ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΥΠΟΥΡΓΕΙΟ ΟΙΚΟΝΟΜΙΑΣ,
ΑΝΑΠΤΥΞΗΣ & ΤΟΥΡΙΣΜΟΥ

ΕΝΙΑΙΟΣ ΔΙΟΙΚΗΤΙΚΟΣ ΤΟΜΕΑΣ
ΕΙΔΙΚΗ ΓΡΑΜΜΑΤΕΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΤΟΜΕΑΚΩΝ Ε.Π. ΤΟΥ ΕΚΤ

ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Ε.Π. «ΑΝΑΠΤΥΞΗ ΑΝΘΡΩΠΙΝΟΥ ΔΥΝΑΜΙΚΟΥ, ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ»

Πίνακας με συνοδευτικές υποστηρικτικές πληροφορίες

Τίτλος πράξης	ΘΑΛΗΣ-ΕΚΠΑ-ΑΣΦΑΛΕΙΣ ΑΣΥΡΜΑΤΕΣ ΜΗ-ΓΡΑΜΜΙΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΣΤΟ ΦΥΣΙΚΟ ΕΠΙΠΕΔΟ
Κωδικός πράξης	380202
Τίτλος υποέργου	ΑΣΦΑΛΕΙΣ ΑΣΥΡΜΑΤΕΣ ΜΗ-ΓΡΑΜΜΙΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΣΤΟ ΦΥΣΙΚΟ ΕΠΙΠΕΔΟ
Τίτλος μελέτης (προσδορίστε από μελέτες, εκπαιδευτικό υλικό, εμπειρογνωμοσύνες, αξιολογήσεις κλπ)	D2.3.1 ΚΕΦΑΛΑΙΟ ΒΙΒΛΙΟΥ: SPARSE NONLINEAR MIMO FILTERING AND IDENTIFICATION
Αριθμός συνημμένων αρχείων μελέτης	1
Τίτλοι υποπαραδοτέων μελέτης (σε περίπτωση που υπάρχουν)	
Ημερομηνία εκπόνησης της μελέτης	01/02/2012 - 30/04/2013
Τελικός δικαιούχος	ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Φορέας υλοποίησης	ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Ανάδοχος	ΚΑΘΗΓΗΤΗΣ ΝΙΚΟΛΑΟΣ ΚΑΛΟΥΠΤΣΙΔΗΣ

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

Επιχειρησιακό Πρόγραμμα
Ανάπτυξη Ανθρώπινου Δυναμικού,
Εκπαίδευση και Διά Βίου Μάθηση
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

MIS 380202

Secure **WI**reless **N**onlinear **COM**-munications *at the Physical Layer*

Deliverable 2.3.1

Sparse Nonlinear MIMO Filtering and Identification

G. Mileounis and N. Kalouptsidis

European Union
European Social Fund

OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION & RELIGIOUS AFFAIRS, CULTURE & SPORTS
MANAGING AUTHORITY

NSRF 2007-2013
programme for development
EUROPEAN SOCIAL FUND

Co-financed by Greece and the European Union

date:
March 11, 2014

version:
1.00

Sparse Nonlinear MIMO Filtering and Identification

G. Mileounis and N. Kalouptsidis

Abstract In this chapter system identification algorithms for sparse nonlinear multi input multi output (MIMO) systems are developed. These algorithms are potentially useful in a variety of application areas including digital transmission systems incorporating power amplifier(s) along with multiple antennas, cognitive processing, adaptive control of nonlinear multivariable systems, and multivariable biological systems. Sparsity is a key constraint imposed on the model. The presence of sparsity is often dictated by physical considerations as in wireless fading channel–estimation. In other cases it appears as a pragmatic modelling approach that seeks to cope with the curse of dimensionality, particularly acute in nonlinear systems like Volterra type series.

Three identification approaches are discussed: conventional identification based on both input and output samples, semi–blind identification placing emphasis on minimal input resources and blind identification whereby only output samples are available plus a–priori information on input characteristics. Based on this taxonomy a variety of algorithms, existing and new, are studied and evaluated by simulations.

1 Introduction

System nonlinearities are present in many practical situations and remedies based on linear approximations often degrade system performance. A popular model that captures system nonlinearities is Volterra series [69, 71, 77]. This model is employed in communications, digital magnetic recording, physiological systems, control of multivariable systems, etc. Volterra series constitute a class of polynomial models that can be regarded as a Taylor series with memory. An attractive feature of this model is that the unknown parameters enter linearly at the output. On the other

G. Mileounis and N. Kalouptsidis are with the Department of Informatics and Telecommunications, Division of Communications and Signal Processing, University of Athens, Panepistimiopolis, 157 84 Ilisia, Greece (e-mail: gmil@di.uoa.gr; kalou@di.uoa.gr)

hand, the number of terms increases exponentially with the order and memory of the model.

Most of the work reported in the literature focuses on modelling and identification of single input single output (SISO) Volterra systems. When the underlying nonlinear system is a MIMO system, the resulting model is more complicated and has received little attention. MIMO models are addressed in this chapter. Nonlinear MIMO systems involve a large number of parameters to be estimated, which increases exponentially with the order, the memory and the number of inputs. Therefore, there is a strong need to reduce complexity by considering those terms that strongly contribute to the outputs. This leads naturally to a sparse approximation of the underlying nonlinear MIMO system. Identification of sparse nonlinear MIMO systems is approached under three different settings: conventional, semi–blind and blind. Blind methods identify the unknown system parameters merely based on the output signals. On the other hand, conventional and semi–blind methods, require a training or a pilot sequence.

The objective of this chapter is twofold. First, it extends existing algorithms for adaptive filtering of SISO models to the MIMO case and demonstrates their applicability to nonlinear MIMO systems. Secondly, it presents new algorithms for blind and semi–blind identification of nonlinear MIMO systems excited by finite alphabet inputs. The chapter is divided into four sections. The sparse nonlinear MIMO models under consideration are presented in Section 2. Adaptive filters for sparse MIMO systems are discussed in Section 3. Then, algorithms for blind and semi–blind identification are addressed in Section 4. Finally, summary and future work are discussed in Section 5.

2 System Model

MIMO polynomial systems form the basic class of models we shall be working with. These finitely parametrizable recursive structures are defined next. First the basic notation from SISO Volterra series is reviewed. Then MIMO extensions are considered and some special cases of interest are introduced. Finally, various applications which employ MIMO Volterra models are briefly reviewed.

Volterra series constitute a popular model for the description of nonlinear behaviour [69, 71]. A SISO discrete–time Volterra model has the following form

$$y(n) = \sum_{p=1}^{\infty} \sum_{\tau_1=-\infty}^{\infty} \cdots \sum_{\tau_p=-\infty}^{\infty} h_p(\tau_1, \ldots, \tau_p) \left[\prod_{i=1}^{p} x(n-\tau_i) \right]. \tag{1}$$

Each output is formed by weighting the input shifted samples $x(n-\tau_i)$ and their products. The weights $h_p(\tau_1, \ldots, \tau_p)$ constitute the *Volterra kernels* of order p. Well possessedness conditions ensuring that inputs give rise to well defined outputs are given in [51, 13]. If only a finite number of nonlinearities enters Eq. (1), the resulting expression defines a finite Volterra system. Suppose the kernels of a finite Volterra

system are causal and absolutely summable. Then Eq. (1) defines a bounded input bounded output (BIBO) stable system and can be approximated by the polynomial system

$$y(n) = \sum_{p=1}^{P} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} h_p(\tau_1, \ldots, \tau_p) \left[\prod_{i=1}^{p} x(n-\tau_i) \right]. \tag{2}$$

Eq. (2) is parametrized by the finite Volterra kernels and has finite memory M. A more general result established by Boyd and Chua [14, 13] states that any shift invariant causal BIBO stable system with *fading memory* can be approximated by Eq. (2). The fading memory is a continuity property with respect to a weighted norm which penalizes the remote past in the formation of the current output. The reader may consult [13, 14, 51] for more details.

A key feature of Eq. (2) is that it is *linear in the parameters*. For estimation purposes it is useful to write Eq. (2) in matrix form using Kronecker products [15]. Indeed, let $\mathbf{x}(n) = [x(n), x(n-1), \cdots, x(n-M)]^T$ (the superscript T denotes the transpose operation) and the pth–order Kronecker power

$$\mathbf{x}_p(n) = \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{p \text{ times}}, \quad p = 2, \ldots, P.$$

The Kronecker power contains all pth–order products of the input. Likewise $\mathbf{h} = \left[\mathbf{h}_1(\cdot), \cdots, \mathbf{h}_p(\cdot)\right]^T$ is obtained by treating the p–dimensional kernel as a M^p column vector. We rewrite Eq. (2) as follows

$$y(n) = \left[\mathbf{x}^T(n) \mathbf{x}_2^T(n) \cdots \mathbf{x}_p^T(n)\right] \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_p \end{bmatrix} = \mathbf{x}^T(n)\mathbf{h}. \tag{3}$$

Collecting n successive output samples from the above equation into the vector $\mathbf{y}(n) = [y(1), \ldots, y(n)]$ results in the following system of linear equations:

$$\mathbf{y}(n) = \mathbf{X}(n)\mathbf{h}$$

when

$$\mathbf{X}(n) = \left[\mathbf{x}^T(1), \ldots, \mathbf{x}^T(n)\right]^T.$$

From a practical viewpoint, Volterra models of order higher than three are rarely considered. This is due to the fact that the number of parameters ($\sum_{p=1}^{P} M^p$) involved in the model of Eq. (2) grows exponentially as a function of the memory size and the order of nonlinearity. To cope with this complexity several sub–families of Eq. (2) have been considered, most notable Wiener, Hammerstein and Wiener–Hammerstein models. In all cases the universal approximation capability is lost. A Wiener system is the cascade of a linear filter followed by a static nonlinearity. If we approximate the static nonlinearity with its Taylor expansion up to a certain order, we obtain the following expression for the output of the Wiener system

$$y(n) = \sum_{p=1}^{P} \left[\sum_{\tau=0}^{M} h_p(\tau) x(n-\tau) \right]^p . \tag{4}$$

The Hammerstein system (or memory polynomial) is composed of a memoryless nonlinearity (a Taylor approximation of the static nonlinearity is employed) followed by a linear filter, and has the following form

$$y(n) = \sum_{p=1}^{P} \sum_{\tau=0}^{M} h_p(\tau) x^p(n-\tau). \tag{5}$$

A Wiener–Hammerstein or sandwich model is composed of a memoryless nonlinearity sandwiched between two linear filters with impulse responses $h(\cdot)$ and $g(\cdot)$ and is defined as

$$y(n) = \sum_{p=1}^{P} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} \sum_{k=0}^{M_{h_p}+M_{g_p}} g_p(k) \prod_{l=1}^{p} h_p(\tau_l - k) x(n-\tau_l). \tag{6}$$

The above models have been employed in a wide range of applications including: satellite, telephone channels, mobile cellular communications, wireless LAN devices, radio and TV stations, digital magnetic systems and others [8, 32, 71, 77, 80].

2.1 Nonlinear MIMO systems with universal approximation capability

The discussion of the previous subsection is next extended to MIMO nonlinear systems. Attention is limited to MIMO polynomial systems. These are finitely parametrizable structures that naturally extend Eq. (2) and preserve a universal approximation capability over a broad class of multivariables systems. We start our discussion by considering cases where either the MIMO system has a single input or a single output. In the end, sparsity is imposed in order to reduce the number of unknown parameters.

The input–output relationship of nonlinear single input multiple output (SIMO) system is

$$y_r(n) = \sum_{p=1}^{p} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} h_p^{(r)}(\tau_1, \ldots, \tau_p) \prod_{i=1}^{p} x(n-\tau_i) \tag{7}$$

where $y_r(n)$ is the output associated with the rth output signal and $h_p^{(r)}(\tau_1, \ldots, \tau_p)$ is the *pth–order Volterra kernel* of the rth output. The difference between Eq. (2) and Eq. (7) is that a distinct kernel $h_p^{(r)}(\tau_1, \ldots, \tau_p)$ is associated with each output signal $y_r(n)$. This is illustrated in Fig. 1. SIMO systems can be obtained by oversampling the output signal of a SISO system at a sufficiently high rate and demultiplexing the samples [44].

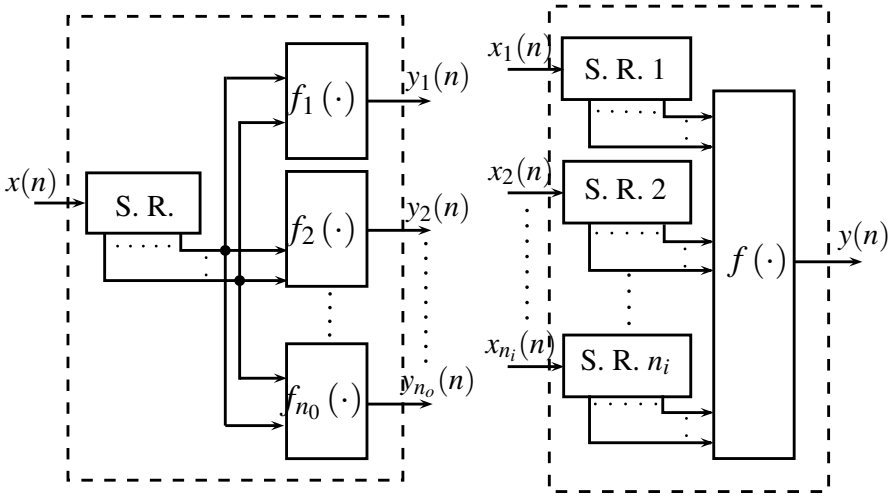


Fig. 1 SIMO and MISO polynomial systems (SR denotes a shift register)

A multiple input single output (MISO) system comprises n_i input signals and a single output. The input–output of a MISO system has the form

$$y(n) = \sum_{p=1}^{P} \sum_{t=1}^{n_i} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} h_p(\tau_1, \ldots, \tau_p) \prod_{i=1}^{p} x_t(n-\tau_i) \tag{8}$$

where $x_t(n)$ is the tth input signal ($1 \leq t \leq n_i$). A shift register (SR) is associated with each input. The contents of all registers are then converted into the output by means of a feed forward polynomial as shown in Fig. 1.

The general MIMO case is readily construed from the above special cases. A MIMO finite support Volterra system with n_i inputs and n_o outputs has the following form:

$$y_r(n) = f_r(x_1(n), x_1(n-1), \ldots, x_1(n-M), \cdots, x_{n_i}(n), x_{n_i}(n-1), \ldots, x_{n_i}(n-M)), \quad r = 1, \ldots, n_o. \tag{9}$$

Each output $y_r(n)$ is obtained by a polynomial combination of the n_i inputs and their shifts. The parameter M specifies the memory of the n_i registers associated with each input. The MIMO finite support Volterra architecture is depicted in Fig. 2. This model is capable of capturing nonlinear effects resulting from any product combinations of the n_i inputs and their shifts. Expanding $f_r(\cdot)$ as a polynomial of degree P gives rise to the nonlinear MIMO Volterra model with n_i inputs and n_o outputs defined as

$$y_r(n) = \sum_{p=1}^{P} \sum_{t_1=1}^{n_i} \cdots \sum_{t_p=1}^{n_i} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} h_p^{(r,t_1 \cdots t_p)}(\tau_1, \ldots, \tau_p) \prod_{i=1}^{p} x_{t_i}(n-\tau_i) \tag{10}$$

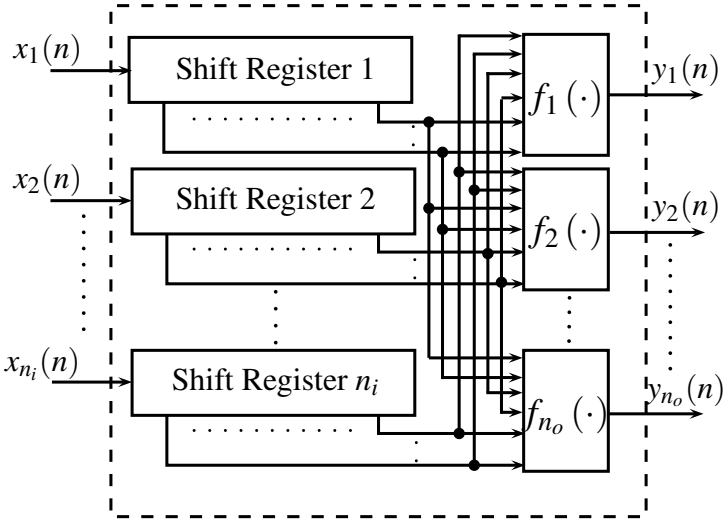


Fig. 2 A nonlinear MIMO Volterra

where $h_p^{(r,t_1\cdots t_p)}(\tau_1,\ldots,\tau_p)$ is the pth order Volterra kernel associated with the rth output and the $(t_1\cdots t_p)$ inputs. In this case, the Volterra kernels have multidimensional indices $(r,t_1\cdots t_p)$.

The above expressions are made complicated by the presence of multiple summations. Kronecker products alleviate this problem. Let

$$\bar{\mathbf{x}}(n)=[x_1(n),x_1(n-1),\ldots,x_1(n-M),\cdots,x_{n_i}(n),x_{n_i}(n-1),\ldots,x_{n_i}(n-M)]^T$$

and hence the nonlinear input vector is given by

$$\mathbf{x}(n)=[\bar{\mathbf{x}}(n),\bar{\mathbf{x}}_2(n),\cdots,\bar{\mathbf{x}}_p(n)]^T. \tag{11}$$

Then Eq. (10) takes the form:

$$\mathbf{y}(n)=\mathbf{H}\mathbf{x}(n) \tag{12}$$

where $\mathbf{y}(n)=[y_1(n),\ldots,y_{n_o}(n)]^T$ is the output vector, and the system matrix is $\mathbf{H}=[\mathbf{h}_{1:},\ldots,\mathbf{h}_{n_o:}]^T$, with $\mathbf{h}_{n_o:}$ containing all the Volterra kernels associated with the rth output. In this case the parameter matrix contains

$$\sum_{i=1}^{p}(n_i\times M)^p$$

parameters. The MIMO polynomial family of Eq. (9) has a universal approximation capability in the following sense: every nonlinear system with more than one inputs and outputs that is causal, shift invariant, bounded input bounded output stable and has fading memory can be approximated by a MIMO polynomial system of the form given in Eq (10). This assertion is established if the same statement is proved for

MISO systems. The latter follows with straightforward modifications of the proof for the SISO case.

2.1.1 Sparsity aware Volterra kernels

A major obstacle in using Volterra series in practical applications is the exponential growth of the model parameters (as a function of the order, the memory length of the systems and the number of inputs). Thus models of order $p > 3$ and memory length $M > 5$, translate into increased computational complexity cost and data requirements for identification purposes. For this reason, parsimonious, reduced order alternatives become relevant.

Sparse representations provide a viable alternative. The parameter matrix $\mathbf{H}$ in Eq. (12) is s–sparse if the number of non–zero elements is less than s, *i.e.*

$$\|\text{vec}[\mathbf{H}]\|_{\ell_0} = \{\#(i,j) : H_{ij} \neq 0\} \leq s.$$

2.2 Special classes of MIMO Nonlinear Systems

In this section, some special classes of MIMO Volterra systems are studied. We start with a simplified version of the MIMO Volterra model. Then structured nonlinear models like Wiener, Hammerstein and Wiener–Hammerstein are extended to the MIMO case. These models are formed by the cascade connection of linear MIMO filters and MIMO static nonlinearities.

2.2.1 Parallel cascade MIMO Volterra

In MIMO systems the signals from the n_i inputs interact with each other and the resulting mixture is received at each output. A special case of Eq. (10) results when the MIMO system obtains from the parallel connection of SISO systems, where each SISO system is often referred to as a path or parallel system. If the path between each input and each output is modelled as a Volterra system, then the rth output is expressed as follows

$$y_r(n) = \sum_{p=1}^{P} \sum_{t=1}^{n_i} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} h_p^{(r,t)}(\tau_1, \ldots, \tau_p) \prod_{i=1}^{p} x_t(n-\tau_i) \tag{13}$$

where $h_p^{(r,t)}(\tau_1, \ldots, \tau_p)$ is the *pth–order Volterra kernel* between the tth input and the rth output for all $t = 1, \ldots, n_i$ and $r = 1, \ldots, n_o$. The above model does not allow product combinations along different inputs. Instead each input is nonlinearly transformed and then all different inputs are linearly mixed. Such a model can be considered as a parallel cascade of n_i SIMO Volterra models.

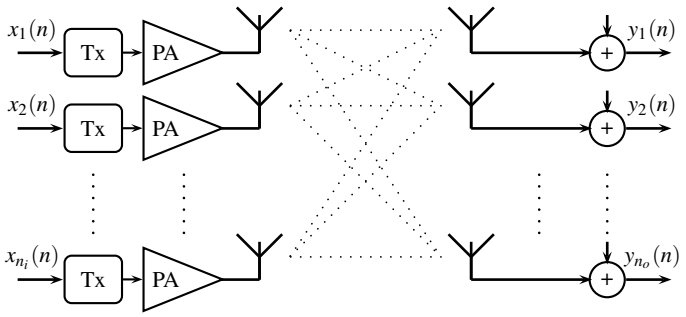


Fig. 3 An example of a parallel cascade MIMO Volterra channel

Eq. (13) can be written in a form identical to that of Eq. (12). Define the tth input regressor vector as

$$\mathbf{x}^{(t)}(n) = [x^{(t)}(n), x^{(t)}(n-1), \ldots, x^{(t)}(n-M)]^T.$$

Then the linearly mixed input vector takes the form:

$$\mathbf{x}(n) = [\mathbf{x}_1^{(1)}(n), \mathbf{x}_2^{(1)}(n), \ldots, \mathbf{x}_p^{(1)}(n), \cdots, \mathbf{x}_1^{(n_i)}(n), \mathbf{x}_2^{(n_i)}(n), \ldots, \mathbf{x}_p^{(n_i)}(n)]^T.$$

The total number of parameters of the above linearly mixed model is

$$n_i \sum_{i=1}^{p} M^p$$

and is considerably reduced when compared to the general case.

The linearly mixed model finds application in nonlinear communications. Communication nonlinearities can be categorized into the following three types: transmitter nonlinearity (due to nonlinearity in amplifiers), inherent physical channel nonlinearity, and receiver nonlinearity (e.g., due to nonlinear filtering). The power amplifier (PA) (which is located at the transmitter) constitutes the main source of nonlinearity. In a system equipped with multiple transmit antennas, each transmitter amplifies the signal. Amplifiers often operate near saturation to achieve power efficiency. In those cases they introduce nolinearities which cause interference and reduce spectral efficiency. At the receiver end, each antenna receives a linear superposition of all transmitted signals, as illustrated in Fig. 3. It should be pointed out that the nonlinear effects are applied to each input signal individually prior to mixing the transmitted signals.

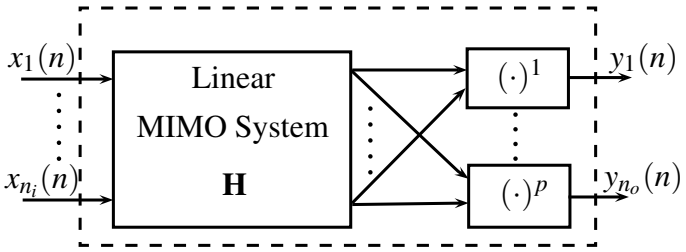


Fig. 4 A MIMO Wiener System

2.2.2 Block–structured classes of nonlinear MIMO systems

The *MIMO Wiener model* is shown in Fig. 4. It consists of a linear MIMO system in cascade with a polynomial nonlinearity for each output. The output is given by

$$y_r(n) = \sum_{p=1}^{P} \sum_{t_1=1}^{n_i} \cdots \sum_{t_p=1}^{n_i} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} \prod_{i=1}^{p} h_p^{(r,t_i)}(\tau_i) x_{t_i}(n-\tau_i). \tag{14}$$

This model is a special subclass of the MIMO Volterra series model. The relationship between the pth–order Volterra kernel and pth–order Wiener kernel is

$$h_p^{(rt_1\cdots t_p)}(\tau_1,\ldots,\tau_p) = \prod_{i=1}^{p} h_p^{(r,t_i)}(\tau_i).$$

Thus a MIMO Wiener model is equivalent to a MIMO Volterra system with separable kernels. The MIMO Hammerstein model is one of the simplest and most popular subclasses of MIMO Volterra models. As the diagram of Fig. 5 shows, the MIMO Hammerstein is a cascade connection of a static polynomial nonlinearity for each input connected in series by a linear MIMO system. It consists of the same building blocks as the Wiener model, but connected in reverse order. It has the following form:

$$y_r(n) = \sum_{p=1}^{P} \sum_{t=1}^{n_i} \sum_{\tau=0}^{M} h_p^{(r,t)}(\tau) x^p(n-\tau_i). \tag{15}$$

The pth–order Volterra kernel of a Hammerstein model is given by

$$h_p^{(rt_1\cdots t_p)}(\tau_1,\ldots,\tau_p) = h_p(\tau_1)\delta(\tau_2-\tau_1)\cdots\delta(\tau_p-\tau_1)\delta(t_2-t_1)\cdots\delta(t_p-t_1) \tag{16}$$

A Hammerstein system prohibits product interactions between different inputs and hence corresponds to a diagonal MIMO Volterra model.

We finally consider the case where the MIMO Volterra kernels have factorable form:

$$h_p^{(rt_1\cdots t_p)}(\tau_1,\ldots,\tau_p) = \sum_{k=0}^{M_h+M_g} g_p^r(k) \prod_{i=1}^{p} h_p^{t_i}(\tau_i - k)$$

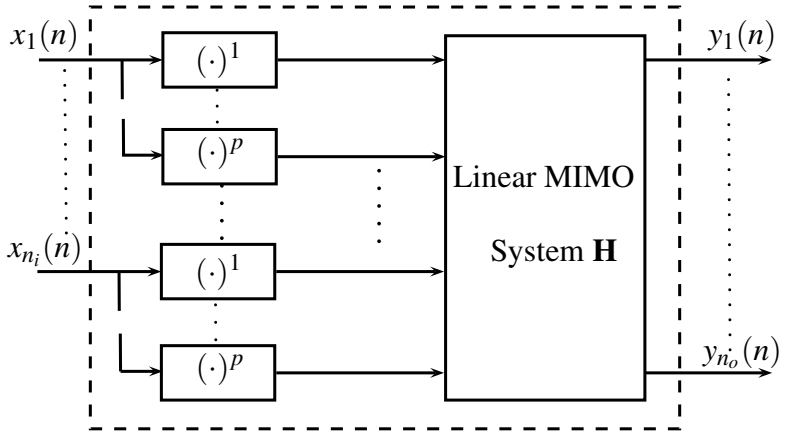


Fig. 5 A MIMO Hammerstein system

Substituting the above form into Eq. (10), we obtain:

$$y_r(n) = \sum_{p=1}^{P} \sum_{t_1=1}^{n_i} \cdots \sum_{t_p=1}^{n_i} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} \sum_{k=0}^{M_h+M_g} g_p^r(k) \prod_{i=1}^{p} h_p^{t_i}(\tau_i - k) x_{t_i}(n-\tau_i). \tag{17}$$

The pth–order kernel corresponds to a cascade connection of a linear MIMO system followed by a memoryless nonlinearity followed by another linear MIMO system and is known as *MIMO Wiener–Hammerstein* or sandwich model. In its simplest form a MIMO Wiener–Hammerstein system has a sandwiched structure with a single input single output static nonlinearity placed between a MISO and a SIMO linear systems. In the general case, illustrated in Fig. 6, the two linear filters can have arbitrary input and output dimensions. Compatibility is secured by proper dimensioning of the MIMO static nonlinearity. The Wiener–Hammerstein has been widely employed in satellite transmission, where both the earth station and the satellite repeater employ (nonlinear) power amplifiers. In such cases the signal bandwidth is very carefully defined depending on the application so that the output signal contains only spectral components near the carrier frequency ω_c. This leads to the *MIMO baseband Wiener–Hammerstein* system [8, Ch. 14], given by

$$\begin{aligned} y_r(n) = \sum_{p=1}^{\lfloor \frac{P-1}{2} \rfloor} \sum_{t_1=1}^{n_i} \cdots \sum_{t_{2p+1}=1}^{n_i} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_{2p+1}=0}^{M} \sum_{k=0}^{M_h+M_g} g_p^r(k) \prod_{i=1}^{p} \\ \times \prod_{i=1}^{p+1} h_{2p+1}^{t_i}(\tau_i - k) x_{t_i}(n-\tau_i) \prod_{j=p+2}^{2p+1} h_{2p+1}^{t_j}(\tau_j - k) x_{t_j}^*(n-\tau_j) \end{aligned} \tag{18}$$

where $\lfloor \cdot \rfloor$ denote the floor operation. The above representation only considers odd–order powers with one more unconjugated input than conjugated input. This way the output does not create spectral components outside the frequency band of interest.

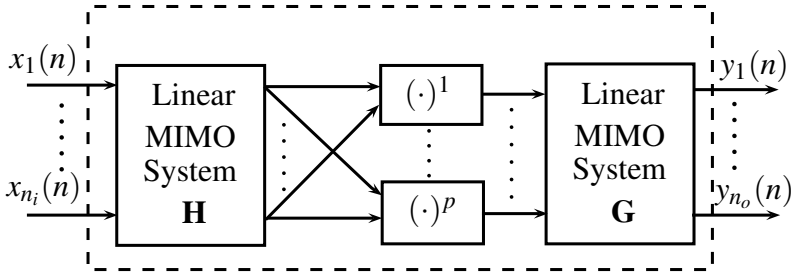


Fig. 6 A MIMO Wiener–Hammerstein system

2.3 Practical applications of MIMO Volterra systems

Nonlinear MIMO systems are found in a range of communication and control applications. These are shortly reviewed next.

2.3.1 Nonlinear communication systems

Communication systems equipped with multiple transmit and/or receive antennas are MIMO systems that help provide spatial diversity. Exploitation of spatial diversity results in higher capacity and performance improvements in interference reduction, fading mitigation and spectral efficiency. Most of existing MIMO schemes are limited to linear systems. However, in many cases, system nonlinearities are present and possible remedies based on linear MIMO approximations degrade performance significantly.

In a communication system, there are often limited resources (power, frequency, and time slots) which have to be efficiently shared by many users. Quite often in practice we encounter a situation whereby the number of users exceeds the number of available frequency or time slots. In infrastructure–based networks, a base station or an access point is responsible for allocating resources among the users, thereby reducing the access delays/transmission latency and improving quality–of–service (QoS). This is established through a variety of *multiple access* schemes. Two key multiple access technologies suitable for higher data rates are: orthogonal frequency–division multiple access (OFDMA) and code–division multiple access (CDMA).

OFDMA dynamically allocates resources both in frequency (by dividing the available bandwidth into a number of subbands, called subcarriers) and in time (via OFDM symbols). The transmission system assigns different users to groups of orthogonal subcarriers and thus allows them to be spaced very close together with no overhead as in frequency division multiple access. Furthermore it prevents interference between adjacent subcarriers. OFDMA has been implemented in several wireless communication standards (IEEE 802.11a/g/n wireless local area networks (WLANs), IEEE 802.16e/m worldwide interoperability for microwave access

(WiMAX), Hiperlan II), high–bit–rate digital subscriber lines (HDSL), asymmetric digital subscriber lines (ADSL), very high-speed digital subscriber lines (VHDSL), digital audio broadcasting (DAB), digital television and high-definition television (HDTV).

OFDMA is capable of mitigating intersymbol interference (ISI), (due to multipath propagation) using low–complexity/simple equalization structures. This is achieved by transforming the available bandwidth into multiple orthogonal narrowband subcarriers, where each subcarrier is sufficiently narrow to experience relatively flat fading. Nevertheless, OFDM is sensitive to synchronization issues and is characterized by high peak–to–average–power–ratio (PAPR), caused by the sum of several symbols with large power fluctuations. Such variations are problematic because practical communication systems are peak powered limited. In addition, OFDM transceivers are intrinsically sensitive to power amplifier (PA) nonlinear distortion [38], which dissipates the highest amount of power. One way to avoid nonlinear distortion is to operate the PA at the so–called "back–off" regime which results in low power efficiency. The trade–off between power efficiency and linearity motivated the development of signal processing tools that cope with MIMO–OFDM nonlinear distortion [40, 45, 38].

CDMA is based upon spread spectrum techniques. It plays an important role in third generation mobile systems (3G) and has found application in IEEE 802.11b/g (WLAN), Bluetooth, and cordless telephony. In CDMA multiple users share the same bandwidth at the same time through the use of (nearly) orthogonal spreading codes. The whole process effectively spreads the bandwidth over a wide frequency range (using pseudo–random code spreading or frequency hopping) several magnitudes higher than the original data rate. Two critical factors that limit the performance of CDMA systems are interchip and intersymbol interference (ICI/ISI), due to multipath propagation, mainly because they tend to destroy orthogonality between user codes and thus prevent interference elimination. Suppression of the detrimental effects of interference (ICI and ISI) get further complicated when nonlinear distortion is introduced due to power amplifiers. The combined effects of ICI, ISI and nonlinearities are comprehensively examined in [40, 67]. However, as recently illustrated in [22], the CDMA system model is sparse due to user inactivity/uncertainty, timing offsets and multipath propagation. CDMA system performance can be expected to improve further if nonlinearities along with sparse ICI/ISI are revisited.

2.3.2 MIMO nonlinear physiological systems

In several physiological applications it is mandatory to gain as much insight information is possible about the functioning of the system. It is well documented in the biomedical literature that nonlinear systems can significantly enhance the quality of modelling [57, 80]. Very often linear approximations discard significant information about the nonlinearities. For this reason, several physiological systems like sensory systems (cockroach tactile spine, auditory system, retina), reflex loops (in

the control of limb and eye position), organ systems (heart rate variability, renal auto–regulation) and tissue mechanics (lung tissue, skeletal muscle) have been approached via nonlinear system analysis using Volterra series [57, 80]. Many of the above physiological systems receive excitation from more than one input, and hence leads naturally to MIMO Volterra models.

2.3.3 Control applications

Quite often control applications exhibit multivariable interactions and nonlinear behaviour, which make the modelling task and design more challenging. Examples of such control systems include: multivariable polymerization reactor [32], fluid catalytic cracking units (FCCU) [83, 84], and rapid thermal chemical vapor decomposition systems (RTCVD) [72].

Multivariable polymerization reactor aims to control the reactor temperature at the unstable steady state by manipulating the cooling water and monomer flow rates. MIMO Volterra models have been employed to capture/track the nonlinear plant output [32]. The FCCU unit constitutes the workhorse of modern refinery and its purpose is to convert gas oil into a range of hydrocarbon products. The major challenges related to FCCU are its internal feedback loops (interactions) and its highly nonlinear behaviour [84]. RTCVD is a process used to deposit thin films on a semiconductor wafer via thermally activated chemical mechanisms. Process and equipment models for RTCVD consist mainly of balance equations for conservation of energy, momentum and mass, along with equations that describe the relevant chemical mechanisms. An important characteristic of RTCVD systems is their wide region of operation, which requires excitation of the system with as many modes as possible and hence a nonlinear MIMO system becomes relevant. A major challenge in all the above control applications is the large number of parameters required by the nonlinear MIMO models.

3 Algorithms for sparse multivariable filtering

Adaptive filters with a large number of coefficients are often encountered in multimedia signal processing, MIMO communications, biomedical applications, robotics, acoustic echo cancellation, and industrial control systems. Often, these applications are subject to nonlinear effects which can be captured using the models of Section 2. The steady–state and tracking performance of conventional adaptive algorithms can be improved by exploiting the sparsity of the unknown system. This is achieved via two different strategies [82]. The first is based on *proportionate adaptive filters*, which update each parameter of the filter independently of the others by adjusting the step size in proportion to the magnitude of the estimated filter parameter. In this manner, the adaptation gain is "proportionately" redistributed among all parameters, emphasizing the large coefficients in order to speed up convergence and increase the

overall convergence rate. The second strategy is motivated by the *compressed sensing* framework [16, 76, 36]. Compressed sensing approaches follow two main paths: (a) the ℓ_1 minimization (also referred to as basis pursuit) and (b) greedy algorithms (matching pursuit). Basis pursuit penalizes the cost function by the ℓ_1–norm of the unknown parameter vector (or a weighted ℓ_1–norm), as the ℓ_1–norm (unlike the ℓ_2–norm) favours sparse solutions. These methods combine conventional adaptive filtering algorithms such as LMS, RLS, etc with a sparsity promoting operation. Additional operations include the soft–thresholding (originally proposed for denoising by D. L. Donoho in [30]) and the metric projection onto the ℓ_1–ball [25, 33]. Greedy algorithms, on the other hand, iteratively compute the support set of the signal and construct an approximation of the parameters until convergence is reached. Proportionate adaptive filtering was developed by D. L. Duttweiler in 2000 [34]. Thereafter, a variety of improved versions has been proposed [64]. A connection between proportionate adaptive filtering and compressed sensing is discussed in [64].

3.1 Sparse multivariable Wiener filter

The block diagram of Fig. 7 shows a discrete–time MIMO filter with n_i inputs and n_o outputs [7, 47]. The output $\mathbf{y}(n)$, the impulse response matrix $\mathbf{H}$ and the input $\mathbf{x}(n)$ are related by:

$$\mathbf{y}(n) = \mathbf{H}\mathbf{x}(n) + \mathbf{v}(n) \tag{19}$$

where $\mathbf{x}(n)$ is defined in Eq. (11) and $\mathbf{v}(n) = [v_1(n), v_2(n), \cdots, v_{n_o}]^T$ is a Gaussian white noise vector. The following equation shows the rth output signal

$$y_r(n) = \sum_{\tau=1}^{n_i} \mathbf{h}_{r\tau}^T \mathbf{x}_\tau(n) + v_r(n) \tag{20}$$

$$= \mathbf{h}_{r:}^T \mathbf{x}(n) + v_r(n), \quad r = 1, \ldots, n_o. \tag{21}$$

and

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_{11}^T & \cdots & \mathbf{h}_{1n_i}^T \\ \vdots & \ddots & \vdots \\ \mathbf{h}_{n_o 1}^T & \cdots & \mathbf{h}_{n_o n_i}^T \end{bmatrix} = \begin{bmatrix} \mathbf{h}_{1:}^T \\ \vdots \\ \mathbf{h}_{n_o:}^T \end{bmatrix}. \tag{22}$$

An adaptive process is employed to cause the rth output to agree as closely as possible with the desired response signal $d_r(n)$. This is accomplished by comparing the outputs with the corresponding desired responses and by adjusting the parameters to minimize the resulting estimation error. More specifically, given an estimate $\hat{\mathbf{H}}(n)$ of $\mathbf{H}$ the estimation error is:

$$e_r(n) = y_r(n) - d_r(n) = y_r(n) - \widehat{\mathbf{h}}_{r:}^T(n)\mathbf{x}(n), \quad r = 1, \ldots, n_o \tag{23}$$

and in vector form:

1

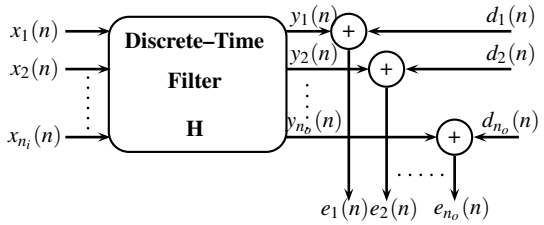


Fig. 7 MIMO filtering

$$\mathbf{e}(n) = \mathbf{y}(n) - \widehat{\mathbf{H}}(n)\mathbf{x}(n). \tag{24}$$

The performance of a filter is assessed by a functional of the estimation error. LS filters, minimize the total squared error:

$$J_{LS}(n) = \sum_{i=1}^{n} \mathbf{e}^{H^1}(i)\mathbf{e}(i) = \sum_{i=1}^{n} \|\mathbf{e}(i)\|_{\ell_2}^2 \tag{25}$$

$$= \sum_{r=1}^{n_o} J_{\mathbf{h}_{r:}}(n). \tag{26}$$

The optimum MIMO filter is given by the system of linear equations

$$\mathbf{H}_o(n)\mathbf{R}_{\mathbf{xx}}(n) = \mathbf{P}_{\mathbf{yx}}(n) \tag{27}$$

where $\mathbf{R}_{\mathbf{xx}}(n)$ is the input sample covariance matrix (which has block Toeplitz structure) with

$$\mathbf{R}_{x_i x_j}(n) = \sum_{t=1}^{n} \mathbf{x}_i(t)\mathbf{x}_j^H(t),$$

and

$$\mathbf{P}_{\mathbf{yx}}(n) = \sum_{i=1}^{n} \mathbf{y}(i)\mathbf{x}^H(i) = \left[\, \mathbf{p}_{yx_1}(n)\ \mathbf{p}_{yx_2}(n)\ \cdots\ \mathbf{p}_{yx_{n_i}}(n) \,\right]. \tag{28}$$

Under broad conditions the solutions of Eq. (27) tends to be the optimum mean squared filter (occasionally referred to as Wiener filter) that minimizes the mean squared error $\mathbb{E}\{\|\mathbf{e}(i)\|_{\ell_2}^2\}$ and satisfies the system of linear equations given by Eq. (27) ($\mathbf{P}_{\mathbf{yx}}(n) = \mathbb{E}\{\mathbf{y}(i)\mathbf{x}^H(i)\}$ and $\mathbf{R}_{\mathbf{xx}}(n) = \mathbb{E}\{\mathbf{x}(i)\mathbf{x}^H(i)\}$). Equation (27) can be decomposed in n_o independent MISO equations each corresponding to an output signal [9, 47], as follows:

$$\mathbf{h}_{r:,o}(n)\mathbf{R}_{\mathbf{xx}}(n) = \mathbf{p}_{y_r\mathbf{x}}(n), \quad r = 1, \ldots, n_o. \tag{29}$$

Consequently, minimizing $J_{LS}(n)$ or minimizing each $J_{\mathbf{h}_{r:}}(n)$ independently gives exactly the same results.

Two popular algorithms for adaptive filtering are the Least Mean Squares (LMS) algorithm and the Recursive Least Squares (RLS) algorithm. The LMS follows a stochastic gradient method and has a computationally simpler implementation. On the other hand, the more complex RLS has better convergence rate.

The LMS seeks to minimize the instantaneous error

$$J_{LMS}(n) = \mathbf{e}^H(n)\mathbf{e}(n). \tag{30}$$

The LMS estimate for the impulse response matrix $\mathbf{H}$ is based on the following update equation:

$$\mathbf{H}(n) = \mathbf{H}(n-1) + \mu \mathbf{e}(n)\mathbf{x}^H(n) \tag{31}$$

where the step size μ determines the convergence rate of the algorithm. To achieve convergence in the mean to the optimal Wiener solution, μ should be chosen so that:

$$0 < \mu < \frac{2}{M\sum_{\tau}^{n_i} \sigma_{x_\tau}^2}. \tag{32}$$

The RLS algorithm attempts to minimize the exponentially weighted cost function:

$$J_{RLS}(n) = \sum_{t=1}^{n} \lambda^{n-t} \mathbf{e}^H(t)\mathbf{e}(t) \tag{33}$$

where λ denotes the forgetting factor. The RLS estimates are updated as follows:

$$\mathbf{H}(n) = \mathbf{H}(n-1) + \mathbf{e}(n)\mathbf{k}^T(n) \tag{34}$$

where

$$\mathbf{k}(n) = \frac{\mathbf{R}_{\mathbf{xx}}^{-1}(n)\mathbf{x}^*(n)}{\lambda + \mathbf{x}^T(n)\mathbf{R}_{\mathbf{xx}}^{-1}(n)\mathbf{x}^*(n)}$$

is known as the *Kalman gain* [46, 70]. The matrix inversion lemma [46, 70], leads to:

$$\mathbf{R}_{\mathbf{xx}}^{-1}(n) = \lambda^{-1}\mathbf{R}_{\mathbf{xx}}^{-1}(n-1) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{R}_{\mathbf{xx}}^{-1}(n-1). \tag{35}$$

3.2 $\mathscr{L}_1$ constrained adaptive filters

These algorithms are based on the minimization of cost functions penalized by the ℓ_1–norm (or a weighted ℓ_1–norm or an approximate ℓ_0–norm) and are inspired by the fact that the ℓ_1–norm promotes sparse solutions and is the best convex relaxation to the ℓ_0 quasi–norm.

Table 1 ZA–LMS Algorithm

Algorithm description
H(0)=**0**
For n:=1,2,... **do**
1: $\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{H}(n-1)\mathbf{x}(n)$
2: $\mathbf{H}(n) = \mathbf{H}(n-1) + \mu\mathbf{e}(n)\mathbf{x}^H(n) - \gamma \text{sgn}(\mathbf{H}(n-1))$
End For

3.2.1 LMS–type filters

The sparse cost function combines the instantaneous error with a sparseness inducing penalty term

$$J_{ZA-LMS}(n) = \frac{1}{2}\left(\mathbf{e}^H(n)\mathbf{e}(n)\right) + \tau \text{pen}(\mathbf{H}(n)) \tag{36}$$

τ is a positive scalar regularization parameter which provides a trade–off between penalization and signal reconstruction error. The most well–known sparsity inducing penalty term is the ℓ_1–norm ($\text{pen}(\mathbf{H}(n)) = \|\text{vec}[\mathbf{H}(n)]\|_{\ell_1}$). Although a large portion of the literature focuses on the ℓ_1–norm there are other functions which promote sparsity [48, 42]. In fact, any penalization term, with $\text{pen}(\mathbf{H}(n))$ being symmetric, monotonically non–decreasing, and with decreasing derivative will serve the same purpose [36].

Sparse LMS–type variants obey the following updating scheme

$$\left\{\begin{array}{c} new \\ parameter \\ estimate \end{array}\right\} = \left\{\begin{array}{c} old \\ parameter \\ estimate \end{array}\right\} + \left\{ stepsize \right\} \left\{\begin{array}{c} new \\ information \end{array}\right\} + \left\{\begin{array}{c} zero \\ attraction \\ term \end{array}\right\}$$

where the new information term is the error vector between the outputs of the filter and the desired signal vector. The *Zero–Attraction* (ZA) term is a norm related regularization function which exerts an attraction to zero on small parameters. Convergence of the recursion may be slow because the two parts are hard to balance. This issue is addressed in some detail later in the subsection.

The first of this type of algorithms (originally developed in [18, 19] for SISO systems) minimizes Eq. (36). The filter parameter matrix is updated by

$$\begin{aligned} \mathbf{H}(n) &= \mathbf{H}(n-1) - \mu\nabla J_{ZA-LMS}(n) \\ &= \mathbf{H}(n-1) + \mu\mathbf{e}(n)\mathbf{x}^H(n) - \gamma\nabla^s \text{pen}(\mathbf{H}(n-1)) \end{aligned} \tag{37}$$

where $\nabla^s \text{pen}(\mathbf{H}(n-1))$ is the sub–gradient of the convex function $\text{pen}(\mathbf{H}(n-1))$, $\gamma = \mu\tau$ is the regularization parameter. In the adaptive filtering context γ is also referred to as *regularization step size*. Usually the regularization step size is fine tuned offline (via exhaustive simulations) or in an ad–hoc manner. A systematic approach to choosing γ is developed in [19].

Under the standard compressive sensing setting, the penalty is given by the ℓ_1–norm and the resulting algorithm is shown in Table 1. Note that $\mathrm{sgn}(\cdot)$ is a *component–wise sign function* defined as

$$\mathrm{sgn}\,(H_{ij}) = \begin{cases} H_{ij}/|H_{ij}| & \text{if } H_{ij} \neq 0, \\ 0 & \text{if } H_{ij} = 0. \end{cases} \tag{38}$$

It is well known that the LMS, in a stationary environment, achieves unbiased convergence in the mean to the Wiener solution (using the independence assumption) [46]. However, unlike the conventional LMS, ZA–LMS leads to a biased behaviour [48], that is

$$\mathbb{E}\,[\mathbf{H}(n)] = \mathbf{H}_o - \frac{\gamma}{\mu}\mathbb{E}\,[\mathbf{H}(n)]\,\mathbf{R}_{\mathbf{xx}}^{-1}(n), \quad \text{as } n \to \infty \tag{39}$$

Recall that a key difference between the ℓ_0 norm and the ℓ_1 norm penalty, is that the ℓ_1 norm depends on the magnitudes of the non–zero components, whereas the ℓ_0–norm penalty does not. As a result, the larger a component is, the heavier it is penalized by the ℓ_1 penalty. To overcome this often unfair penalization two different penalty terms are introduced in the conventional LMS cost function. Both form better approximations to the ℓ_0 norm. The first is based on an approximation of the step function [79]

$$\mathrm{pen}\,(\mathbf{H}(n)) = \sum_i \left(1 - \exp^{-a'|\mathrm{vec}_i[\mathbf{H}(n)]|}\right) \tag{40}$$

where $a > 0$ is a parameter that must be chosen. The authors in [49] reduce the computational complexity of the resulting zero attraction term by considering the first order Taylor series expansion of exponential functions. The resulting filter update iteration (named ℓ_0–LMS) becomes

$$\mathbf{H}(n) = \mathbf{H}(n-1) + \mu\mathbf{e}(n)\mathbf{x}^H(n) - \gamma a\,(1 - a|\mathbf{H}(n-1)|)_+\,\mathrm{sgn}\,(\mathbf{H}(n-1)) \tag{41}$$

where $(x)_+ = \max\{x, 0\}$. Motivated by the re–weighted ℓ_1 cost function in [17], the authors in [18] follow this approach in order to reinforce the ZA–LMS by re–weighting the sparse penalty term. The proposed penalty term is given by

$$\mathrm{pen}\,(\mathbf{H}(n)) = \sum_i \log\left(1 + \varepsilon'^{-1}|\mathrm{vec}_i\,[\mathbf{H}(n)]\,|\right). \tag{42}$$

According to the stochastic gradient approach, the resulting filter update iteration is

$$\mathbf{H}(n) = \mathbf{H}(n-1) + \mu\mathbf{e}(n)\mathbf{x}^H(n) - \gamma\frac{\mathrm{sgn}\,(\mathbf{H}(n-1))}{1 + \varepsilon|\mathbf{H}(n-1)|} \tag{43}$$

and the algorithm is named RZA–LMS. Small coordinates of the estimated matrix are more heavily weighted (by $1/(1 + \varepsilon|\mathbf{H}(n-1)|)$) towards zero, and small weights encourage larger coordinates. As a result, the bias of the mean value of the converged matrix for RZA–LMS is reduced.

So far we have examined how to solve the penalized LMS cost function of Eq. (36) by embedding additional terms to the update formula. A different viewpoint arises by considering *proximity splitting* methods [21]. The proximity operator of a (possibly non–differentiable) convex function $\Omega(\mathbf{H}(n))$ is defined as

$$\operatorname{prox}_{\tau,\Omega}(\mathbf{H}(n)) := \underset{\mathbf{H}(n)}{\operatorname{argmin}} \quad \frac{1}{2\tau}\|\mathbf{Y}(n)-\mathbf{H}(n)\|_{\ell_2}^2 + \Omega(\mathbf{H}(n)).$$

Proximity operators are the main ingredient of proximal methods [21] which arise in many well–known algorithms (*e.g.*, iterative thresholding, projected Landweber, projected gradient, alternating projections, alternating-direction method of multipliers, alternating split Bregman). In these algorithms, proximal methods can be understood as generalizations of quasi–Newton methods to non–differentiable convex problems. An important example is the Iterative Thresholding procedure [24] which solves problems of the form:

$$\min_{\mathbf{H}} \quad J(\mathbf{H}) + \Omega(\mathbf{H}), \tag{44}$$

$J(\mathbf{H})$ is differentiable with Lipschitz gradient. By iterating the fixed point equation

$$\mathbf{H}(n) := \underbrace{\operatorname{prox}_{\mu,\Omega}}_{\text{backward step}} \underbrace{[\mathbf{H}(n-1) - \mu\nabla J(\mathbf{H}(n-1))]}_{\text{forward step}} \tag{45}$$

for values of the step–size parameter μ in a suitable bounded interval. This scheme is known as a forward–backward splitting algorithm. In some cases, the proximity operator $\operatorname{prox}_{\mu,\Omega}$ can be evaluated in closed form.

If we consider the minimization of the cost function J_{ZA-LMS} (defined in Eq. (36)) with $\operatorname{pen}(\mathbf{H}(n)) = \|\operatorname{vec}[\mathbf{H}(n)]\|_{\ell_1}$ we obtain

$$\min_{\mathbf{H}} \quad \frac{1}{2}|\mathbf{e}(n)|^2 + \tau\|\operatorname{vec}[\mathbf{H}(n)]\|_{\ell_1}.$$

We observe that the above problem is a special case of Eq. (44) with

$$\begin{cases} J: & \mathbf{H} \mapsto \frac{1}{2}|\mathbf{e}(n)|, \\ \Omega: & \mathbf{H} \mapsto \tau\|\operatorname{vec}[\mathbf{H}(n)]\|_{\ell_1}. \end{cases}$$

Then it follows from [21, 61] that the proximity operator $\operatorname{prox}_{\mu,\Omega}$ leads to a nonlinear component–wise shrinkage operation known as soft–thresholding [30]. The component-wise *soft–thresholding* operation is defined by

$$\mathbb{S}_\tau[H_{ij}] = \begin{cases} H_{ij} - \tau & \text{if } H_{ij} \geq \tau, \\ 0 & \text{if } |H_{ij}| \leq 0, \\ H_{ij} + \tau & \text{if } H_{ij} \leq \tau \end{cases} \tag{46}$$

or in compact notation $\mathbb{S}_\tau[H_{ij}] = \text{sgn}(H_{ij})(|H_{ij}| - \tau)_+$ [30]. This operation shrinks coefficients above the threshold in magnitude by an amount equal to τ. An instantaneous proximity operation leads to the soft–thresholded LMS filter

$$\mathbf{H}(n) = \mathbb{S}_\tau\left[\mathbf{H}(n-1) + \mu\mathbf{e}(n)\mathbf{x}^H(n)\right]. \tag{47}$$

Detailed analysis of the dynamics of Eq. (47) in its batch format, has shown that the algorithm converges initially relatively fast, then it overshoots the ℓ_1 penalty, and it takes very long to re–correct back. To avoid such a behaviour in the adaptive case, we force the successive iterates to remain within a particular ℓ_1 ball B_R [25]. To achieve this the thresholding operation is replaced by a projection $\mathbb{P}_{B_R}$, where, for any closed convex set $\mathscr{C}$ and any $\mathbf{H}$, the projection $\mathbb{P}_{\mathscr{C}}(\mathbf{H})$ is defined as the unique point in $\mathscr{C}$ for which the ℓ_2 distance to $\mathbf{H}$ is minimal. We thus obtain the projected LMS on ℓ_1 ball

$$\mathbf{H}(n) = \mathbb{P}_{B_R}\left[\mathbf{H}(n-1) + \mu\mathbf{e}(n)\mathbf{x}^H(n)\right]. \tag{48}$$

The projection operator $\mathbb{P}_{B_R}[H_{ij}(n)]$ is obtained by a suitable thresholding of $H_{ij}(n)$, given

$$\mathbb{P}_{B_R}[H_{ij}] = \begin{cases} \mathbb{P}_{B_R}[H_{ij}] = \mathbb{S}_\mu[H_{ij}] & \text{if } \|\,\text{vec}[\mathbf{H}(n)]\,\|_{\ell_1} > R, \text{ and choose } \mu \\ & \text{such that } \|\mathbb{S}_\mu[\text{vec}[\mathbf{H}(n)]]\,\|_{\ell_1} = R \\ \mathbb{P}_{B_R}[H_{ij}] = \mathbb{S}_0[H_{ij}] & \text{if } \|vec[\mathbf{H}(n)]\,\|_{\ell_1} \leq R. \end{cases} \tag{49}$$

Using proximal splitting methods other types of adaptive filters, such as NLMS/APA and Adaptive Projection algorithms, can be modified to promote sparsity [54, 61].

3.2.2 RLS–type filters

Sparse RLS–type filters modify the RLS cost function (33) by the addition of a sparsifying term:

$$J_{ZA-RLS}(n) = \frac{1}{2}J_{RLS}(n) + \tau\text{pen}(\mathbf{H}(n)). \tag{50}$$

The regularization parameter τ controls sparsity and weighted squared error. The sparse RLS filter can be seen as an adaptive version of Gauss–Newton or Newton–Raphson search with sparse updates [55]. Alternatively, the RLS algorithm is a special case of a Kalman filter [46, 70]. The main recursion takes the following form:

$$\left\{\begin{array}{c} new \\ parameter \\ estimate \end{array}\right\} = \left\{\begin{array}{c} old \\ parameter \\ estimate \end{array}\right\} + \left\{\begin{array}{c} Kalman \\ gain \end{array}\right\} \left\{\begin{array}{c} innovation \\ vector \end{array}\right\} + \left\{\begin{array}{c} zero \\ attraction \\ term \end{array}\right\}$$

The correction term is proportional to the innovation error vector between the predicted observations and the actual observations. The coefficients of this correction are provided by the Kalman gain. For the regularized Recursive Least Square prob-

lem of Eq. (50) the solution of the Wiener equation takes the form [35]:

$$\mathbf{H}(n) = \mathbf{P}_{\mathbf{yx}}(n)\mathbf{C}(n) - \gamma(1-\lambda)\nabla^s \text{pen}\left(\mathbf{H}(n-1)\right)\mathbf{C}(n) \tag{51}$$

where $\mathbf{C}(n) = \mathbf{R}_{\mathbf{xx}}^{-1}(n)$, $\lambda \in (0,1)$ is the forgetting factor and $\nabla^s \text{pen}\left(\mathbf{H}(n-1)\right)$ is a subgradient since $J_{ZA-RLS}(n)$ is non–differentiable at any point where $H_{ij}(n) = 0$ [10, p. 227]. The exponentially weighted autocorrelation and cross–correlation matrices are recursively updated as:

$$\mathbf{R}_{\mathbf{xx}}(n) = \sum_{t=1}^{n} \lambda^{n-t}\mathbf{x}(t)\mathbf{x}^H(t) = \lambda\mathbf{R}_{\mathbf{xx}}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n) \tag{52}$$

$$\mathbf{P}_{\mathbf{yx}}(n) = \sum_{t=1}^{n} \lambda^{n-t}\mathbf{y}(t)\mathbf{x}^H(t) = \lambda\mathbf{P}_{\mathbf{yx}}(n-1) + \mathbf{y}(n)\mathbf{x}^H(n). \tag{53}$$

The regularized RLS filter relies on the following recursion [35]:

$$\mathbf{H}(n) = \mathbf{H}(n-1) + \mathbf{e}(n)\mathbf{k}^T(n) - \gamma(1-\lambda)\nabla^s \text{pen}\left(\mathbf{H}(n-1)\right)\mathbf{C}(n) \tag{54}$$

where the regularization parameter γ is usually fine tuned offline or using the selection rule proposed in [35] (for white inputs). In this case the corresponding subgradient is $\nabla^s \|H_{ij}(n-1)\|_{\ell_1} = \text{sgn}(H_{ij}(n-1))$. Instead we may utilize the penalty functions, suggested in the LMS context, given by Eqs. (40) and (42).

RLS algorithms are developed based on the batch LASSO estimator in [4, 3]. This method modifies the LASSO cost function to include a forgetting factor:

$$\underset{\mathbf{H}(n)}{\text{argmin}} \quad \frac{1}{\sigma^2}\sum_{i=1}^{n} \lambda^{n-i}\|\mathbf{y}(i) - \mathbf{H}(i)\mathbf{x}(i)\|_{\ell_2}^2 + \gamma\text{pen}\left(\mathbf{H}(n)\right). \tag{55}$$

The first order subgradient based optimality conditions for the exponentially weighted LASSO cost imply:

$$\begin{cases} \nabla_{ij}J_{RLS}(n) + \tau\,\text{sgn}\left(H_{ij}(n)\right), & \text{if } H_{ij}(n) \neq 0 \\ |\nabla_{ij}J_{RLS}(n)| \leq \tau & \text{if } H_{ij}(n) = 0. \end{cases}$$

These conditions and the value of $\nabla_{ij}J_{RLS}(n)$ are used to define a pseudo–gradient for each component of $\mathbf{H}$ [2]. The pseudo–gradient of $J_{R-LASSO}(n)$ is the element of the sub–differential of $J_{R-LASSO}(n)$ at $\mathbf{H}(n)$ with minimum norm and is given by:

$$\nabla_{ij}J_{R-LASSO}(n) = \begin{cases} \nabla_{ij}J_{RLS}(n) + \tau\,\text{sgn}\left(H_{ij}(n)\right), & \text{if } H_{ij}(n) \neq 0 \\ \nabla_{ij}J_{RLS}(n) + \tau & \text{if } H_{ij}(n) = 0, \nabla_{ij}J_{RLS}(n) < -\tau \\ \nabla_{ij}J_{RLS}(n) - \tau & \text{if } H_{ij}(n) = 0, \nabla_{ij}J_{RLS}(n) > \tau \\ 0 & \text{if } H_{ij}(n) = 0, -\tau \leq \nabla_{ij}J_{RLS}(n) \leq \tau. \end{cases}$$

In the first case the function is differentiable, so the pseudo–gradient is simply the gradient with respect to ij (the only element of the sub–gradient). In the remain-

Table 2 R-LASSO Algorithm

Algorithm description
$\mathbf{R_{xx}}(0)=\mathbf{0}$,$\mathbf{P_{yx}}(0)=\mathbf{0}$,$\mathbf{H}(0)=\mathbf{0}$
For n:=1,2,... **do**
1: $\mathbf{R_{xx}}(n)=\lambda\mathbf{R_{yx}}(n-1)+\mathbf{x}(n)\mathbf{x}^H(n)$
2: $\mathbf{P_{yx}}(n)=\lambda\mathbf{P_{yx}}(n-1)+\mathbf{y}(n)\mathbf{x}^H(n)$
3: $\nabla J_{R-LASSO}(n)=\begin{cases}\mathbf{H}(n-1)\mathbf{R_{xx}}(n)-\mathbf{P_{yx}}(n)+\tau\mathrm{sgn}(\mathbf{H}(n-1)) & \text{if } H_{ij}\neq 0,\\ \mathbb{S}_\tau\left[\mathbf{H}(n-1)\mathbf{R_{xx}}(n)-\mathbf{P_{yx}}(n)\right] & \text{if } H_{ij}=0.\end{cases}$
4: $\mathbf{H}(n)=\mathbf{H}(n-1)+\mu_n\nabla J_{R-LASSO}(n)$
End For

ing three cases we obtain the minimum–norm solution by the soft–thresholding operation to $\nabla_{ij}J_{RLS}(n)$. The global solution to the smooth part of the LASSO cost function is the Wiener equation, where the autocorrelation and the cross–correlation matrices are recursively updated from Eqs. (52) and (53).

Using the subgradient, an instantaneous subgradient descent strategy is employed for online updating as follows

$$\mathbf{H}(n)=\mathbf{H}(n-1)+\mu\nabla J_{R-LASSO}(n). \tag{56}$$

The Recursive LASSO (R–LASSO) filter outlined here is summarized in Table 2. As with the batch LASSO estimator, the R–LASSO does not necessarily converge to the true parameter $\mathbf{H}$ since it fails to recover the correct support and at the same time estimate the non–zero entries of $\mathbf{H}$ consistently [4].

In order to improve the performance of the R–LASSO filter, one could use a different penalty term which is signal dependent and weights differently the entries in the ℓ_1 norm, that is $\mathrm{pen}\left(\mathbf{H}(n)\right)=\sum_i w_\tau(|\,\mathrm{vec}_i\left[\widehat{\mathbf{H}}^{RLS}(n)\right]\,|)\|\,\mathrm{vec}_i\left[\mathbf{H}(n)\right]\|_{\ell_1}$. By generalizing the *Smoothly Clipped Absolute Deviation* (SCAD) regularizer introduced for the batch weighted LASSO estimator to its adaptive case, the following weight function is obtained

$$w_\tau(|\,\mathrm{vec}_i\left[\mathbf{H}(n)\right]\,|)=\frac{\left[\alpha\tau-|\,\mathrm{vec}_i\left[\mathbf{H}(n)\right]\,|\right]_+}{\tau(\alpha-1)}u(|\,\mathrm{vec}_i\left[\mathbf{H}(n)\right]\,|-\tau)+u(\tau-|\,\mathrm{vec}_i\left[\mathbf{H}(n)\right]\,|)$$

$u(\cdot)$ stands for the step function and α is usually set to 3.7. The reweighted LASSO estimator (RW–LASSO) places higher weight to small entries, and lower weight to entries with large amplitudes. In fact, the estimates of size less than τ are penalized as in R–LASSO, while estimates between τ and $\alpha\tau$ are penalized in a linearly decreasing manner. Estimates larger than $\alpha\tau$ are not penalized at all. The implementation of RW–LASSO is established using an instantaneous pseudo–gradient descent strategy, similar to R–LASSO. The downside of this estimator is its high complexity because it requires running in parallel an RLS algorithm to supply the needed weights.

Table 3 spaRLS Algorithm

Algorithm description
$\mathbf{R_{xx}}(0)=\mathbf{0}$,$\mathbf{P_{yx}}(0)=\mathbf{0}$,$\mathbf{H}(0)=\mathbf{0}$
For n:=1,2,... **do**
1: $\mathbf{R_{xx}}(n)=\lambda\mathbf{R_{xx}}(n-1)+\frac{a^2}{\sigma_x^2}\mathbf{x}(n)\mathbf{x}^H(n)$
2: $\mathbf{P_{yx}}(n)=\lambda\mathbf{P_{yx}}(n-1)+\frac{a^2}{\sigma^2}\mathbf{y}(n)\mathbf{x}^H(n)$
3: **Repeat**
4: $\widehat{\mathbf{G}}^{(\lambda)}(n)=\widehat{\mathbf{H}}^{(\lambda)}(n)\left(\mathbf{I}-\mathbf{R_{xx}}(n)\right)+\mathbf{P_{yx}}(n)$
5: $\widehat{\mathbf{H}}^{(\lambda)}(n)=\mathbb{S}_{\gamma a^2}\left[\widehat{\mathbf{G}}^{(\lambda)}(n)\right]$
6: **Until** $\lambda=k$
End For

A different viewpoint to sparse RLS algorithms is provided in [5] (and its MIMO extension in [53]). This approach makes use of the Expectation Maximization (EM) method [27] to derive an adaptive filter that solve a penalized Maximum Likelihood problem. The penalized Recursive Least Squares problem may be posed as a penalized Maximum Likelihood problem [41]. This penalized ML problem can be efficiently solved by an EM algorithm following the noise decomposition idea (proposed in [41]) in order to divide the optimization problem into a denoising and a filtering problem. Consider the following decomposition for $\mathbf{V}(n)$

$$\mathbf{V}(n)=\alpha\mathbf{V}_1(n)\mathbf{X}(n)+\mathbf{V}_2(n). \tag{57}$$

The noise matrices are ensembles of Gaussian–distributed random matrices

$$\begin{aligned}\mathbf{V}_1(n) &= (\mathbf{0},\mathbf{I}_{n_i}\otimes\mathbf{I}_{n_o})\\ \mathbf{V}_2(n) &= \left(\mathbf{0},\left(\sigma^2\Lambda^{-1}-\alpha^2\mathbf{X}(n)\mathbf{X}^H(n)\right)^T\otimes\mathbf{I}_{n_o}\right)\end{aligned}$$

where $\Lambda:=\text{diag}\left[\lambda^{n-1}\cdots\lambda^0\right]$ and α is a constant which must fulfil $\alpha\leq\sigma^2/\lambda_{max}[\mathbf{X}(n)\mathbf{X}^H(n)]$ with $\lambda_{max}[\cdot]$ being the maximum eigenvalue. Since $\lambda_{max}[\mathbf{X}(n)\mathbf{X}^H(n)]\approx n_i$ for large n and for independent input, $\alpha^2=\sigma^2/5n_i$ satisfies this condition with high probability. Therefore the model is rewritten as follows:

$$\begin{cases}\mathbf{Y}(n)=\mathbf{G}(n)\mathbf{X}(n)+\mathbf{V}_2(n)\\ \mathbf{G}(n)=\mathbf{H}(n)+\alpha\mathbf{V}_1\end{cases}. \tag{58}$$

The EM algorithm is used to solve the following penalized ML problem

$$\mathbf{H}(n)=\underset{\mathbf{H}(n)}{\text{argmax}}\quad\log P(\mathbf{Y}(n),\mathbf{V}(n),|\mathbf{H}(n))-\gamma\text{pen}\left(\mathbf{H}(n)\right)) \tag{59}$$

which is easier to solve, by employing $\mathbf{V}(n)$ as the auxiliary variable. The λth iteration of the EM algorithm is defined as [5]:

$$\begin{cases} \text{E–Step} & Q(\mathbf{H},\mathbf{H}(n)) = -\frac{1}{2\alpha^2}\|\mathbf{G}^{(\lambda)}(n)-\mathbf{H}\|_{\ell_2}^2 - \gamma\|\operatorname{vec}[\mathbf{H}]\|_{\ell_1} \\ \text{M–Step} & \mathbf{H}^{(\lambda+1)}(n) = \underset{\mathbf{H}(n)}{\operatorname{argmax}} \quad Q(\mathbf{H},\mathbf{H}(n)) = \mathbb{S}_{\gamma\alpha^2}\left(\mathbf{G}^{(\lambda)}(n)\right) \end{cases} \tag{60}$$

where

$$\mathbf{G}^{(\lambda)}(n) = \mathbf{H}^{(\lambda)}(n)\left(\mathbf{I} - \frac{\alpha^2}{\sigma^2}\mathbf{X}(n)\Lambda\mathbf{X}^H(n)\right) + \frac{\alpha^2}{\sigma^2}\mathbf{Y}(n)\Lambda\mathbf{X}^H(n)$$

The above algorithm is an iterated shrinkage method. The soft thresholding function tends to decrease the support of $\mathbf{H}(n)$, since it shrinks the support to those elements whose absolute value is greater than $\gamma\alpha^2$. The algorithm described above can be further simplified by considering only the corresponding positions of the non–zero entries within the thresholding step [5]. The autocorrelation and cross–corellation matrices, which appear in the E–step of the algorithm, can be obtained recursively and the resulting algorithm (known as spaRLS) is summarized in Table 3.

Another algorithm related to the EM approach is presented in [52]. Unlike the noise decomposition idea which is followed in [5], their approach uses normal priors on the unknown parameter matrix. In the EM approach the individual parameters are treated as missing variables, and the E–step computes the conditional expectation of the missing variables given past observations. Subsequently, the M–Step maximizes this expectation minus a sparsity inducing penalty (like the ℓ_1 norm). To apply the EM approach the complete and incomplete data must be specified. The matrix $\mathbf{H}(n)$ at time n is taken to represent the complete data vector, whereas $\mathbf{Y}(n-1)$ accounts for the incomplete data [39, pp. 31–33]. The resulting EM approach is summarized by the following equation:

$$\mathbf{G}(n) = \arg\max_{\mathbf{G}} \left\{ \mathbb{E}_{p(\mathbf{H}(n)|\mathbf{Y}(n-1);\mathbf{G}(n-1))}\left[\log p(\mathbf{H}(n);\mathbf{G})\right] - \gamma\|\operatorname{vec}[\mathbf{G}]\|_{\ell_1} \right\}. \tag{61}$$

The EM algorithm aims to maximize the log–likelihood of the complete data, $\log p(\mathbf{H}(n);\mathbf{G})$. However, because $\mathbf{H}(n)$ is an unknown parameter, it maximizes instead its expectation given the incomplete data $\mathbf{Y}(n-1)$ and a current estimate of the parameters $\mathbf{G}(n-1)$. The E–step, computes the conditional expectation of the log–likelihood, given observations $\mathbf{Y}(n-1)$ and parameter estimate $\mathbf{G}(n-1)$ from the previous iteration

$$\begin{aligned} \textbf{E-step :} \quad Q\big(\mathbf{G},\mathbf{G}(n-1)\big) &= \mathbb{E}_{p(\mathbf{H}(n)|\mathbf{Y}(n-1);\mathbf{G}(n-1))}\left[\log p(\mathbf{H}(n);\mathbf{G})\right] \\ &= \text{constant} + \mathbf{G}^H\mathbf{S}^{-1}(n)\mathbb{E}[\mathbf{H}(n)|\mathbf{Y}(n-1);\mathbf{G}(n-1)] - \frac{1}{2}\mathbf{G}^H\mathbf{S}^{-1}(n)\mathbf{G} \end{aligned} \tag{62}$$

where $\mathbf{S}(n)$ is a diagonal covariance matrix, and the constant incorporates all terms that do not involve $\mathbf{G}$ and hence do not affect maximization. The M–step, described below, calculates the maximum of the penalized Q–function

$$\begin{aligned} \textbf{M-step :} \quad \mathbf{G}(n) &= \arg\max_{\mathbf{G}} \left\{ Q\big(\mathbf{G},\mathbf{G}(n-1)\big) - \gamma\|\operatorname{vec}[\mathbf{G}]\|_{\ell_1} \right\} \\ &= \mathbb{S}_{\gamma\mathbf{S}_{ii}(n)}\left(\mathbb{E}\left[\mathbf{H}(n)|\mathbf{Y}(n-1);\mathbf{G}(n-1)\right]\right) \end{aligned} \tag{63}$$

Table 4 EM–RLS Algorithm

Algorithm description
$\mathbf{H}(0)=\mathbf{0}$, $\mathbf{C}_0=\delta^{-1}I$ with δ =const.
For n:=1,2,... **do**
1: $\mathbf{k}(n)=\dfrac{\mathbf{C}(n-1)\mathbf{x}^*(n)}{\lambda+\mathbf{x}^T(n)\mathbf{C}(n-1)\mathbf{x}^H(n)}$
2: $\mathbf{G}(n)=\mathbf{H}(n-1)+(\mathbf{y}(n)-\mathbf{H}(n-1)\mathbf{x}(n))\mathbf{k}^T(n)$
3: $\mathbf{C}(n)=\lambda^{-1}\mathbf{C}(n-1)-\lambda^{-1}\mathbf{k}(n)\mathbf{x}^T(n)\mathbf{C}(n-1)$
4: $\mathbf{H}(n)=\mathbb{S}_{\gamma\lambda^{-1}\mathbf{C}(n-1)}[\mathbf{G}(n)]$
End For

which in turn leads to the *soft thresholding* function. In order to carry out the conditional expectation of Eq. (62) (essentially the E–step), one needs to assume a prior on $\mathbf{H}(n)$ given the past observations $\mathbf{Y}(n-1)$ and $\mathbf{G}(n-1)$. Consider the Gaussian prior of the form

$$\text{Prior}=p(\mathbf{H}(n)|\mathbf{Y}(n-1);\mathbf{G}(n-1))\simeq\mathscr{N}(\mathbf{G}(n-1),\mathbf{S}(n)).$$

It is well known that this conditional expectation may be obtained recursively using the Kalman filter, if a Gaussian prior is assumed on $\mathbf{H}(n)$ given the past observation. The Kalman filter then determines the posterior probability density function for $\mathbf{H}(n)$ recursively over time. In a Bayesian context if $\mathbf{H}(n)$ is assumed to be Gaussian, the RLS filter can be regarded as a Kalman filter [55]. Therefore, the main recursion takes the form [55, 70]

$$\begin{aligned}\mathbf{H}(n)&=\mathbf{H}(n-1)+\mathbf{e}(n)\mathbf{k}^T(n)\\ \mathbf{C}(n)&=\lambda^{-1}\mathbf{C}(n-1)-\lambda^{-1}\mathbf{k}(n)\mathbf{x}^T(n)\mathbf{C}(n-1)\end{aligned}$$

where $\mathbf{k}(n)$ is the Kalman gain and $\mathbf{e}(n)$ denotes the prediction error given by $\mathbf{e}(n)=\mathbf{y}(n)-\mathbf{H}(n-1)\mathbf{x}(n)$. Hence $\mathbf{H}(n)$ depends linearly on $\mathbf{G}$. The Riccati equation that updates $\mathbf{C}(n)=\mathbf{R_{xx}}^{-1}(n)$ indicates that $\mathbf{C}(n)$ does not depend on $\mathbf{G}$. Moreover, $\mathbb{E}[\mathbf{e}(n)\mathbf{Y}(n-1)]=0$ because the prediction error $\mathbf{e}(n)$ is uncorrelated to measurements. The i^{th} diagonal component of the prior covariance $\mathbf{S}_i(n)$ can be computed as follows

$$\mathbf{S}_i(n)=\lambda^{-1}\mathbf{C}_i(n-1).$$

The method outlined above is named EM–RLS filter and is summarized in Table 4.

3.3 Greedy adaptive filters

Greedy algorithms provide an alternative approach to ℓ_1 penalization methods. For the recovery of a sparse parameter matrix in the presence of noise, greedy algorithms iteratively improve the current estimate by modifying one or more elements

until a halting condition is met. The basic principle behind greedy algorithms is to iteratively find the support set of the sparse matrix and reconstruct it using the restricted support Least Squares (LS) estimate. The computational complexity depends on the number of iterations required to find the correct support set. One of the earliest algorithms proposed for sparse signal recovery is the Orthogonal Matching Pursuit (OMP) [26, 65, 75]. At each iteration, OMP finds the entry of the proxy matrix $\mathbf{P}(n) = (\mathbf{Y}(n) - \mathbf{H}\mathbf{X}(n))\mathbf{X}^H(n)$ with the largest magnitude, and adds it to the support set. Then, it solves the following least squares problem:

$$\hat{\mathbf{H}} = \arg\min_{\mathbf{H}} \|\mathbf{Y}(n) - \mathbf{H}\mathbf{X}(n)\|_{\ell_2}^2$$

and updates the residual. By repeating these steps a total of s times, the support of $\mathbf{H}$ is recovered.

Several improvements have been proposed for greedy reconstruction. The Stagewise OMP (StOMP), proposed in [31], selects all proxy components whose values are above a certain threshold. Due to the multiple selection step, StOMP achieves better runtime than OMP. On the other hand, parameter tuning in StOMP might be difficult and there are rigorous asymptotic results available. A more sophisticated algorithm was developed by Needell and Vershynin, and is known as Regularized OMP (ROMP) [63]. ROMP chooses the s largest components of the proxy, and applies a regularization step to ensure that not too many incorrect components are selected. The recovery bounds obtained in [63] are optimal up to a logarithmic factor. Tighter recovery bounds which avoid the presence of the logarithmic factor are obtained by Needell and Tropp via the Compressed Sampling Matching Pursuit algorithm (CoSaMP) [62]. CoSaMP provides tighter recovery bounds than ROMP that are optimal up to a constant factor. An algorithm similar to the CoSaMP, was presented by Dai and Milenkovic and is known as Subspace Pursuit (SP) [23].

As with most greedy algorithms, CoSaMP takes advantage of the measurement matrix $\mathbf{X}(n)$ which is assumed to be approximately orthonormal ($\mathbf{X}(n)\mathbf{X}^H(n)$ is close to the identity matrix). Hence, the largest components of the signal proxy $\mathbf{P}(n) = \mathbf{H}\mathbf{X}(n)\mathbf{X}^H(n)$ most likely correspond to the non–zero rows of $\mathbf{H}$. Next, the algorithm adds the largest components of the signal proxy to the running support set and performs least squares to get an estimate for the signal. Finally, it prunes the least square estimation and updates the error residual. The main ingredients of the CoSaMP algorithm are outlined below:

Identification of the largest $2s$ components of the proxy signal
Support Merger: forms the union of the set of newly identified components with the set of indices corresponding to the s largest components of the least squares estimate obtained in the previous iteration
Estimation via least squares on the merged set of components
Pruning: restricts the LS estimate to its s largest components
Sample update: updates the error residual.

The above steps are repeated until a halting criterion is met. The main difference between CoSaMP and SP is in the identification step where the SP algorithm chooses the s largest components.

It was established in [58] that greedy algorithms can be converted into an adaptive mode, while maintaining their superior performance gains. We demonstrate below that this conversion is applicable in the multichannel set up. We focus our analysis on CoSaMP/SP due to their superior performance, but similar ideas are applicable to other greedy algorithms as well. Multichannel greedy algorithms can be approached via two strategies. The first approach assumes that the subsystems share the same sparsity pattern. Hence the greedy algorithm simultaneously recovers the support set (also known as joint sparsity or group sparsity) [12, 75] by choosing an element which reaches the maximum value of the multichannel energy. Under the second strategy adopted here, the subsystems exhibit different sparsity patterns [56]. Next greedy versions of the main adaptive multichannel algorithms are presented based on the CoSaMP/SP platform.

3.3.1 Greedy LMS filter

The multichannel adaptive greedy LMS algorithm modifies the proxy identification, estimation and error residual update. The error residual is evaluated by

$$\mathbf{v}(n) = \mathbf{y}(n) - \mathbf{H}(n)\mathbf{x}(n). \tag{64}$$

The above formula involves the current sample only, in contrast to the CoSaMP/SP scheme which requires all previous samples. A new proxy signal that is more suitable for the adaptive mode, is defined as:

$$\mathbf{P}(n) = \sum_{i=1}^{n-1} \lambda^{n-1-i} \mathbf{v}(i)\mathbf{x}^H(i)$$

and is updated by

$$\mathbf{P}(n) = \lambda \mathbf{P}(n-1) + \mathbf{v}(n-1)\mathbf{x}^H(n)$$

This way the algorithm is capable of capturing variations on the support of $\mathbf{H}$. The estimate $\mathbf{H}(n)$ is updated by the LMS recursion [46, 70]. At each iteration the current regressor $\mathbf{x}(n)$ and the previous estimate $\mathbf{H}(n-1)$ are restricted to the instantaneous support originated from the support merging step. However, because the row support corresponding to each output is different, some extra care is required. Recall that any MIMO filter with n_o outputs is simplified to n_o MISO adaptive filters (all of which have different row support). Let Λ denote the estimated set of indices and $\Lambda^{(r)}$ $(r = 1, 2, \ldots, n_o)$ the set of indices associated with the rth row of $\mathbf{H}(n)$. The update equation for the rth output is given by

$$\mathbf{h}_{r:|\Lambda^{(r)}}(n) = \mathbf{h}_{r:|\Lambda^{(r)}}(n-1) + \mu e_r(n)\mathbf{x}^H_{|\Lambda^{(r)}}(n), \quad \forall r = 1, \ldots, n_o \tag{65}$$

where $\mathbf{x}_{|\Lambda^{(r)}}(n)$ denotes the sub–vector corresponding to the index set $\Lambda^{(r)}$. If all rows of $\mathbf{H}$ share the same row support then the update step can be performed jointly

Table 5 SpAdOMP Algorithm

Algorithm description	
$\mathbf{H}(0)=\mathbf{0},\mathbf{W}(0)=\mathbf{0},\mathbf{P}(0)=\mathbf{0}$	{Initiliazation}
$\mathbf{v}(0)=\mathbf{y}(0)$	{Initial residual}
$0<\lambda\leq 1$	{Forgetting factor}
$0<\mu<2\lambda_{\max}^{-1}$	{Step size}
For $n:=1,2,\ldots$ **do**	
1: $\mathbf{P}(n)=\lambda\mathbf{P}(n-1)+\mathbf{v}(n-1)\mathbf{x}^H(n-1)$	{Form signal proxy}
2: $\Omega=\text{supp}(\mathbf{P}_{2s}(n))$	{Identify large components}
3: $\Lambda=\Omega\cup\text{supp}(\mathbf{H}(n-1))$	{Merge supports}
4: $e_r(n)=y_r(n)-\mathbf{w}_{r:\lvert\Lambda^{(r)}}(n-1)\mathbf{x}_{\lvert\Lambda^{(r)}}(n)$	{Prediction error}
5: $\mathbf{w}_{r:\lvert\Lambda^{(r)}}(n)=\mathbf{w}_{r:\lvert\Lambda^{(r)}}(n-1)+\mu e_r(n)\mathbf{x}^H_{\lvert\Lambda^{(r)}}(n)$	{LMS iteration}
6: $\Lambda_s=\max(\lvert\mathbf{H}_{\lvert\Lambda}(n)\rvert,s)$	{Obtain the pruned support}
7: $\mathbf{H}_{\lvert\Lambda_s}(n)=\mathbf{W}_{\lvert\Lambda_s}(n),\quad \mathbf{H}_{\lvert\Lambda_s^c}(n)=0$	{Prune the LMS estimates}
8: $\mathbf{v}(n)=\mathbf{y}(n)-\mathbf{H}(n)\mathbf{x}(n)$	{Update error residual}
end For	

for all outputs and the selection of the largest proxy signal components is simplified [75].

The multichannel Sparse Adaptive Orthogonal Matching Pursuit (SpAdOMP) algorithm, is presented in Table 5. The operator $\max(|a|,s)$ returns s indices of the largest elements of a and Λ^c represents the complement of Λ. An important point to note about step 5 of Table 5 is that the choice of a proper step–size μ that ensures convergence is difficult. The Normalized LMS (NLMS) addresses this issue by scaling with the input power

$$\mathbf{h}_{r:\lvert\Lambda^{(r)}}(n)=\mathbf{h}_{r:\lvert\Lambda^{(r)}}(n-1)+\frac{\mu}{\varepsilon+\|\mathbf{x}_{\lvert\Lambda^{(r)}}(n)\|^2}e_r(n)\mathbf{x}^H_{\lvert\Lambda^{(r)}}(n),\quad \forall r=1,\ldots,n_o$$

where $0<\mu<2$ and ε is a small positive constant (inserted to avoid division by small numbers). NLMS may be viewed as an LMS with time–varying step–size. This partially explains the superior tracking performance as compared to LMS in non–stationary environments.

3.3.2 Greedy RLS filter

In this subsection we develop greedy adaptive schemes whose estimation part is based on rank one updates for the autocorrelation and cross–correlation matrices. A straightforward forward attempt towards this direction, would be to re–use the framework adapted by the SpAdOMP algorithm [58] (Table 5) and replace the estimation step with the RLS algorithm. However in doing so, we will have to up-

date the entries of the inverse covariance matrix as well as the Kalman gain entries which are required to perform an RLS update for the currently estimated support set. A more efficient technique avoids the last action of the CoSaMP/SP framework (Sample Update) and is described next.

Consider the normal equations

$$\mathbf{H}\mathbf{X}(n)\mathbf{X}^H(n) = \mathbf{Y}(n)\mathbf{X}^H(n). \tag{66}$$

An iterative method known as Landweber–Fridman or Van Cittert iteration [29, 78] is incorporated in order to express Eq. (66) into an equivalent fixed point equation of the form

$$\mathbf{H} = \mathbf{H} + (\mathbf{Y}(n) - \mathbf{H}\mathbf{X}(n))\,\mathbf{X}^H(n).$$

The Landweber iteration starts from an initial guess $\mathbf{H}^0$ and solves $\mathbf{y}(n) = \mathbf{H}\mathbf{x}(n)$ iteratively by

$$\mathbf{H}^{(t)} = \mathbf{H}^{(t-1)} + \left(\mathbf{Y}(n) - \mathbf{H}^{(t-1)}\mathbf{X}(n)\right)\mathbf{X}^H(n) \quad t = 1,2,\ldots$$

The above iteration requires the norm of $\mathbf{X}(n)$ to be less than or equal to one, otherwise it diverges or converges too slowly. To avoid divergence and accelerate the speed of convergence a step size term μ is introduced

$$\mathbf{H}^{(t)} = \mathbf{H}^{(t-1)} + \mu\left(\mathbf{Y}(n) - \mathbf{H}^{(t-1)}\mathbf{X}(n)\right)\mathbf{X}^H(n) \quad t = 1,2,\ldots \tag{67}$$

where $\mu \in (0, 2/\|\mathbf{X}(n)\mathbf{X}^H(n)\|)$. The above iterations is similar to Steepest Descent except that the step size term is fixed. To derive an adaptive Landweber filter we rewrite Eq. (67) as

$$\mathbf{H}^{(t)} = \mathbf{H}^{(t-1)}\left(\mathbf{I} - \mu\mathbf{X}(n)\mathbf{X}^H(n)\right) + \mu\mathbf{Y}(n)\mathbf{X}^H(n) \quad t = 1,2,\ldots \tag{68}$$

The above iteration requires the autocorrelation matrix $\mathbf{R}_{\mathbf{xx}}(n) = \mathbf{X}(n)\mathbf{X}^H(n)$ and the cross–correlation matrix $\mathbf{P}_{\mathbf{yx}}(n) = \mathbf{Y}(n)\mathbf{X}(n)^H$. In practice, the data arrive sequentially and might vary with time. For this reason we approximate $\mathbf{R}_{\mathbf{xx}}(n)$ and $\mathbf{P}_{\mathbf{yx}}(n)$ via exponentially weighted sample averages [46, 70]. Therefore the Landweber iteration takes the form

$$\mathbf{H}(n) = \mathbf{H}(n-1)\left(\mathbf{I} - \mu\mathbf{R}_{\mathbf{xx}}(n)\right) + \mu\mathbf{P}_{\mathbf{yx}}(n) \quad n = 1,2,\ldots \tag{69}$$

The resulting expression is identical to the one derived in [5] (Step 4 in Table 3) via the EM formulation and the decomposition of the noise vector.

Finally let us take a second look at the proxy signal and the sample update, which are described in section 3.3. The authors in [58] proposed an adaptive mechanism to estimate the signal proxy and the sample update. Examination of

$$\mathbf{P}(n) = (\mathbf{Y}(n) - \mathbf{H}\mathbf{X}(n))\,\mathbf{X}^H(n) \tag{70}$$

Table 6 SpAdOMP (RLS) Algorithm

Algorithm description	
$\mathbf{H}(0)=\mathbf{0},\mathbf{W}(0)=\mathbf{0},\mathbf{P}(0)=\mathbf{0},\mathbf{R}_{\mathbf{xx}}(0)=\mathbf{0},\mathbf{P}_{\mathbf{yx}}(0)=\mathbf{0}$	{Initiliazation}
For $n:=1,2,\ldots$ **do**	
1: $\mathbf{R}_{\mathbf{xx}}(n)=\lambda\mathbf{R}_{\mathbf{xx}}(n-1)+\mathbf{x}(n)\mathbf{x}^H(n)$	{Update autocorrelation}
2: $\mathbf{P}_{\mathbf{yx}}(n)=\lambda\mathbf{P}_{\mathbf{yx}}(n-1)+\mathbf{y}(n)\mathbf{x}^H(n)$	{Update cross–correlation}
3: $\mathbf{P}(n)=\mathbf{P}_{\mathbf{yx}}(n)-\mathbf{H}(n)\mathbf{R}_{\mathbf{xx}}(n)$	{Form signal proxy}
4: $\Omega=\mathrm{supp}(\mathbf{P}_{2s}(n))$	{Identify large components}
5: $\Lambda=\Omega\cup\mathrm{supp}(\mathbf{H}(n-1))$	{Merge supports}
6: $\mathbf{W}(n)=\mathbf{W}(n-1)\left(\mathbf{I}-\mu\mathbf{R}_{\mathbf{xx}}(n)\right)+\mu\mathbf{P}_{\mathbf{yx}}(n)$	{Recursive Landweber iteration}
7: $\Lambda_s=\max(\lvert\mathbf{W}_{\lvert\Lambda}(n)\rvert,s)$	{Obtain the pruned support}
8: $\mathbf{H}_{\lvert\Lambda_s}(n)=\mathbf{W}_{\lvert\Lambda_s}(n),\quad \mathbf{H}_{\lvert\Lambda_s^c}(n)=0$	{Prune the Landweber estimates}
end For	

shows that the sample update constitutes an ingredient of the signal proxy. Additionally, the above equation can be re–expressed as follows

$$\mathbf{P}(n)=\mathbf{Y}(n)\mathbf{X}^H(n)-\mathbf{H}\mathbf{X}(n)\mathbf{X}^H(n)\simeq\mathbf{P}_{\mathbf{yx}}(n)-\mathbf{H}\mathbf{R}_{\mathbf{xx}}(n) \tag{71}$$

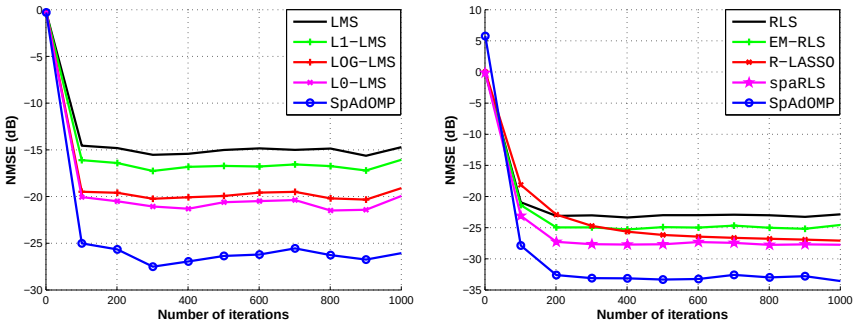
and hence there is no need for the sample update, since all the required information is obtained from the correlation and cross–correlation matrices. The algorithm is summarized in Table 6. The key difference between spaRLS and this version of SpAdOMP algorithm is that the latter has two mechanisms for support estimation (the proxy signal followed by pruning which is a special form of hard thresholding) and hence can achieve better support estimation.

3.4 Computer Simulations of Sparse Adaptive MIMO Filters

In this subsection we demonstrate and compare the performance of the algorithms outlined in this section. Computer simulations are conducted under different scenarios in order to evaluate performance over a wide range of conditions. The Normalized Mean Square Error (NMSE, in dB scale)

$$\mathrm{NMSE}_{ij}:=\mathrm{MC}^{-1}\sum_{t=1}^{\mathrm{MC}}\frac{\sum_{n=1}^{N}|\hat{H}_{ij}^{(t)}(n)-H_{ij}(n)|^2}{\sum_{n=1}^{N}|H_{ij}(n)|^2}$$

is used as performance measure, where $\hat{H}_{ij}^{(t)}(n)$ denotes the estimate of the ij subsystem for the tth Monte Carlo (MC) run. The overall NMSE is obtained by averaging over all subsystems



(a) LMS–type filters (b) RLS–type filters

Fig. 8 Learning curves of adaptive MIMO filters

$$\text{NMSE} := \frac{1}{n_o \times n_i \times M} \sum_{i=1}^{n_o} \sum_{j=1}^{n_i \times M} \text{NMSE}_{ij}. \tag{72}$$

All NMSE results were obtained for 50 different system realizations (every non–zero parameter at each realization is assigned to random locations and their values are generated randomly from a complex normal distribution). The experiments are conducted in a moderate noise environment with Signal to Noise Ratio (SNR $:= 10\log \|\mathbf{H}\|_{\ell_2}^2 / \|\mathbf{v}\|_{\ell_2}^2$) of 15 dB.

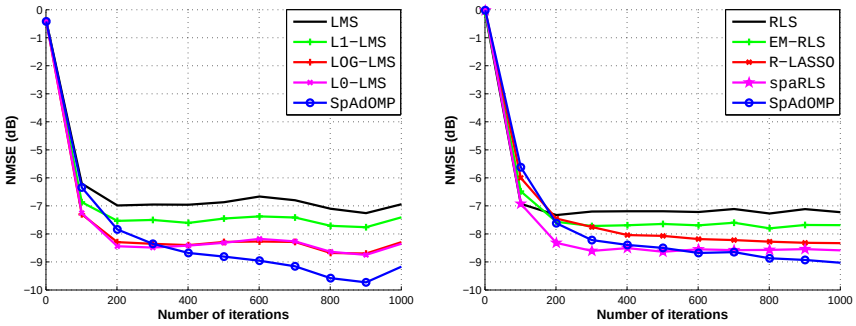
To compare the performance of different adaptive filters we use their corresponding *learning curves* which are plots of the NMSE versus the number of iterations. Learning curves help us visualize the convergence and tracking behaviour of adaptive filters. Note that although the LMS and RLS type filters are examined under the same scenarios, we have chosen to plot them separately due to different convergence speeds and computational complexity requirements.

Adaptive identification of linear MIMO systems

First we consider a linear $(3,3)$–MIMO system with a memory length $M = 5$ and 5 non–zero elements. The system is excited by a complex Gaussian input signal with zero mean and variance $1/5$. For a fair comparison between all competing LMS–type filters the step size is common and equal to

$$\mu_n = \frac{1}{\|\mathbf{x}(n)\|_{\ell_2}^2}.$$

The regularization step size γ for the ZA–LMS (or ℓ_1–LMS) and the RZA–LMS (or *log*–LMS) are adjusted adaptively following the systematic approach introduced in [19]. For the ℓ_0–LMS filter the regularization parameters required offline fine tuning



(a) LMS–type filters (b) RLS–type filters

Fig. 9 Learning curves of adaptive nonlinear MIMO filters

and the best performance is obtained when $\alpha = 5$ and $\gamma = 0.01$. The SpAdOMP filter required a–priori knowledge of the sparsity level in order to perform the adaptive greedy selection procedure. Fig. 8 (a) shows that SpAdOMP obtains the faster convergence and better steady state accuracy, followed by the ℓ_0–LMS and log–LMS whose performance is nearly identical.

The RLS–type filters share a common forgetting factor $\lambda = 0.98$. The R–LASSO, spaRLS and SpAdOMP follow an instantaneous steepest descent pattern (that involves the autocorrelation and cross–correlation matrices) and employ a step size to accelerate convergence. The step size is set to

$$\mu_n = \frac{0.3}{\|\mathbf{x}(n)\|^2_{\ell_2}} \tag{73}$$

for all schemes. The EM–RLS, R–LASSO and SpaRLS required offline processing to find the optimum regularization parameter for each filter ($\gamma_{EM-RLS} = 6 \times 10^{-4}$, $\tau_{R-LASSO} = 0.3$ and $\alpha^2\gamma_{spaRLS} = 0.03$). The adaptive greedy filter (SpAdOMP) is fine tuned using a–priori knowledge of the sparsity level ($s = 5$). Fig. 8 (b) presents the learning curves of RLS–type of filters. We observe that the adaptive greedy filter gives the best performance. It is followed by spaRLS, R–LASSO and EM–RLS. The convergence rate of spaRLS, R–LASSO and EM–RLS can be significantly improved if a more sparsity aware regularization function is employed (like those discussed in section 3.2.1).

Adaptive identification of nolinear MIMO systems

Next, we evaluate the filtering performance of sparse nonlinearly mixed MIMO systems. The MIMO system consists of 3 inputs, 3 outputs, has memory length $M = 2$ and poses a quadratic nonlinearity where all different product combinations of the inputs are allowable. The combination of sparsity with nonlinearity significantly in-

creases the parameter space of the unknown system matrix and may give rise to *degeneracy* in the parameters. Note that degeneracy causes all important parameters to be close to zero and as a result some outputs may also be zero. To avoid this situation we consider 9 non–zero parameters, 6 of which belong to the linear part of the system (spread among different inputs) and 3 correspond to the nonlinear part. The input sequence is drawn from a complex Gaussian distribution of zero mean and variance 1/9.

Initially we compare the learning curves of LMS–type filters. The step size is common to all filters and given by Eq. (73). Unlike the linear case, it was experimentally found that the systematic approach (developed in [19]) for choosing the best regularization parameter for ZA–LMS and RZA–LMS (or ℓ_1–LMS and log–LMS as we will refer to respectively) does not perform that well in the case of nonlinearly mixed MIMO. Therefore, ℓ_1–LMS, log–LMS and ℓ_0–LMS are required to optimize their parameters via exhaustive simulations and the corresponding values are summarized in the following table.

ℓ_1**–LMS** (γ)	**log–LMS** (γ,ε)	ℓ_0**–LMS** (γ,α)	**EM–RLS** (γ)	**R–LASSO** (γ)	**spaRLS** ($\alpha^2\gamma$)
5×10^{-3}	1×10^{-2},10	1×10^{-3},5	2×10^{-3}	9×10^{-2}	2×10^{-3}

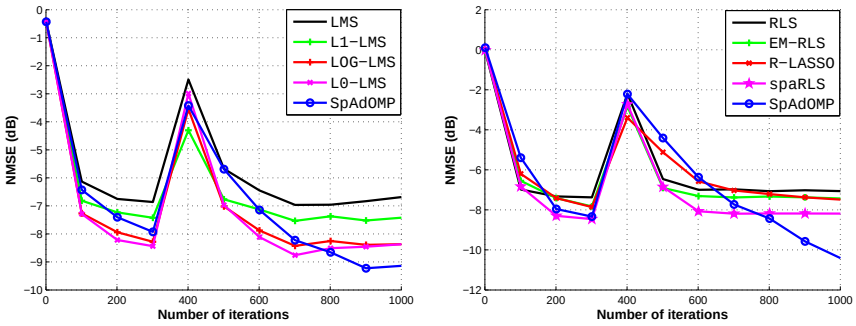
The conclusions drawn from inspection of Fig. 9 (a), are almost identical to those in the linear case. However, this time the convergence speed of the greedy filter is slightly worse than the one obtained by ℓ_0–LMS and log–LMS filter.

Next, we study the performance of RLS–type of filters in nonlinearly mixed MIMO systems. As in the linear case, some filters require offline processing to fine tune their regularization parameters and the optimum values are summarized in the above table. Fig. 9 (b) shows that almost all RLS–type of filters achieve relatively similar steady–state accuracy, and spaRLS has the fastest convergence speed.

One common conclusion for LMS and RLS type of filters, operating in nonlinear MIMO systems, is that the extraordinary good performance of adaptive greedy filters is slightly degraded. This is because greedy filters require strongly incoherent dictionaries (this has been studied for the linear case in [58]). The problem of designing input sequences with incoherent dictionary for nonlinear MIMO systems requires further work.

Tracking performance of sparse adaptive filters

The time–varying nonlinear MIMO system is initialized using the same parameters as used to generate Fig. 9. At the 400th iteration the system experiences a sudden change, where all active parameters of the nonlinear part randomly change locations. We note from Fig. 10 that spaRLS, ℓ_0–LMS and log–LMS have the fastest support tracking behaviour and that adaptive greedy filters achieve better steady state accuracy.



(a) LMS–type filters (b) RLS–type filters

Fig. 10 Comparison of tracking performances on nonlinear MIMO systems

4 Blind and semi–blind identification of sparse MIMO systems excited by finite–alphabet inputs

This section is concerned with the sparse MIMO parameter estimation problem encountered in blind system identification whereby the unknown sparse MIMO system is estimated using output information as well as some a priori knowledge of the system. This problem arises in digital communications, seismic data, image deblurring and speech coding.

The sparse blind MIMO identification problem has been approached by two different methodologies, namely: 1) dictionary learning [36, Ch. 12] and 2) maximum penalized likelihood estimator (via the Expectation Maximization algorithm) [60]. The first approach solves an optimization problem by iteratively applying two convex steps: the parameter update step on a fixed measurement matrix and the measurement matrix update step on a fixed parameter. The second approach, employs Expectation–Maximization for finding maximum penalized likelihood estimates. Both algorithms do not converge to global minima, whereas for the case of dictionary learning even a local minimum can not be guaranteed.

In this section we discuss joint state estimation and sparse parameter estimation techniques under the finite–alphabet property. Two different techniques are described. The first algorithm maximizes the likelihood of the received sequence over all possible input sequences and system parameters. It does so by converting the joint maximization into a two stage maximization problem. For a given parameter value at the end of the ℓ–th iteration, the most likely state (equivalently input sequence) is estimated by carrying out the inner maximization. This maximization can be performed by the Viterbi algorithm since the polynomial MIMO system is represented by a Hidden Markov Model (HMM). Once the inner maximization is completed and the most likely state sequence is determined, the outer maximization takes over. Given the state sequence at step ℓ, maximization of the penalized likelihood with respect to system parameters is effected by sparsity aware schemes.

The two main stages (state estimation, parameter estimation) iterate until a stopping criterion is satisfied.

The second blind estimation method considered in this section is based on Expectation Maximization (EM). Instead of working with likelihood, EM employs the augmented likelihood formed by the so called complete data which consist of the state sequence and the output sequence. It turns out that maximization of the augmented likelihood is easier to perform. Then the EM procedure alternates between the E–step during which the log–likelihood function of the complete data is estimated, and the M–step which maximizes the augmented likelihood to generate an updated parameter matrix. Parameter sparsity is naturally embedded in the M–step by the insertion of a penalty term (typically the ℓ_1 norm of the parameters).

4.1 An Alternating Maximum Likelihood procedure to state estimation and sparse system estimation

Let us consider the basic set up defined in Section 2. The input–output relationship is given by

$$\mathbf{y}(n) = f(x_1(n), x_1(n-1), \ldots, x_1(n-M), \cdots, x_{n_i}(n), x_{n_i}(n-1), \ldots, x_{n_i}(n-M)) + \mathbf{v}(n). \tag{74}$$

The noise vector $\mathbf{v}(n)$ is a multivariate Gaussian i.i.d. with mean and covariance matrix $\mathscr{N}(0, \mathbf{Q})$. Following the analysis of Section 2.1 let

$$\bar{\mathbf{x}}(n) = [x_1(n), x_1(n-1), \ldots, x_1(n-M), \cdots, x_{n_i}(n), x_{n_i}(n-1), \ldots, x_{n_i}(n-M)]^T.$$

Hence the nonlinear input vector is given by

$$\mathbf{x}(n) = [\bar{\mathbf{x}}(n), \bar{\mathbf{x}}_2(n), \cdots, \bar{\mathbf{x}}_p(n)]^T.$$

We shall refer to $\mathbf{x}(n)$ as the augmented state or simply the state. Eq. (74) is compactly written as

$$\mathbf{y}(n) = \mathbf{H}\mathbf{x}(n) + \mathbf{v}(n).$$

In a blind (or semi–blind) environment, information on the input sequence that generated a given output is not available. Suppose $\mathbf{Y}(n) = [\mathbf{y}(1), \mathbf{y}(2), \cdots, \mathbf{y}(n)]$ denotes the known $n_i \times n$ observation sequence. The task of joint state estimation and system parameter estimation is based solely on a small number of measurements n. The probability density function (PDF) of the observation matrix $\mathbf{Y}(n)$ conditioned on $(\mathbf{X}(n), \mathbf{H})$ is given by

$$p(\mathbf{Y}(n)|\mathbf{H}, \mathbf{X}(n)) = \frac{1}{(2\pi\sigma^2)^{n_o \times n}} \exp\left(-\frac{1}{2\sigma^2} \sum_{t=1}^{n} \|\mathbf{y}(t) - \mathbf{H}\mathbf{x}(t)\|_{\ell_2}^2 \right) \tag{75}$$

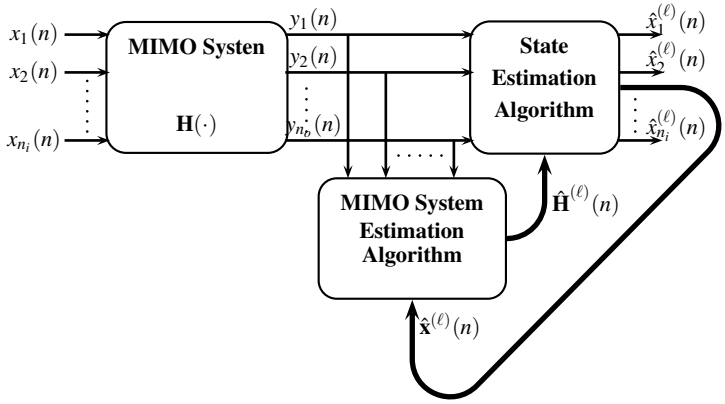


Fig. 11 Alternating MIMO detector–estimator

The joint Maximum Likelihood (ML) estimator of $\mathbf{X}(n)$ and $\mathbf{H}$ is obtained by jointly maximizing $p(\mathbf{Y}(n)|\mathbf{H},\mathbf{X}(n))$ over $\mathbf{X}(n)$ and $\mathbf{H}$, as follows:

$$(\hat{\mathbf{X}}(n),\hat{\mathbf{H}}) = \arg\max_{\mathbf{X}(n),\mathbf{H}} \log p(\mathbf{Y}(n)|\mathbf{H},\mathbf{X}(n)).$$

The above optimization problem is intractable. We thus convert it into a two stage maximization problem that is iteratively performed over $\mathbf{X}(n)$ and $\mathbf{H}$ (see Fig. 11) as

$$(\hat{\mathbf{X}}(n),\hat{\mathbf{H}}) = \arg\max_{\mathbf{H}}\max_{\mathbf{X}(n)} \log p(\mathbf{Y}(n)|\mathbf{H},\mathbf{X}(n)). \tag{76}$$

The iterative procedure alternates information between a state estimation scheme and a system parameter estimation scheme. In several applications, including communications, the input signals take values in a finite–alphabet. Then the state vector evolves as a Markov chain and the input–output relationship becomes a Hidden Markov Process (HMP) [37]. Therefore the inner maximization at step (ℓ) can be accomplished by dynamic programming and the Viterbi Algorithm (VA). Given $\mathbf{X}^{(\ell)}$ the iterative process updates the system parameters. The outer level maximization is equivalent to a quadratic minimization problem. Hence, the optimum solution leads to a set of normal equations:

$$\mathbf{H}^{(\ell)}\mathbf{X}^{(\ell)}(n)\,\mathbf{X}^{(\ell)\,H}(n) = \mathbf{Y}(n)\,\mathbf{X}^{(\ell)\,H}(n).$$

The algorithm is repeated until a fixed point is reached or until a stopping criterion is met. Local convergence of the algorithm can be established [37]. The above procedure is known in the literature under several different names: Baum–Viterbi [37], ML Alternating Least Squares [1, 68] and bootstrap equalization [73].

Remark: Although the above procedure can operate in a pure blind fashion, it converges very slowly and suffers from an inherent permutation and scaling ambiguity problem [74]. This ambiguity is resolved if very few training input samples

Table 7 Baum–Viterbi Algorithm

Algorithm description		
$\ell = 0:$	$\mathbf{H}^{(0)} = \arg\max_{\mathbf{H}} \log p(\mathbf{Y}(T)\|\mathbf{H},\mathbf{X}(T)) - \tau\|\text{vec}[\mathbf{H}]\|_{\ell_1}$	{Initiliazation}
Repeat		
	$\ell = \ell + 1$	
1:	$\mathbf{X}^{(\ell)}(n) = \arg\max_{\mathbf{X}(n)\in S} \log p(\mathbf{Y}(n)\|\mathbf{H}^{(\ell-1)})$	{Viterbi Algorithm}
2:	$\mathbf{H}^{(\ell)} = \arg\max_{\mathbf{H}} \log p(\mathbf{Y}(n)\|\mathbf{X}^{(\ell)}(n)) - \tau\|\text{vec}[\mathbf{H}]\|_{\ell_1}$	{System Parameter Re–estimation}
Until $(\mathbf{X}^{(\ell)}(n),\mathbf{H}^{(\ell)}(n)) \approx (\mathbf{X}^{(\ell-1)}(n),\mathbf{H}^{(\ell-1)}(n))$		

are used to provide an initial parameter matrix $\mathbf{H}^{(0)}$ estimate. The initial estimate does not need to be accurate enough since it is improved through successive iterations. The minimum number of training data, namely, $T = n_i M$, is equal to the rank of the MIMO system. The training symbol matrix $\mathbf{X}^{(0)}$ can be designed to yield the optimal estimation performance.

In the MIMO models discussed in Section 2 the parameter space increases exponentially and often the number of parameters exceeds the number of available measurements and the resulting system becomes underdetermined. Additionally, the unknown system may exhibit slow time–variations, so that during the time period of n data the entries of $\mathbf{H}$ may be considered constant. Therefore, even if $\mathbf{X}(n)$ is known, estimating $\hat{\mathbf{H}}$ remains an underdetermined problem. The key observation here is to consider the parameters of $\mathbf{H}$ that actually contribute to the output (see Section 2.1.1). This motivates the addition of a regularization term into the cost function for joint state estimation and sparse parameter estimation. Following the Compressed Sensing paradigm, the ℓ_1 penalty term is added and the cost function takes the form:

$$(\hat{\mathbf{X}}(n),\hat{\mathbf{H}}) = \arg\left\{\max_{\mathbf{H}}\left[\max_{\mathbf{X}(n)\in S} \log p(\mathbf{Y}(n)|\mathbf{H},\mathbf{X}(n)) - \tau\|\text{vec}[\mathbf{H}]\|_{\ell_1}\right]\right\}. \tag{77}$$

The maximization of the likelihood with respect to $\mathbf{H}$ resembles the basis pursuit or LASSO criterion and hence any compressed sensing algorithm can be used to perform this maximization [76, 36].

A two stage maximization algorithm of this type is summarized in Table 7. The first step involves an approximation to $\hat{\mathbf{X}}(n)$ which is obtained using a Maximum A Posteriori (MAP) criterion:

$$\arg\max_{\mathbf{X}(n)} \quad p(\mathbf{X}(n)|\mathbf{Y}(n);\mathbf{H}^{(\ell-1)}). \tag{78}$$

The above is solved using the *Viterbi* algorithm. The Viterbi algorithm searches among all possible paths through the state trellis in order to efficiently find the most probable path. A pseudo–code is provided in Table 8.

Table 8 Viterbi Algorithm

Algorithm description	
$\delta_1(i) = \log p(\mathbf{y}(1)\|\mathbf{x}^{(i)}(1);\mathbf{H}^{(\ell-1)}(1)), \quad i = 1,\ldots,A^{n_i M}$	{Initiliazation}
For $t := 2,\ldots,n$ **do**	
For $j := 1,2,\ldots,A^{n_i M}$ **do**	
1: $\delta_t(j) = \log p(\mathbf{y}(t)\|\mathbf{x}^{(j)}(t);\mathbf{H}^{(\ell-1)}(t)) + \max_i [\delta_{t-1}(i)]$	{Recursion}
2: $\psi_t(j) = \arg\max_i [\delta_{t-1}(i)]$	
End	
End	
3: $i_n = \arg\max_i \delta_n(i), \quad \hat{\mathbf{x}}(n) = \mathbf{x}^{(i_n)}(n)$	{Termination}
4: $i_t = \psi_{t+1}(i_{t+1}), \quad \hat{\mathbf{x}}(t) = \mathbf{x}^{(i_t)}, \quad t = n-1,\ldots,1$	{Backtracking}

Maximization of the penalized likelihood over $\mathbf{H}$ is equivalent to maximizing the auxiliary function [37, 50]:

$$\sum_{t=1}^{n}\sum_{i}^{A^{n_i M}} \delta(\hat{\mathbf{x}}^{(\ell)}(n) - \mathbf{x}_i) \log p\left(\mathbf{y}(t)|\mathbf{H}\right) - \tau\|\mathrm{vec}[\mathbf{H}]\|_{\ell_1} \tag{79}$$

where $\delta(\cdot)$ is the delta function that is equal to one when $\mathbf{x}^{(\ell)}(n) = \mathbf{x}_i$ and zero otherwise. Since the noise is Gaussian, expression (79) is equivalent to penalized least squares estimation. The linearity in the parameters leads to the following closed form expression

$$\mathbf{H}^{(\ell)} = \mathbb{S}_\tau \left[\left(\mathbf{X}^{(\ell)}(n)\, \mathbf{X}^{(\ell)\,H}(n) \right)^{-1} \mathbf{Y}(n)\, \mathbf{X}^{(\ell)\,H}(n) \right]. \tag{80}$$

ML Detection via sphere decoding. The state estimator based on the Viterbi algorithm requires searching over $A^{M\times n_i}$ possible trellis state. This is affordable when A and $M \times n_i$ are small but it is not realistic when M is large. An alternative decoder structure, employs a sphere decoder [11, 20]. The underlying principle of sphere decoding is to search the closest lattice point (or vector) to the output signal within a sphere of radius r centered at the output signal. Sphere decoding techniques increase the radius when there exists no vector within a sphere, and decrease the radius when there exist multiple vectors within the sphere. The main idea is to limit the search among the possible states to those located within a sphere having radius r. We write with some abuse of notation the following:

$$\hat{\mathbf{X}}(n) = \arg\min_{\mathbf{X}(n)} \quad \|\mathbf{Y}(n) - \mathbf{H}\mathbf{X}(n)\|_{\ell_2}^2 \le r^2 \tag{81}$$

$$= \arg\min_{\mathbf{X}(n)} \quad \|\mathbf{H}(\mathbf{X}(n) - \bar{\mathbf{X}})\|_{\ell_2}^2 \le r^2 \tag{82}$$

where (the MMSE estimate) $\bar{\mathbf{X}} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{Y}(n)$ is the center of the sphere of radius r. The ML solution is contained in this sphere and can be found via low–complexity tree based search algorithm [43]. This way an exhaustive search procedure is avoided and the complexity is independent of alphabet size.

4.2 An Expectation Maximization and smoothing approach to MIMO parameter recovery

The alternating ML detector and parameter estimation procedure outlined in Section 4.1 can also be performed by employing the Expectation Maximization (EM) framework [27]. Instead of maximizing the likelihood $p(\mathbf{Y}(n)|\mathbf{H}) = \sum_{\mathbf{X}(n)} p(\mathbf{Y}(n),\mathbf{X}(n)|\mathbf{H})$ EM works with the complete likelihood $p(\mathbf{Y}(n),\mathbf{X}(n)|\mathbf{H})$. Of course, the complete likelihood can not be evaluated, since the data $\mathbf{X}(n)$ are unknown. Instead the expected value is used. The conditional log likelihood

$$\log p(\mathbf{Y}(n)|\mathbf{H}) = \log p(\mathbf{Y}(n),\mathbf{X}(n)|\mathbf{H}) - \log p(\mathbf{X}(n)|\mathbf{Y}(n),\mathbf{H})$$

is employed to evaluate the estimated complete log–likelihood

$$\begin{aligned} Q\left(\mathbf{H},\mathbf{H}^{(\ell-1)}\right) &= \mathbb{E}_{p(\mathbf{X}(n)|\mathbf{Y}(n),\mathbf{H}^{(\ell-1)})}\left[\log p\left(\mathbf{Y}(n),\mathbf{X}(n)|\mathbf{H}\right)\right] \\ &= \sum_{\mathbf{X}(n)} p(\mathbf{X}(n)|\mathbf{Y}(n),\mathbf{H}^{(\ell-1)}) \log p(\mathbf{Y}(n),\mathbf{X}(n)|\mathbf{H}). \end{aligned} \tag{83}$$

The EM algorithm iterates between the E–step and the M–step until convergence (see Table 9). The expectation step (E–Step), where the conditional density of the unknown data, given the actual observations is estimated based on the current values of the unknown parameters is used to evaluate the expected value of the complete log–likelihood function. The maximization step (M–Step) finds the maximum of the estimated complete log–likelihood function with respect to the unknown system parameters.

Quite often in practice the number of available observations, n, is significantly smaller than $n_i M$, and the resulting system of equations is severely underdetermined. Furthermore, the effective rank of $\mathbf{H}$ is often significantly less than n. Such problems are often reduced by use of a Bayesian prior to favour some solutions over others. The prior is incorporated as a penalty term in the maximization step which is maximized to estimate the unknown system parameter matrix. Therefore the M–step seeks to solve the following problem:

$$\mathbf{H}^{(\ell)} = \arg\max_{\mathbf{H}} \quad Q\left(\mathbf{H},\mathbf{H}^{(\ell-1)}\right) + \log p\left(\mathbf{H}\right).$$

A widely used prior which promotes sparsity and avoids underdetermined problems is the Laplacian prior

$$p\left(\mathbf{H}\right) \propto \exp\left(-\tau\|\mathrm{vec}[\mathbf{H}]\|_{\ell_1}\right).$$

Table 9 EM Algorithm

Algorithm description	
$\ell = 0 : \mathbf{H}^{(0)}$	{Initiliazation}
Repeat	
$\ell = \ell + 1$	
1: $Q\left(\mathbf{H}, \mathbf{H}^{(\ell-1)}\right) = \mathbb{E}_{p(\mathbf{X}(n)\|\mathbf{Y}(n), \mathbf{H}^{(\ell-1)})}\left[\log p\left(\mathbf{Y}(n), \mathbf{X}(n)\|\mathbf{H}\right)\right]$	{E–Step}
2: $\mathbf{H}^{(\ell)} = \arg\max_{\mathbf{H}} Q\left(\mathbf{H}, \mathbf{H}^{(\ell-1)}\right)$	{M–Step}
Until $\|\text{vec}[\mathbf{H}^{(\ell)}] - \text{vec}[\mathbf{H}^{(\ell-1)}]\|_{\ell_2}^2 < \varepsilon$	{Termination Condition}

The introduction of such prior, allows the algorithm to choose only the non–zero components of $\mathbf{H}$.

The log–likelihood function increases monotonically at successive iterates $\mathbf{H}^{(\ell)}$ of the parameter vector [27], *i.e.*

$$p(\mathbf{Y}(n)|\mathbf{H}^{(\ell)}) \geq p(\mathbf{Y}(n)|\mathbf{H}^{(\ell-1)}).$$

Consequently the sequence $\{p(\mathbf{Y}(n)|\mathbf{H}^{(\ell)}), \quad \ell > 0\}$ converges as $\ell \to \infty$. For practical purposes, we truncate the number of iterations to a finite number L. Although the convergence of the likelihood values does not by itself ensure the convergence of the iterates $\mathbf{H}^{(\ell)}$, under relatively mild smoothness conditions for the log–likelihood function $p(\mathbf{Y}(n)|\mathbf{H})$ the sequence converges to a local maximum of $p(\mathbf{Y}(n)|\mathbf{H}^{(\ell)})$ [81]. However, its monotonic convergence behaviour is dependent on initialization [81]. To avoid it from being trapped to a stationary point which is not a local (global) maximum we may have to use several different initializations and also incorporate prior information about the distribution of $\mathbf{H}^{(0)}$. For Gaussian noise the corresponding log–likelihood function is log–concave. The log–concavity of the likelihood ensures convergence of the EM iteration to a stationary point, regardless of initialization.

Since $\mathbf{X}(n)$ is independent of $\mathbf{H}$ in Eq. (83), we can keep only the terms that depend on $\mathbf{H}$. Thus

$$Q\left(\mathbf{H}, \mathbf{H}^{(\ell-1)}\right) = \mathbb{E}_{p(\mathbf{X}(n)|\mathbf{Y}(n), \mathbf{H}^{(\ell-1)})}\left[\log p\left(\mathbf{Y}(n)|\mathbf{H}\right)\right]$$

with

$$p(\mathbf{Y}(n)|\mathbf{H}) = \frac{1}{(2\pi\sigma^2)^{n_o \times n}} \exp\left(-\frac{1}{2\sigma^2}\sum_{t=1}^{n} \|\mathbf{y}(t) - \mathbf{H}\mathbf{x}(t)\|_{\ell_2}^2\right).$$

Let us next take a closer look at the E–step. The resulting function, still denoted by Q takes the form

$$\mathcal{Q}\left(\mathbf{H},\mathbf{H}^{(\ell-1)}\right)=-\frac{1}{2\sigma^2}\sum_{t=1}^{n}\mathbb{E}\left\{\|\mathbf{y}(t)-\mathbf{H}\mathbf{x}(t)\|_{\ell_2}^2|\mathbf{y}(t),\mathbf{H}^{(\ell-1)}\right\}.$$

The E–step depends on first and second order statistics of the hidden variable $\mathbf{X}(n)$, which are not available since it is unknown. Therefore the complete log likelihood is given by

$$\mathcal{Q}\left(\mathbf{H},\mathbf{H}^{(\ell-1)}\right)=-\frac{1}{2\sigma^2}\sum_{t=1}^{n}\sum_{i}^{A^{n_i M}}\|\mathbf{y}(t)-\mathbf{H}\mathbf{x}(t)\|_{\ell_2}^2\gamma_{ti}^{(\ell)}$$

where

$$\gamma_{ti}^{(\ell)}=p\left(\mathbf{x}(t)=s_i|\mathbf{Y}(n);\mathbf{H}^{(\ell-1)}\right)$$

and thus a primary goal of the E–step is to compute the *a posteriori probabilities* (APPs), $\gamma_{ti}^{(\ell)}$. These in turn are computed by the forward–backward recursions presented next.

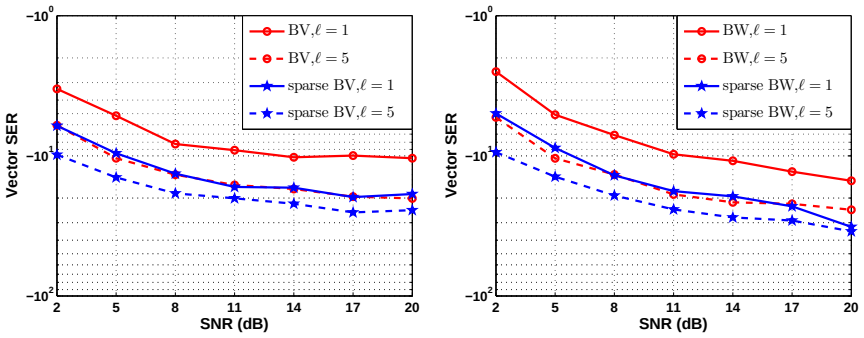
Maximization of the regularized Q–function with respect to $\mathbf{H}$ at the M–step, has a closed form expression and is given by the soft–thresholding function

$$\mathbf{H}^{(\ell)}=\mathbb{S}_{\tau}\left[\left(\sum_{i=1}^{A^{n_i M}}\mathbf{X}_i(n)\mathbf{X}_i^H(n)\gamma_{ni}^{(\ell)}\right)^{-1}\left(\mathbf{Y}(n)\left[\sum_{i=1}^{A^{n_i M}}\mathbf{X}_i^H(n)\gamma_{ni}^{(\ell)}\right]\right)\right] \tag{84}$$

The above is a convex problem and can be solved using linear programming methods, interior-point methods, and iterative thresholding [36, 76]. Note that the M–step can also be executed by Greedy algorithms.

Computation of smoothing probabilities. To implement the EM iteration, the APP's $\gamma_{ti}^{(\ell)}=p\left(\mathbf{x}(t)=s_i|\mathbf{Y}(n);\mathbf{H}^{(\ell-1)}\right)$ are needed. They correspond to the E–step of the EM algorithm, and they can be computed by soft decoders if the underlying structure of the MIMO system enables us to follow a Hidden Markov Model (HMM) formulation, where the APP's are expressed as functions of the transition probabilities. In such case the a posteriori distribution of the hidden variables is obtained using a two–stage message passing algorithm. In the context of HMM models, it is known as *forward–backward* algorithm [66], or *Baum–Welch*, or the *BCJR* algorithm [6].

A complete description by a HMM model requires a trellis diagram with state set $S=\{s_1,s_2,\ldots,s_{A^{M\times n_i}}\}$, where A is the alphabet size. The algorithm is split up into two stages. The first stage calculates the filtering probabilities $p(\mathbf{x}(t)=\cdots|\mathbf{Y}(t);\mathbf{H}^{(\ell-1)})$, while the second stage calculates the future probabilities $p(\mathbf{x}(t)=\cdots|\mathbf{Y}(t+1:n);\mathbf{H}^{(\ell-1)})$ (where $\mathbf{Y}(t+1:n)=[\mathbf{y}(t+1),\ldots,\mathbf{y}(n)]$. Assume that the two stages are already computed for all $t\in\{1,\ldots,n\}$. Then using the Markov chain rule we obtain the smoothing probabilities $p(\mathbf{x}(t)=s_i|\mathbf{Y}(n);\mathbf{H}^{(\ell-1)})$ for each $s_i\in S$



(a) Baum–Viterbi's performance (b) Baum–Welch's performance

Fig. 12 Symbol error rate (SER) for sparse MIMO systems

$$p(\mathbf{x}(n)=s_i|\mathbf{Y}(n);\mathbf{H}^{(\ell-1)})= \tag{85}$$
$$\underbrace{p(\mathbf{x}(t)=s_i|\mathbf{Y}(t-1);\mathbf{H}^{(\ell-1)})}_{\alpha_t(\mathbf{x}(t))}\underbrace{p(\mathbf{x}(t)=s_i)}_{b_t(\mathbf{x}(t),\mathbf{x}(t+1))}\underbrace{p(\mathbf{x}(t)=s_i|\mathbf{Y}(t+1:n);\mathbf{H}^{(\ell-1)})}_{\beta_{t+1}(\mathbf{x}(t))}$$

Next forward/backward recursions are derived that allow the probabilities of Eq. (85) efficiently. The filtering or forward probability $\alpha_t(\mathbf{x}(t))$ is obtained by summing all the lookahead probabilities as

$$\alpha_t(\mathbf{x}(t))=\sum_{\forall\mathbf{x}(t-1)\in S}\alpha_{t-1}(\mathbf{x}(t-1))b_{t-1}(\mathbf{x}(t-1),\mathbf{x}(t)). \qquad \text{(Forward Recursion)}$$

The derivation of the backward filtering is similar to the filtering probability

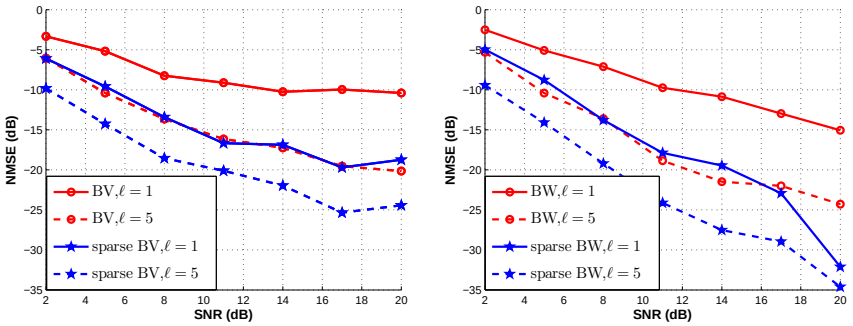
$$\beta_t(\mathbf{x}(t))=\sum_{\forall\mathbf{x}(t+1)\in S}\beta_{t+1}(\mathbf{x}(t+1))b_t(\mathbf{x}(t),\mathbf{x}(t+1)) \qquad \text{(Backward Recursion)}$$

b is determined from the received signal and a–priori information

$$b_t(\mathbf{x}(t),\mathbf{x}(t+1))=\exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{y}(t)-\mathbf{H}\mathbf{x}(t)\|_{\ell_2}^2\right\}Pr(\mathbf{x}(t+1)=s|\mathbf{x}(t)=s'). \tag{86}$$

4.3 Computer simulations of blind identification algorithms

In this subsection we compare the performance of the methods outlined here under two different operating modes: semi–blind and blind. Performance is measured in terms of Normalized Mean Square Error (NMSE, defined in Section 3.4) and Vector Symbol Error Rate (SER, that is the probability of at least one of the transmitted symbols is in error) for a frame of 100 vector symbols from BPSK constellations



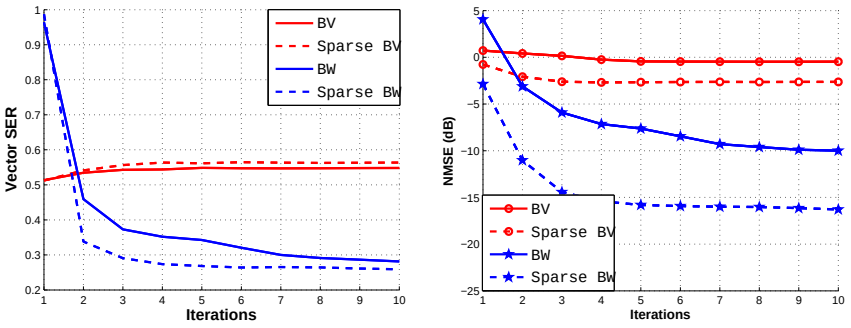
(a) Baum–Viterbi's performance (b) Baum–Welch's performance

Fig. 13 NMSE performance comparison under different noise conditions

averaged over 100 different system realizations. The non–zero coefficients are i.i.d. (independent, identically distributed) complex Gaussian random variables with zero mean and variance 1. The positions of the non–zero parameters are randomly selected in each realization, ensuring each output in non–zero. We consider a 2×2 linear MIMO system of memory length 4 and sparsity level 4.

We start by considering a semi–blind operation in which a short training sequence (consisting of five symbols) is available at the receiver side; the short training sequence is sent over the unknown system by the transmitter prior to the actual data transmission session. This training sequence, is used to initialize the algorithms. We note from inspection of Figs. 12, 13 that the performance of Baum–Viterbi's (sparse and non–sparse) is identical to Baum–Welch's (sparse and non–sparse) for an SNR range of $2-10$ dB; whereas in less noisy conditions Baum–Welch performs better. The conventional algorithms (Baum–Viterbi and Baum–Welch) la behind their sparse counterpart by approximately 5dB. We then inspect the vector SER for a maximum sequence detector Fig. 12 (a) and a maximum a posteriori detector Fig. 12 (b) where the sparse algorithms achieve better SER performance since they provide more accurate system estimates.

Next, we consider a blind operational mode where a key issue is how to acquire a reliable initial estimate for the parameter matrix. To avoid using different initial conditions we employ the single–spike strategy [28] in which all the parameters are set to zero except the dominant parameter which is set to ± 1, depending on its sign. By using this initialization, both algorithms converge in approximately 5 iterations. The algorithms are tested under a fixed noise condition (SNR 10dB). As it can be seen from Fig. 14, Baum–Viterbi fails to converge whereas Baum–Welch is more robust to initial conditions. For both algorithms (Baum–Viterbi and Baum–Welch) the sparse versions are better than the conventional counterparts and achieve faster convergence, see Fig. 14 (b).



(a) Symbol error rate evaluation (b) NMSE versus iterations

Fig. 14 Comparison of the two methods for fixed SNR of 10dB

5 Summary and future directions

In this chapter adaptive filtering and identification for multi input multi output nonlinear polynomial systems was considered. The exponential growth of complexity was addressed by sparsity aware schemes. Sparse LMS, RLS and greedy adaptive algorithms were described and their performance was demonstrated by simulations under a wide range of operating conditions. The above methods were combined with state estimation techniques such as the Viterbi family, in a semi–blind context. Alternative algorithms based on expectation maximization and smoothing methods were also discussed.

A number of other methods have been developed for linear systems. These include subspace methods, second order statistics and higher order statistics [59]. Adaptation of these methods to the nonlinear case and assessment of their performance is worth to pursue.

Acknowledgements This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: THALIS–UOA– SECURE WIRELESS NONLINEAR COMMUNICATIONS AT THE PHYSICAL LAYER

References

1. M. Abuthinien, S. Chen, and L. Hanzo. Semi-blind joint maximum likelihood channel estimation and data detection for MIMO systems. *IEEE Signal Processing Letters*, 15:202 –205, 2008.
2. G. Andrew and J. Gao. Scalable training of L1 regularized log linear models. In *International Conference on Machine Learning*, 2007.

3. D. Angelosante, J.A. Bazerque, and G.B. Giannakis. Online adaptive estimation of sparse signals: Where RLS meets the ℓ_1-norm. *IEEE Transactions on Signal Processing*, 58(7):3436 –3447, july 2010.
4. D. Angelosante and G.B. Giannakis. RLS-weighted lasso for adaptive estimation of sparse signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009.*, pages 3245 –3248, 2009.
5. B. Babadi, N. Kalouptsidis, and V. Tarokh. Sparls: The sparse RLS algorithm. *IEEE Transactions on Signal Processing*, 58(8):4013 –4025, aug. 2010.
6. L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Trans. Inf. Theory*, 20(2):284–287, Mar 1974.
7. J.R. Barry, E.A. Lee, and D.G. Messerschmitt. *Digital communication*. Springer, third edition, 2003.
8. S. Benedetto and S. Biglieri. *Principles of Digital Transmission: with wireless applications*. Kluwer Academic, 1998.
9. J. Benesty, T. Gnsler, Y. Huang, and M. Rupp. Adaptive algorithms for MIMO acoustic echo cancellation. In Yiteng Huang and Jacob Benesty, editors, *Audio Signal Processing for Next-Generation Multimedia Communication Systems*. Springer, 2004.
10. D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, Cambridge, 2003.
11. E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj, and V.H. Poor. *MIMO Wireless Communications*. Cambridge University Press, 2007.
12. P.T. Boufounos, B. Raj, and P. Smaragdis. Joint sparsity models for broadband array processing. In *Proc. SPIE Wavelets and Sparsity XIV*, 2011.
13. S. Boyd. *Volterra Series: Engineering Fundamentals*. PhD thesis, UC Berkeley, 1985.
14. S. Boyd and L. Chua. Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Transactions on Circuits and Systems*, 32(11):1150–1161, Nov. 1985.
15. J. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, 25(9):772 – 781, Sep 1978.
16. A.M. Bruckstein, D.L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.
17. E. Cands, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, 14:877–905, 2008.
18. Y. Chen, Y. Gu, and A.O. Hero. Sparse LMS for system identification. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009*, pages 3125 –3128, april 2009.
19. Y. Chen, Y. Gu, and A.O. Hero. Regularized Least-Mean-Square algorithms. *Arxiv preprint stat.ME/1012.5066v2*, 2010.
20. Fu-Hsuan Chiu. *Transceiver design and performance analysis of bit–interleaved coded MIMO–OFDM systems*. PhD thesis, University of Southern California, 2006.
21. P.L. Combettes and J.C. Pesquet. Proximal splitting methods in signal processing. In Heinz H. Bauschke, Regina S. Burachik, Patrick L. Combettes, Veit Elser, D. Russell Luke, and Henry Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer, 2011.
22. G.B. Giannakis D. Angelosante, E. Grossi and M. Lops. Sparsity–aware estimation of CDMA system parameters. *EURASIP Journal on Advances in Signal Processing*, 2010.
23. W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Trans. Inf. Theory*, 55(5):2230–2249, 2009.
24. I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.
25. I. Daubechies, M. Fornasier, and I. Loris. Accelerated projected gradient method for linear inverse problems with sparsity constraints. *Journal of Fourier Analysis and Applications*, 14:764–792, 2008.
26. S. Davis, G.M. Mallat and Z. Zhang. Adaptive time-frequency decompositions. *SPIE J. Opt. Engin.*, 33(7):2183–2191, 1994.

27. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J R Stat Soc B*, 39:1–38, 1977.
28. Z. Ding and Y. Li. *Blind Equalization and Identification*. Marcel Dekker, 2001.
29. A. Doicu, T. Trautmann, and F. Schreier. *Numerical Regularization for Atmospheric Inverse Problems*. Springer, 2010.
30. D.L. Donoho and I.M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika Trust*, 81(3):425 –455, Aug 1994.
31. D.L. Donoho, Y. Tsaig, I. Drori, and J.L. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. *Submitted for publication*.
32. F.J.III. Doyle, R.K. Pearson, and B.A. Ogunnaike. *Identification and control using Volterra series*. Springer, 2002.
33. J. Duchi, S. S.-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1 ball for learning in high dimensions. In *Proceedings of International Conference on Machine Learning (ICML 08)*, page 272279, 2008.
34. D.L. Duttweiler. Proportionate normalized Least-Mean-Squares adaptation in echo cancellers. *IEEE Trans. Speech Audio Processing*, 8(5):508–518, Sept. 2000.
35. E.M. Eksioglu and A.K. Tanc. RLS algorithm with convex regularization. *IEEE Signal Processing Letters*, 18(8):470 –473, aug. 2011.
36. M. Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.
37. Y. Ephraim and N. Merhav. Hidden markov processes. *IEEE Transactions on Information Theory*, 48(6):1518 –1569, jun 2002.
38. S. Werner T. Riihonen F. Gregorio, J. Cousseau and R. Wichman. Power amplifier linearization technique with iq imbalance and crosstalk compensation for broadband MIMO-OFDM transmitters. *EURASIP Journal on Advances in Signal Processing*, 2011.
39. M. Feder. *Statistical Signal Processing Using a Class of Iterative Estimation Algorithms*. PhD thesis, M.I.T., Cambridge, MA, 1987.
40. C. A. R. FERNANDES. *Nonlinear MIMO Communication Systems: Channel Estimation and Information Recovery using Volterra Models*. PhD thesis, Universite de Nice-Sophia Antipolis, 2009.
41. M.A.T. Figueiredo and R.D. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906 – 916, aug. 2003.
42. A. Gholami and S.M. Hosseini. A general framework for sparsity-based denoising and inversion. *IEEE Transactions on Signal Processing*, 59(11):5202 –5211, nov. 2011.
43. G. Giannakis, Z. Liu, X. Ma, and S. Zhou. *Space Time Coding for Broadband Wireless Communications*. Wiley-Interscience, 2003.
44. G.B. Giannakis and E. Serpedin. Linear multichannel blind equalizers of nonlinear FIR Volterra channels. *IEEE Transactions on Signal Processing*, 45(1):67 –81, jan 1997.
45. F. Gregorio, S. Werner, T.I. Laakso, and J. Cousseau. Receiver cancellation technique for nonlinear power amplifier distortion in SDMA–OFDM systems. *IEEE Transactions on Vehicular Technology*, 56(5):2499 –2516, Sept. 2007.
46. S.O. Haykin. *Adaptive Filter Theory*. Springer, fourth edition, 2001.
47. Y. Huang, J. Benesty, and J. Chen. *Acoustic MIMO Signal Processing*. Springer, 2006.
48. N. Hurley and S. Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723 –4741, oct. 2009.
49. Jian Jin, Yuantao Gu, and Shunliang Mei. A stochastic gradient approach on compressive sensing signal reconstruction based on adaptive filtering framework. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):409 –420, april 2010.
50. G.K. Kaleh and R. Vallet. Joint parameter estimation and symbol detection for linear or nonlinear unknown channels. *IEEE Trans. on Comm.*, 42(7):2606–2413, 1994.
51. N. Kalouptsidis. *Signal Processing Systems Theory and Design*. John Wiley and Sons, 1997.
52. Nicholas Kalouptsidis, Gerasimos Mileounis, Behtash Babadi, and Vahid Tarokh. Adaptive algorithms for sparse system identification. *Signal Processing*, 91(8):1910 – 1919, 2011.

53. T. Koike-Akino, A.F. Molisch, Man-On Pun, R. Annavajjala, and P. Orlik. Order-extended sparse RLS algorithm for doubly-selective MIMO channel estimation. In *IEEE International Conference on Communications (ICC), 2011*, pages 1 –6, june 2011.
54. Y. Kopsinis, K. Slavakis, and S. Theodoridis. Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls. *IEEE Transactions on Signal Processing*, 59(3):936 –952, march 2011.
55. L. Ljung. *General structure of adaptive algorithms: Adaptation and tracking*. in Adaptive System Identification and Signal Processing Algorithms, Eds. N. Kalouptsidis and S. Theodoridis, 1993.
56. R. Maleh and A.C Gilbert. Multichannel image estimation via simultaneous orthogonal matching pursuit. In *Proc. Workshop Stat. Signal Process.*, 2007.
57. P.Z. Marmarelis and V.Z. Marmarelis. *Analysis of Physiological Systems*. New York: Plenum Press, 1978.
58. G. Mileounis, B. Babadi, N. Kalouptsidis, and V. Tarokh. An adaptive greedy algorithm with application to nonlinear communications. *IEEE Transactions on Signal Processing*, 58(6):2998 –3007, june 2010.
59. G. Mileounis and N. Kalouptsidis. A sparsity driven approach to cumulant based identification,. In *IEEE International Workshop on Signal Processing Advances in Wireless Communications*, Jun. 2012.
60. G. Mileounis, N. Kalouptsidis, B. Babadi, and V. Tarokh. Blind identification of sparse channels and symbol detection via the EM algorithm. In *International Conference on Digital Signal Processing (DSP)*, pages 1 –5, july 2011.
61. Y. Murakami, M. Yamagishi, M. Yukawa, and I. Yamada. A sparse adaptive filtering using time-varying soft-thresholding techniques. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 3734 –3737, 2010.
62. D. Needell and J.A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comput. Harmon. Anal*, 26:301–321, 2009.
63. D. Needell and R. Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Found. Comput. Math*, 9(3):317–334, 2009.
64. C. Paleologu, J. Benesty, and S. Ciochina. *Sparse Adaptive Filters for Echo Cancellation*. Morgan and Claypool Publishers, 2010.
65. Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *27th Asilomar Conf.on Signals, Systems and Comput.*, pages 40–44, 1993.
66. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *IEEE Proc.*, 77(2):257–286, 1989.
67. A. J. Redfern and G. T. Zhou. Blind zero forcing equalization of multichannel nonlinear CDMA systems. *IEEE Transactions on Signal Processing*, 49(10):23632371, Oct. 2001.
68. C. Rizogiannis, E. Kofidis, C.B. Papadias, and S. Theodoridis. Semi-blind maximum-likelihood joint channel/data estimation for correlated channels in multiuser MIMO networks. *Signal Processing*, 90(4):1209 – 1224, 2010.
69. W.J. Rugh. *Nonlinear System Theory*. The Johns Hopkins University Press, 1981.
70. A.H. Sayed. *Adaptive Filters*. Wiley-Blackwell, 2008.
71. M. Schetzen. *The Volterra and Wiener Theories of Nonlinear Systems*. Willey and Sons, 1980.
72. C. Seretis. *Control-relevant identification for constrained and nonlinear systems*. PhD thesis, University of Maryland, 1997.
73. C. Tidestav and E. Lindskog. Bootstrap equalization. In *IEEE International Conference on Universal Personal Communications (ICUPC)*, volume 2, pages 1221 –1225, oct 1998.
74. 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.
75. J.A. Tropp, A.C. Gilbert, and M.J. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572 – 588, 2006.
76. J.A. Tropp and S.J. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6):948 –958, june 2010.

77. G.L. Sicuranza V.J. Mathews. *Polynomial Signal Processing*. Wiley-Blackwell, 2000.
78. Y. Wang, A.G. Yagola, and C. Yang. *Optimization and Regularization for Computational Inverse Problems and Applications*. Springer, 2010.
79. J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *J. Mach. Learn. Res.*, 3:1439–1461, 2003.
80. D.T. Westwick and R.E. Kearney. *Identification of nonlinear physiological systems*. IEEE Press, 2003.
81. C. Wu. On the convergence properties of the EM algorithm. *Ann. Statist.*, 11:95–103, 1983.
82. M. Yukawa. *Adaptive Filtering Based on Projection Method*. Block Seminar in Elite Master Study Course SIM, 2010.
83. Q. Zheng. *A Volterra series approach to nonlinear process control and control relevant identification*. PhD thesis, University of Maryland, 1995.
84. Q. Zheng and E. Zafiriou. Volterra Laguerre models for nonlinear process identification with application to a fluid catalytic cracking unit. *Industrial & Engineering Chemistry Research*, 43(2):340–348, 2004.