichocki, and H. H. Yang. Novel on-line adaptive learning algorithms for blind deconvolution using the natural gradient approach. In *Proc. 11th IFAC Symposium on System Identification (SYSID-97)*, pages 1057–1062, Kitakyushu, Japan, July 8–11 1997.

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

[8] S. Bellini. Bussgang techniques for blind equalization. In *IEEE Global Telecommunications Conference*, pages 1634–1640, Houston, TX, December 1–4 1986.

[9] S. Bellini. Bussgang techniques for blind deconvolution and equalization. In S. Haykin, editor, *Blind Deconvolution*, pages 8–59. Prentice Hall, 1994.

[10] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, 45(2):434–444, February 1997.

[11] A. Benveniste and M. Goursat. Blind equalizers. *IEEE Transactions on Computers*, COM-32(8):871–883, August 1984.

[12] A. Benveniste, M. Goursat, and G. Ruget. Robust identification of a nonminimum phase system: Blind adjustment of a linear equalizer in data communications. *IEEE Transactions on Automatic Control*, AC-25(3):385–399, June 1980.

[13] R. E. Blahut. *Fast Algorithms for Digital Signal Processing*. Addison-Wesley, 1985.

[14] D. H. Brandwood. A complex gradient operator and its application in adaptive array theory. *Proceedings of the IEE*, 130(1):11–16, February 1983.

[15] J. W. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, 25(9):772–781, September 1978.

[16] J.-F. Cardoso. Blind signal separation: Statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025, October 1998.

[17] J.-F. Cardoso and B. H. Laheld. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing*, 44(12):3017–3030, December 1996.

[18] G. A. Clark, S. K. Mitra, and S. R. Parker. Block implementation of adaptive digital filters. *IEEE Transactions on Circuits and Systems*, CAS-28(6):584–592, June 1981.

[19] G. A. Clark, S. R. Parker, and S. K. Mitra. A unified approach to time- and frequency-domain realization of FIR adaptive digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-31(5):1073–83, October 1983.

[20] P. M. Clarkson. *Optimal and Adaptive Signal Processing*. CRC Press, 1993.

[21] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, April 1994.

[22] P. Comon, C. Jutten, and J. Hérault. Blind separation of sources, part II: Problems statement. *Signal Processing*, 24(1):11–20, July 1991.

[23] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[24] S. Cruces, A. Cichocki, and L. Castedo. An iterative inversion method for blind source separation. In *Proc. First International Conference on Independent Component Analysis and Blind Signal Separation ICA'99*, pages 307–312, Aussois, France, January 11–15, 1999.

[25] P. J. Davis. *Circulant Matrices*. John Wiley & Sons, 1979.

[26] D. Donoho. On minimum entropy deconvolution. *Applied Time Series Analysis II*, pages 565–608, 1981.

[27] S. C. Douglas. Equivariant adaptive selective transmission. *IEEE Transactions on Signal Processing*, 47(5):1223–1231, May 1999.

[28] S. C. Douglas. Self-stabilized gradient algorithms for blind source separation with orthogonality constraints. *IEEE Transactions on Neural Networks*, 11(6):1490–1497, November 2000.

[29] S. C. Douglas and S.-I. Amari. Natural-gradient adaptation. In S. Haykin, editor, *Unsupervised Adaptive Filtering, Volume I: Blind Source Separation*, pages 13–61. John Wiley & Sons, 2000.

[30] S. C. Douglas, S.-I. Amari, and S.-Y. Kung. On gradient adaptation with unit-norm constraints. *IEEE Transactions on Signal Processing*, 48(6):1843–1847, June 2000.

[31] S. C. Douglas and A. Cichocki. Neural networks for blind decorrelation of signals. *IEEE Transactions on Signal Processing*, 45(11):2829–2841, November 1997.

[32] S. C. Douglas, A. Cichocki, and S. Amari. Multichannel blind separation and deconvolution of sources with arbitrary distributions. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 436–445, Almelia Island Plantation, FL, September 1997.

[33] S. C. Douglas and S. Haykin. On the relationship between blind deconvolution and blind source separation. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, volume II, pages 1591–1595, Pacific Grove, CA, November 1997.

[34] S. C. Douglas and S. Haykin. Relationships between blind deconvolution and blind source separation. In S. Haykin, editor, *Unsupervised Adaptive Filtering, Volume II: Blind Deconvolution*, pages 113–145. John Wiley & Sons, 2000.

[35] S. C. Douglas and S.-Y. Kung. Gradient adaptive algorithms for contrast-based blind deconvolution. *Journal of VLSI Signal Processing*, 26:47–60, 2000.

[36] E. R. Ferrara. Fast implementations of LMS adaptive filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-28(4):474–475, August 1980.

[37] E. R. Ferrara Jr. Frequency-domain adaptive filtering. In C.F.N. Cowan and P.M. Grant, editors, *Adaptive Filters*, pages 145–179. Prentice-Hall, 1985.

[38] N. Fliege. *Multirate Digital Signal Processing*. John Wiley & Sons, 1995.

[39] S. L. Gay and J. Benesty, editors. *Acoustic Signal Processing for Telecommunication*. Kluwer Academic Publishers, 2000.

[40] M. Girolami and C. Fyfe. Negentropy and kurtosis as projection pursuit indices provide generalised ICA algorithms. In *Advances in Neural Information Processing Systems*, Aspen, CO, Dec 7, 1996.

[41] D. N. Godard. Self-recovering equalization and carrier tracking in two-dimensional data communication systems. *IEEE Transactions on Communications*, COM-28(11):1867–1875, November 1980.

[42] R. Godfrey and F. Rocca. Zero memory non-linear deconvolution. *Geophysical Prospecting*, 29:189–228, 1981.

[43] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 2nd edition, 1993.

[44] G. C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control*. Prentice Hall, 1984.

[45] A. Gorokhov and P. Loubaton. Semi-blind second order identification of convolutive channels. In *International Conference on Acoustics, Speech & Signal Processing*, pages 3905–3908, Munich, Germany, April 21–24, 1997.

[46] A. Graham. *Kronecker Products and Matrix Calculus with Applications*. Ellis Horwood Limited, 1981.

[47] D. Graupe. *Identification of Systems*. Robert E. Krieger Publishing Company, 1975.

[48] R. M. Gray. *Toeplitz and Circulant Matrices: A review*. Stanford Electron. Lab., Tech. Rep. 6502-1, June 1971.

[49] R. M. Gray. On the asymptotic eigenvalue distribution of Toeplitz matrices. *IEEE Transactions on Information Theory*, pages 725–730, November 1972.

[50] W. W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, June 1989.

[51] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 3rd edition, 1996.

[52] S. Haykin, editor. *Unsupervised Adaptive Filtering, Volume I, Blind Source Separation*. John Wiley & Sons, 2000.

[53] S. Haykin, editor. *Unsupervised Adaptive Filtering, Volume II, Blind Deconvolution*. John Wiley & Sons, 2000.

[54] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[55] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[56] S. Ikeda and N. Murata. A method of blind separation based on temporal structure of signals. In *Proceedings of 1998 International Conference on Neural Information Processing (ICONIP'98)*, pages 737–742, October 1998.

[57] C. R. Johnson, P. Schniter, I. Fijalkow, L. Tong, J. D. Behm, M. G. Larimore, D. R. Brown, R. A. Casas, T. J. Endres, S. Lambotharan, A. Touzni, H. H. Zeng, M. Green, and J. R. Treichler. The core of FSE-CMA behavior theory. In S. Haykin, editor, *Unsupervised Adaptive*

Filtering, Volume II: Blind Deconvolution, pages 13–112. John Wiley & Sons, 2000.

[58] D. H. Johnson and D. E. Dudgeon. *Array Signal Processing — Concepts and Techniques*. Prentice Hall, 1993.

[59] M. Joho and H. Mathis. Performance comparison of combined blind/non-blind source separation algorithms. In *Proc. International Conference on Independent Component Analysis and Blind Signal Separation*, pages 139–142, Aussois, France, January 11–15, 1999.

[60] M. Joho, H. Mathis, and R. H. Lambert. Overdetermined blind source separation: Using more sensors than source signals in a noisy mixture. In *Proc. International Conference on Independent Component Analysis and Blind Signal Separation*, pages 81–86, Helsinki, Finland, June 19–22, 2000.

[61] M. Joho, H. Mathis, and G. S. Moschytz. Combined blind/non-blind source separation based on the natural gradient. submitted.

[62] M. Joho and G. S. Moschytz. Connecting partitioned frequency-domain filters in parallel or in cascade. *IEEE Transactions on Circuits and Systems–II*, 47(8):685–698, August 2000.

[63] D. L. Jones. A new method for blind source separation of nonstationary signals. In *International Conference on Acoustics, Speech & Signal Processing*, volume 5, pages 2893–2896, Phoenix, AZ, May 15–19, 1999.

[64] C. Jutten and J. Hérault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10, July 1991.

[65] T. Kailath. *Linear Systems*. Prentice Hall, 1980.

[66] W. Kellermann. Echoes and noise with seamless acoustic man-machine interfaces—the challenge persists. In *International Workshop on Acoustic Echo and Noise Control*, pages 1–7, Pocono Manor, Pennsylvania, USA, Sept. 27–30 1999.

[67] M. Kendall and A. Stuart. *The Advanced Theory of Statistics*, volume 1, Distribution Theory. Charles Griffin & Comp. Ltd., 1977.

[68] E. Kreyszig. *Introductory Functional Analysis with Applications*. John Wiley & Sons, 1978.

[69] R. H. Lambert. *Multichannel Blind Deconvolution: FIR Matrix Algebra and Separation of Multipath Mixtures*. PhD thesis, University of Southern California, 1996.

[70] R. H. Lambert. Difficulty measures and figures of merit for source separation. In *Proc. International Conference on Independent Component Analysis and Blind Signal Separation*, pages 133–138, Aussois, France, January 11–15, 1999.

[71] R. H. Lambert and C. L. Nikias. Blind deconvolution of multipath mixtures. In S. Haykin, editor, *Unsupervised Adaptive Filtering, Volume I: Blind Source Separation*, pages 377–436. John Wiley & Sons, 2000.

[72] T.-W. Lee. *Independent Component Analysis*. Kluwer Academic Publishers, 1998.

[73] T.-W. Lee, M. S. Lewicki, M. Girolami, and T. Sejnowski. Blind source separation of more sources than mixtures using overcomplete representations. *IEEE Signal Processing Letters*, 6(4):87–90, April 1999.

[74] M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):338–365, February 2000.

[75] L. Ljung. *System Identification*. Prentice Hall, 1987.

[76] R. W. Lucky. Techniques for adaptive equalization of digital communication systems. *Bell System Tech. J.*, 45:255–286, 1966.

[77] D. Mansour and A. H. Gray Jr. Unconstrained frequency-domain adaptive filter. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-30(5):726–734, October 1982.

[78] H. Mathis, M. Joho, and G. S. Moschytz. A simple threshold nonlinearity for blind signal separation. In *IEEE International Symposium on Circuits and Systems*, volume IV, pages 489–492, Geneva, Switzerland, May 28–31, 2000.

[79] H. Mathis, T. P. von Hoff, and M. Joho. Blind separation of mixed-kurtosis signals using an adaptive threshold nonlinearity. In *Proc. International Conference on Independent Component Analysis and Blind Signal Separation*, pages 221–226, Helsinki, Finland, June 19–22, 2000.

[80] L. Molgedey and H. Schuster. Separation of independent signals using time-delayed correlations. *Physical Review Letters*, 72(23):3634–3637, October 1994.

[81] D. R. Morgan and J. C. Thi. Delayless subband adaptive filter architecture. *IEEE Transactions on Signal Processing*, 43(8):1819–1830, August 1995.

[82] N. Murata and S. Ikeda. A on-line algorithm for blind source separation on speech signals. *Proceedings of 1998 International Symposium on Nonlinear Theory and its Applications (NOLTA'98)*, pages 923–926, September 1998.

[83] A. K. Nandi. *Blind Estimation Using Higher-Order Statistics*. Kluwer Academic Publishers, 1999.

[84] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice Hall, 1975.

[85] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 3rd edition, 1991.

[86] L. Parra and C. Spence. Convolutive blind source separation based on multiple decorrelation. In *IEEE Workshop on Neural Networks and Signal Processing*, Cambridge, UK, September 1998.

[87] L. Parra and C. Spence. Convolutive blind separation of non-stationary sources. *IEEE Transactions on Speech and Audio Processing*, 8(3):320–327, 2000.

[88] L. Parra and C. Spence. On-line convolutive source separation of non-stationary signals. *Journal of VLSI Signal Processing*, 26(1/2), August 2000.

[89] D. T. Pham, P. Garat, and C. Jutten. Separation of mixture of independent sources through a maximum likelihood approach. *Signal Processing VI: Theories and Applications*, pages 771–774, Oct. 28–31, 1992.

[90] R.-W. Liu, Ed. Special issue on blind system identification and estimation. *Proceedings of the IEEE*, 86(10), October 1998.

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

[92] D. W. E. Schobben. *Efficient Adaptive Multi-channel Concepts in Acoustics: Blind Signal Separation and Echo Cancellation*. PhD thesis, Technical University Eindhoven, 1999.

[93] D. W. E. Schobben and P. C. W. Sommen. A new blind signal separation algorithm based on second order statistics. In *IASTED International Conference Signal and Image Processing*, pages 1–6, Las Vegas, USA, October 27–31 1998.

[94] D. W. E. Schobben and P. C. W. Sommen. Transparent communication. In *IEEE Benelux Signal Processing Chapter Symposium*, pages 171–174, Leuven, Belgium, March 26–27 1998.

[95] O. Shalvi and E. Weinstein. New criteria for blind deconvolution of non-minimum phase systems (channels). *IEEE Transactions on Information Theory*, 36(2):312–321, March 1990.

[96] J. J. Shynk. Frequency-domain and multirate adaptive filtering. *IEEE Signal Processing Magazine*, pages 14–37, January 1992.

[97] P. C. W. Sommen. Partitioned frequency domain adaptive filters. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, pages 676–681, Pacific Grove, CA, November 1989.

[98] P. C. W. Sommen. *Adaptive Filtering Methods*. PhD thesis, Technical University Eindhoven, 1992.

[99] K. P. J. Soo. Multidelay block frequency domain adaptive filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(2):373–376, February 1990.

[100] B. Widrow S. D. Stearns. *Adaptive Signal Processing*. Prentice Hall, 1985.

[101] W.-H. Steeb. *Kronecker Product of Matrices and Applications*. BI Wissenschaftsverlag, 1991.

[102] C. W. Therrien. *Discrete Random Signals and Statistical Signal Processing*. Prentice Hall, 1992.

[103] L. Tong, , G. Xu, B. Hassibi, and T. Kailath. Blind channel identification based on second-order statistics: A frequency-domain approach. *IEEE Transactions on Information Theory*, 41(1):329–354, January 1995.

[104] L. Tong, , G. Xu, and T. Kailath. Blind channel identification based on second-order statistics: A time domain approach. *IEEE Transactions on Information Theory*, 40(2):340–349, March 1995.

[105] L. Tong, R.-W. Liu, V. C. Soon, and Y.-F. Huang. Indeterminacy and identifiability of blind identification. *IEEE Transactions on Circuits and Systems*, 38(5):499–509, May 1991.

[106] K. Torkkola. Blind separation of delayed and convolved sources. In S. Haykin, editor, *Unsupervised Adaptive Filtering, Volume I: Blind Source Separation*, pages 321–375. John Wiley & Sons, 2000.

[107] M. K. Tsatsanis and C. Kweon. Blind source separation of non-stationary sources using second-order statistics. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, volume II, pages 1574–1578, Pacific Grove, CA, November 1998.

[108] J. K. Tugnait, L. Tong, and Z. Ding. Single-user channel estimation and equalization. *IEEE Signal Processing Magazine*, 17(3):16–28, May 2000.

[109] T. P. von Hoff, A. G. Lindgren, and A. N. Kaelin. Transpose properties in the stability and performance of the classic adaptive algorithms for blind source separation and deconvolution. *Signal Processing*, 80(9):1807–1822, August 2000.

[110] A. T. Walden. Non-Gaussian reflectivity, entropy, and deconvolution. *Geophysics*, 50(12):2862–2888, December 1985.

[111] Z. Wang and G. B. Giannakis. Wireless multicarrier communications. *IEEE Signal Processing Magazine*, 17(3):29–48, May 2000.

[112] E. Weinstein, M. Feder, and A. V. Oppenheim. Multi-channel signal separation by decorrelation. *IEEE Transactions on Speech and Audio Processing*, 1(4):405–413, October 1993.

[113] B. Widrow, J. R. Glover Jr., J. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. D. Dong Jr., and R. C. Goodlin. Adaptive noise cancelling: Principles and applications. *Proceedings of the IEEE*, 63(12):1692–1719, December 1975.

[114] B. Widrow, J. McCool, and M. Ball. The complex LMS algorithm. *Proceedings of the IEEE*, pages 719–720, April 1975.

[115] S. Wyrsch. *Adaptive Subband Signal Processing for Hearing Instruments*. PhD thesis, ETH Zürich, 2000.

Curriculum Vitae

Marcel Joho
citizen of Bettwil AG, Switzerland
born October 8, 1967, in Zürich, Switzerland

1974 – 1979	Primary School, Endingen
1979 – 1983	Secondary School, Endingen
1983 – 1987	High School, Baden, Matura Typus C
1988 – 1993	Study in Electrical Engineering at ETH Zürich
1993	Dipl. El.-Ing. ETH degree (M.Sc. in EE)
1993 – 1999	Post-diploma study in Information Technology at ETH Zürich
1999	Post-diploma degree in Information Technology, ETH Zürich
1993 – 1995	Teaching Assistant at the Signal and Information Processing Laboratory, ETH Zürich
1995 – 2000	Research Assistant at the Signal and Information Processing Laboratory, ETH Zürich