Graz University of Technology

Doctoral Thesis

Adaptive Digital Predistortion of Nonlinear Systems

Li Gan

Faculty of Electrical and Information Engineering
Graz University of Technology, Austria

First Examiner:
Univ.-Prof. Dipl.-Ing. Dr.techn. Gernot Kubin
Graz University of Technology, Austria

Second Examiner:
Prof. Giovanni Sicuranza
University of Trieste, Italy

Co-Advisor:
Dr. Emad Abd-Elrady
Graz University of Technology, Austria

Graz, April 2009

Kurzfassung

Die Kompensation oder Reduktion von Nichtlinearen Verzerrungen, welche üblicherweise von Nichtlinearen Systemen resultieren, wird zunehmend eine unentbehrliche Voraussetzung in vielen Anwendungsgebieten. In dieser Arbeit werden digitale Vorverzerrungtechniken für eine breite Klasse von Nichtlinearen Systemen präsentiert. Für die Parameterschätzung der Vorverzerrung werden verschiedene Architekturen behandelt: die direkte Lernarchitektur (Direct Learning Architecture, DLA) und die indirekte Lernarchitektur (Indirect Learning Architecture, ILA). Für den DLA Ansatz schlagen wir einen neuen Adaptionsalgorithmus vor (Nonlinear Filtered-x Prediction Error Method, NFxPEM), welcher den konventionellen Algorithmus (Nonlinear Filtered-x Least Mean Squares, NFxLMS) bezüglich Konvergenzgeschwindigkeit und Leistungsfähigkeit übertrifft. All diese Algorithmen operieren im Zeitbereich und benötigen eine genaue Identifikation des Nichtlinearen Systems. Um diese strenge Einschränkung abzuschwächen oder ganz zu vermeiden wurden die Algorithmen NFxLSM und NFxPEM mit einer einfachen, initialen Teilsystem Identifikationsmethode (Initial Subsystem Estimates, ISE) zu NFxLMS-ISE und NFxPEM-ISE erweitert. Weiters schlagen wir eine Vorverzerrungsmethode im Frequenzbereich vor, welche das Betragsspektrum am Ausgang eines Nichtlinearen Systems kompensiert (Spectral Magnitude Matching, SMM).

Der ILA Ansatz ist in ILA-I und ILA-II Ansätze unterteilt, für die eine rekursive Fehlerprädiktionsmethode (Recursive Prediction Error Method, RPEM) vorgeschlagen wird. Für den ILA-I Ansatz kann der RPEM Algorithmus das durch Nichtlinearitäten verursachte neuerliche Anwachsen von bereits zur Bandbegrenzung unterdrückten Spektralkomponenten (spectral regrowth) reduzieren und die Nichtlinearen Verzerrungen kompensieren. Für den ILA-II Ansatz kann der RPEM Algorithmus die Leistungsfähigkeit der Vorverzerrung im Vergleich zu konventionellen LMS (Least Mean Squares) Algorithmen stark verbessern. Für die Implementierung dieser Algorithmen wird für die Berechnung des Gradienten eine allgemeine Architektur (General Gradient Calculation Architecture, GGCA) für verschiedene Nichtlineare Systeme vorgeschlagen. Schließlich wenden wir diese Techniken für die Vorverzerrung von Nichtlinearen Modellen an, welche in existierenden Kommunikationssystemen eingesetzt werden. Diese Modelle sind zum Beispiel das parallele Wiener Modell (parallel Wiener-type model) und das Modell mit gedächtnisbehafteten Polynomen (memory polynomial model).

Abstract

Compensating or reducing the nonlinear distortion - usually resulting from a nonlinear system - is becoming an essential requirement in many areas. In this thesis adaptive digital predistortion techniques for a wide class of nonlinear systems are presented. For estimating the coefficients of the predistorter, different learning architectures are considered: the Direct Learning Architecture (DLA) and Indirect Learning Architecture (ILA). In the DLA approach, we propose a new adaptation algorithm - the Nonlinear Filtered-x Prediction Error Method (NFxPEM) algorithm, which has much faster convergence and much better performance compared to the conventional Nonlinear Filtered-x Least Mean Squares (NFxLMS) algorithm. All of these time domain adaptive algorithms require accurate system identification of the nonlinear system. In order to relax or avoid this strict requirement, the NFxLMS with Initial Subsystem Estimates (NFxLMS-ISE) and NFxPEM-ISE algorithms are proposed. Furthermore, we propose a frequency domain predistortion technique - the Spectral Magnitude Matching (SMM) method. The ILA approach is classified into ILA-I and ILA-II approaches and the Recursive Prediction Error Method (RPEM) algorithm is proposed. In the ILA-I approach, the RPEM algorithm can well reduce the spectral regrowth and compensate the nonlinear distortion. Also, using the RPEM algorithm in the ILA-II approach can greatly improve the performance of the predistorter, compared to the traditional Least Mean Squares (LMS) algorithm. For implementation of these algorithms, General Gradient Calculation Architectures (GGCAs) are proposed for different nonlinear systems. Finally we apply these techniques for the predistortion of some nonlinear models used in practical communication systems, *e.g.*, the parallel Wiener-type model and the memory polynomial model.

Acknowledgement

The research work for this doctoral thesis was carried out within a cooperation between Infineon Technologies, Villach and the Christian Doppler Laboratory for Nonlinear Signal Processing at the Signal Processing and Speech Communication Laboratory, Graz University of Technology. I am grateful to Infineon Technologies and the Christian Doppler Association for the financial support of this work.

Special thanks go to my supervisor, Professor Gernot Kubin, for leading me to the nonlinear signal processing area with his excellent guidance, support and encouragement. Furthermore, I would like to thank Dr. Emad Abd-Elrady, who cooperates with me in the project and gives me lots of help. I would also like to thank Professor Giovanni Sicuranza for being my second examiner for this thesis.

Many thanks to my family for their continuous support and to all my friends in China and Austria. Finally, I would like to thank everybody in SPSC for the wonderful time we had during my PhD study.

Graz, April 2009 Li Gan

Contents

List of Abbreviations

AWGN	Additive White Gaussian Noise
CC	Computational Complexity
CDMA	Code Division Multiple Access
DFT	Discrete Fourier Transform
DLA	Direct Learning Architecture
DSP	Digital Signal Processing
FDMA	Frequency Division Multiple Access
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FRF	Frequency Response Function
FT	Fourier Transform
IDFT	Inverse Discrete Fourier Transform
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
IIR	Infinite Impulse Response
ILA	Indirect Learning Architecture
ISE	Indirect Subsystem Estimate
KF	Kalman Filter
LD	Line Driver
LS	Least Squares
LMS	Least Mean Squares
LUT	Look Up Table
MSD	Mean Square Distortion
MSE	Mean Square Error
MSND	Mean Square Nonlinear Distortion

NFxLMS	Nonlinear Filtered-x Least Mean Squares
NFxLMS-ISE	Nonlinear Filtered-x Least Mean Squares with Initial Subsystem Estimate
NFxPEM	Nonlinear Filtered-x Prediction Error Mehtod
NFxPEM-ISE	Nonlinear Filtered-x Prediction Error Method with Initial Subsystem Estimate
NFxRLS	Nonlinear Filtered-x Recursive Least Squares
PA	Power Amplifier
PSD	Power Spectral Density
RF	Radio-Frequency
RLS	Recursive Least Squares
RMS	Root Mean Square
RPEM	Recursive Prediction Error Method
SMM	Spectral Magnitude Matching
SPR	Strictly Positive Real
SNR	Signal-to-Noise Ratio

Chapter 1

Introduction

Nowadays, cancelling or reducing the effects of nonlinear distortion is an essential requirement in many areas. In wireless communication systems, *e.g.*, frequency division multiple access (FDMA) and code division multiple access (CDMA) systems, power amplifiers (PAs) are often driven into their nonlinear region in order to increase the efficiency [1,2,3,4]. The nonlinearity of the PA will warp the signal constellation, generate spectral regrowth and distort the signal pulse shape. In optical communication systems, nonlinear distortion caused by laser diodes should be reduced in order to satisfy the system requirements [5, 6]. In audio systems, the loudspeaker has several major sources of nonlinearity [7,8] and the small distortion caused by nonlinear components can dominate the overall performance. Other examples can be found in integrated filters, radio systems, speech processing and control engineering, see [9, 10, 11, 12].

The nonlinearity can be described using different kinds of nonlinear models, *e.g.*, Volterra, Wiener and Hammerstein models. Volterra series [13, 14] is a general model for nonlinear systems, which can be described as

$$z(n) = h_0 + \sum_{k=1}^{\infty} \sum_{i_1=-\infty}^{\infty} \sum_{i_2=-\infty}^{\infty} \cdots \sum_{i_k=-\infty}^{\infty} h_{i_1,\cdots,i_k} y(n-i_1) \cdots y(n-i_k) \tag{1.1}$$

where $y(n)$ and $z(n)$ are the input and output signals, respectively, and $h_{i_1,\cdots,i_k}$ is the kth-order Volterra kernel of the system. If $h_{i_1,\cdots,i_k} = 0$ for all $i_k < 0$, the Volterra system is causal, and (1.1) becomes

$$z(n) = h_0 + \sum_{k=1}^{\infty} \sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} \cdots \sum_{i_k=0}^{\infty} h_{i_1,\cdots,i_k} y(n-i_1) \cdots y(n-i_k). \tag{1.2}$$

Normally, the memory length required to approximate a nonlinear system is finite, and the Volterra series is truncated to a finite order q, a finite-memory, finite-order Volterra series is obtained as

$$z(n) = h_0 + \sum_{k=1}^{q} \sum_{i_1=0}^{M_k-1} \sum_{i_2=0}^{M_k-1} \cdots \sum_{i_k=0}^{M_k-1} h_{i_1,\cdots,i_k} y(n-i_1) \cdots y(n-i_k) \tag{1.3}$$

where M_k is the memory length. However, the main problem encountered while using Volterra models is high computational complexity due to the large number of parameters. For this

reason, block-structured models such as Winer and Hammerstein models are considered in order to decrease the number of parameters - hence decrease the computational complexity.

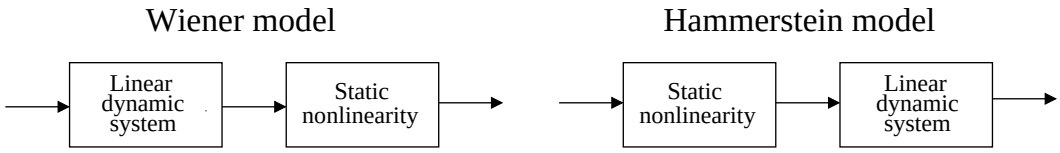

Figure 1.1 Wiener and Hammerstein models.

The Wiener and Hammerstein models are particular cases of the truncated Volterra series [14], the Wiener model structure consists of a linear dynamic system followed by a static nonlinearity. On the other hand, in the Hammerstein model structure the static nonlinearity precedes the linear dynamic system, see Fig. 1.1. For these nonlinear models, it is assumed that only the input and the output signals of the model are measurable.

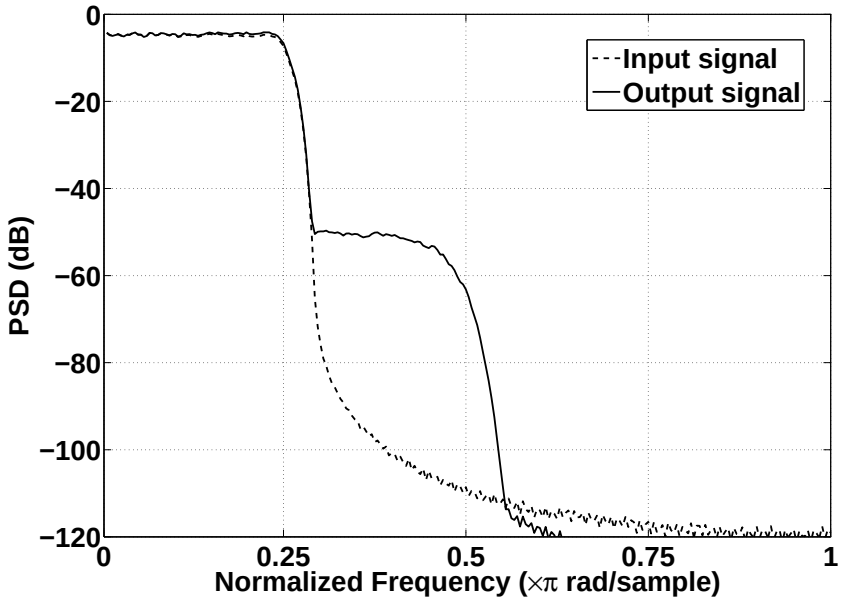

Figure 1.2 Nonlinear distortion.

The distortion caused by the nonlinearity can be easily observed in frequency domain. If a system is described by a nonlinear model, the output singal of the nonlinear model will contain new frequency components, namely spectral regrowth, compared to the input signal. Fig. 1.2 shows the Power Spectral Densities (PSDs) of the input and output signals of a Volterra system. The dashed line represents the PSD of the input signal, and the solid line represents the PSD of the output signal. It is very obvious that there is significant spectral regrowth in the normalized frequency band $(0.30\pi, 0.55\pi)$.

In order to reduce the spectral regrowth - hence to reduce or compensate the nonlinear distortion, there are two kinds of linearization techniques: predistortion and post-distortion [15], see Fig. 1.3. A nonlinear filter can be connected in cascade before the nonlinear system

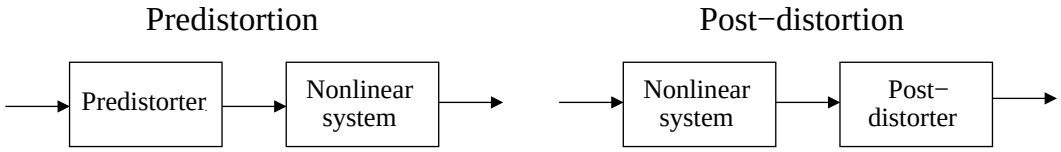

Figure 1.3 Predistortion and post-distortion.

(called predistorter) or after the nonlinear system (called post-distorter or equalizer), which results in an overall system whose characteristics correspond to a reference linear system, in the range of input signals of interest and in the desired frequency band. In many applications, predistortion is more efficient than post-distortion. For example, in wireless communication systems, PA is an analog device and its output is a radio signal. Therefore, implementing post-distortion needs to include a nonlinear filter (usually adaptive) in the analog domain which is difficult and expensive. In this case, predistortion is more suitable since it can implement the predistorter in the digital domain.

This thesis has three primary objectives: first, to introduce new and robust adaptation algorithms for estimating the coefficients of the predistorter. Second, to discuss the important aspects during the implementation of adaptive nonlinear predistortion. Third, to utilize the proposed adaptation algorithms in selected areas of telecommunications.

In the next section, the motivation for implementing adaptive nonlinear predistortion will be further elaborated.

1.1. Motivation of This Thesis

Predistortion of nonlinear systems was first studied in [16]. The proposed pth-order inverse technique can remove nonlinear distortion up to pth order, see Appendix A. However, this method is a non-adaptive technique to estimate the coefficients of the predistorter. Adaptive predistortion is usually required in implementation in case of time-varying and/or unknown nonlinear systems. Adaptive predistortion is first considered in digital radio systems [17] based on adaptive Lookup Table (LUT) technique, but the predistorter using the LUT technique is usually restricted to particular modulation formats. Adaptive predistortion for PAs has been considered in [18, 19] but without considering the existence of memory effects in PAs. However, recently research results have indicated that the memory effects in an PA could seriously affect the performance of wireless communication systems [20, 21, 22], and the updating techniques for a memoryless predistorter in [18, 19] can't be extended to update nonlinear predistorters for PAs with memory. Therefore, adaptive predistortion techniques for nonlinear physical system models with memory, such as Volterra models etc., should be investigated.

In adaptive predistortion, two important aspects needs to be considered for finding the coefficients of the predistorter:

- Learning architecture
- Adaptation algorithm

In general, there are two learning architectures: the direct learning architecture (DLA) and the indirect learning architecture (ILA).

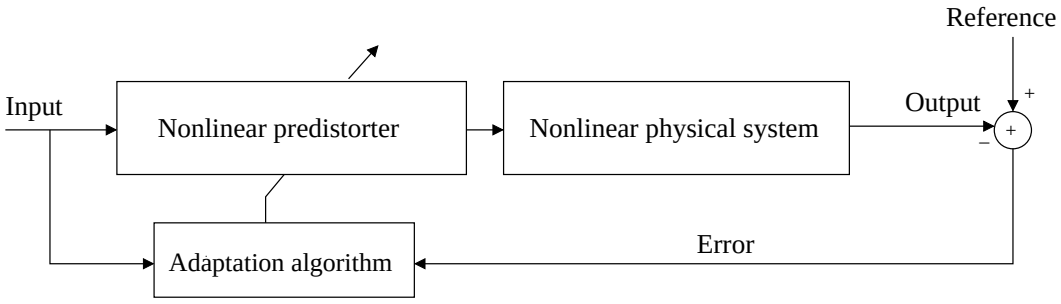

Figure 1.4 The DLA approach.

The basic scheme of the DLA approach is given in Fig. 1.4. The nonlinear predistorter is connected tandemly with the nonlinear physical system. The coefficients of the predistorter are estimated directly using the feedback error signal and adaptation algorithms. The most commonly used adaptation algorithm is the Nonlinear Filtered-x Least Mean Squares (NFxLMS) algorithm [23, 24, 25]. However, the NFxLMS algorithm usually suffers from slow convergence. The Nonlinear Filtered-x Recursive Least Squares (NFxRLS) algorithm has been proposed in [26] in order to speed up the convergence, but it can only be derived for the scenario where the outputs of the the nonlinear physical system and the predistorter are linear in their coefficients. The challenge is to find an adaptation algorithm with fast convergence and suitable for predistortion regardless the model type of the nonlinear physical system and predistorter. Besides, all mentioned algorithms require adaptive system identification of the nonlinear physical system prior to adaptation of the predistorter. To relax or avoid this strict requirement is another interesting topic to investigate.

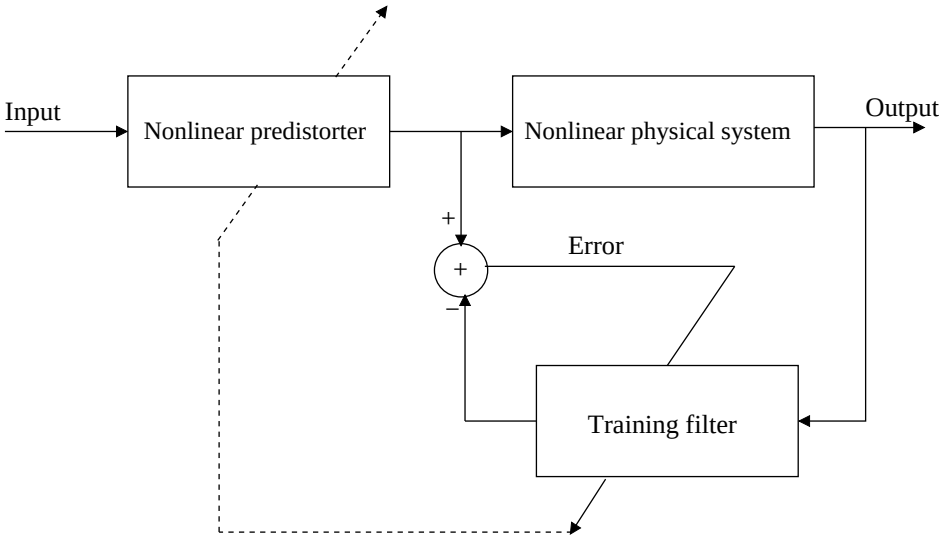

Figure 1.5 The ILA-I approach.

The ILA approaches can be classified into two kinds: the ILA-I approach and the ILA-II approach. The ILA-I approach is demonstrated in Fig. 1.5. The coefficients of the predistorter are a copy of the coefficients of the training filter connected as a post-distorter to the

nonlinear physical system. The coefficients of the training filter are estimated using the error signal and adaptation algorithms [27, 28, 29, 30, 31]. In [31], the training filter is modeled as a Volterra system, a static polynomial system and a memory polynomial system, respectively. The well known Least Squares (LS) method is used to evaluate its coefficients. In [28], the training filter is modeled by combining the memory polynomial model with an envelope memory term. The LS method is also used to evaluate its coefficients. However, the noisy measurement of the output of the nonlinear physical system makes the training filter converge to a biased estimate and hence degrade the performance of the ILA-I approach. The publications [29, 30] have tried to solve this problem by proposing a modified configuration or new methods to update the coefficients of the memory polynomial predistorter directly. In [27], the Recursive Least Squares (RLS) algorithm has been proposed, where both the training filter and the predistorter are modeled as Volterra systems. Adaptation algorithms suitable for other nonlinear models should be investigated as well.

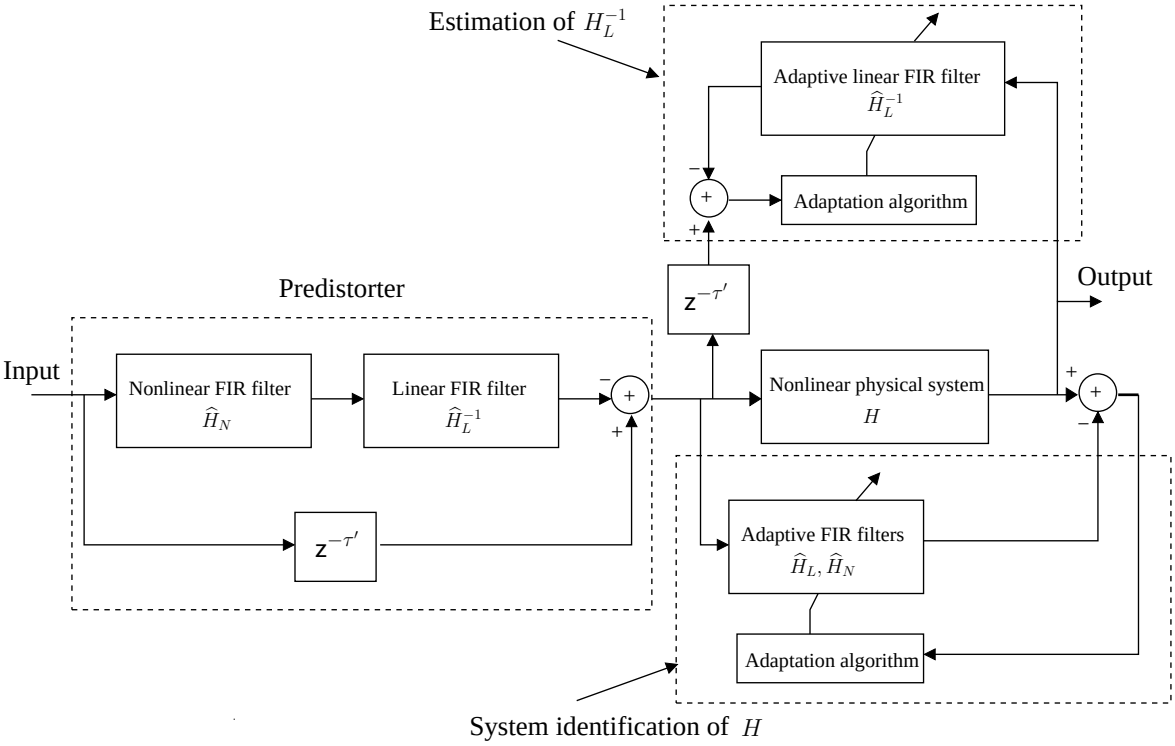

Figure 1.6 The ILA-II approach.

The ILA-II approach is first proposed in [9] for predistortion of Volterra systems. The structure of this approach is given in Fig. 1.6. The Volterra system H is assumed to be a weakly nonlinear system and can be divided into two subsystems, one is the purely linear subsystem H_L and the other is the purely nonlinear subsystem H_N, where $H = H_L + H_N$. The predistorter can be constructed using H_N, the inverse of the purely linear subsystem H_L^{-1} and the delayed input signal. In order to construct the predistorter, first, the subsystems H_L, H_N are identified by using adaptive linear and nonlinear FIR filters [32], $\widehat{H}_L$ and $\widehat{H}_N$. H_L^{-1} is estimated directly using an adaptive linear FIR filter $\widehat{H}_L^{-1}$. Note that the input of $\widehat{H}_L^{-1}$

should be the output of $\widehat{H}_L$, and it can use the output of H under the assumption that H is a weakly nonlinear system. Then, the predistorter is constructed by copying the estimated coefficients from $\widehat{H}_N$ and $\widehat{H}_L^{-1}$ to the predistorter.

The suggested adaptation algorithm for estimating the coefficients of H_L, H_N and H_L^{-1} in [9] is the Least Mean Squares (LMS) algorithm. However, since identifying Volterra systems using the LMS algorithm usually provides inaccurate estimates [33] due to slow convergence, inaccurate estimates of H_N and H_L^{-1} will then degrade the performance of the predistorter. Finding an adaptation algorithm to obtain more accurate estimates and hence to improve the performance of the predistorter is an important issue.

Most of the existing adaptation algorithms based on these two architectures are time domain adaptation algorithms, predistortion using frequency domain adaptation algorithms is also an interesting topic to investigate since predistortion aims to reduce the spectral regrowth in frequency domain.

1.2. Thesis Outline and Contributions

In chapter 2, the adaptation algorithms for predistortion using the DLA approach will be covered. The existing time domain algorithms, *i.e.*, the NFxLMS and NFxRLS algorithms are first reviewed. Then, the Nonlinear Filtered-x Prediction Error Method (NFxPEM) algorithm is proposed for predistortion of different nonlinear physical system models. Both the NFxLMS and NFxPEM algorithms require accurate identification of the nonlinear physical system. In order to avoid or relax this requirement, we propose the NFxLMS with Initial Subsystem Estimates (NFxLMS-ISE) and NFxPEM with Initial Subsystem Estimates (NFxPEM-ISE) algorithms, using the Initial Subsystem Estimate (ISE) method instead of accurate system identification.

A frequency domain predistortion technique, the Spectral Magnitude Matching (SMM) method, is also proposed in this chapter to compensate the nonlinear distortion.

The contributions of chapter 2 have been previously presented in the following publications:

- E. Abd-Elrady and L. Gan: Direct predistortion of nonlinear systems using adaptive Volterra systems and prediction error method. Submitted to *IEEE Signal Processing letters*. [34]
- E. Abd-Elrady and L. Gan: Direct predistortion of Hammerstein and Wiener systems using prediction error method. Submitted to *Signal Processing*. [35]
- E. Abd-Elrady and L. Gan: Direct linearization of weakly nonlinear Volterra systems using adaptive linear and nonlinear FIR filters. In *Proceeding of the IFAC Symposium on System Identification (SYSID'09)*, Saint-Malo, France, July 6-8, 2009. [36]
- E. Abd-Elrady, L. Gan and G. Kubin: Adaptive Predistortion of Nonlinear Volterra Systems using Spectral Magnitude Matching. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'09)*, Taiwan, April 19-24, 2009, pp. 2985-2988. [37]
- L. Gan and E. Abd-Elrady: Adaptive predistortion of Wiener systems using the NFxLMS algorithm and initial subsystem estimates. In *Proceedings of the European Signal Processing Conference (EUSIPCO'08)*, Lausanne, Switzerland, August 25-29, 2008. [38]

- L. Gan and E. Abd-Elrady: Adaptive predistortion of IIR Hammerstein systems using the nonlinear filtered-x LMS algorithm. In *Proceedings of the IEEE Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP'08)*, Graz, Austria, July 23-25, 2008, pp. 702-705. [39]

The main ideas of [34, 35, 36, 37] came from Dr. E. Abd-Elrady and L. Gan made major contributions to the simulation section and performance analysis. The publications [38, 39] were based on L. Gan's idea and simulation results, under Dr. E. Abd-Elrady's supervision.

Chapter 3 introduces the adaptation algorithms for the ILA approach. The existing RLS algorithm for predistortion of Volterra systems using the ILA-I approach is first reviewed. Within the same learning architecture, the Kalman Filter (KF) and Recursive Prediction Error Method (RPEM) algorithms are then derived for predistortion of Volterra systems. The RPEM algorithm is also derived for predistortion of Wiener and Hammerstein systems. In the ILA-II approach, the RPEM algorithm is used instead of the LMS algorithm to improve the performance of the predistorter.

The contributions of chapter 3 have been previously presented in the following publications:

- E. Abd-Elrady, L. Gan and G. Kubin: Direct and Indirect Learning Methods for Adaptive Predistortion of IIR Hammerstein Systems. In *e&i Elektrotechnik und Informationstechnik Special issue on Analog & Mixed Signal-Schaltungen und -Systeme*, 125/4: 126-131, 2008. [40]

- L. Gan and E. Abd-Elrady: Linearization of weakly nonlinear systems using adaptive FIR filters and recursive prediction error method. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP'08)*, Cancun, Mexico, October 16-19, 2008, pp. 409-414. [41]

- E. Abd-Elrady and L. Gan: Adaptive predistortion of Hammerstein systems based on indirect learning architecture and prediction error method. In *Proceedings of the IEEE International Conference on Signals and Electronic Systems (ICSES'08)*, Krakow, Poland, September 14-17, 2008, pp. 389-392. [42]

- E. Abd-Elrady, L. Gan and G. Kubin: Distortion compensation of nonlinear systems based on indirect learning architecture. In *Proceedings of the IEEE International Symposium on Communications, Control and Signal Processing (ISCCSP'08)*, St. Julians, Malta, March 12-14, 2008, pp. 184-187. [43]

The publication [40] was the combination of the previous works from Dr. E. Abd-Elrady and L. Gan, supervised by Prof. G. Kubin. The main ideas of [42, 43] came from Dr. E. Abd-Elrady and L. Gan made major contributions to the simulation section and performance analysis. The publication [41] was based on L. Gan's idea and simulation results, under Dr. E. Abd-Elrady's supervision.

Chapter 4 proposed the General Gradient Calculation Architecture (GGCA) for predistortion of Volterra, Wiener and Hammerstein systems, respectively. The GGCA is a common structure to calculate the gradient vector required in the adaptation algorithms using the DLA approach and the ILA-I approach. Also, several important aspects of adaptive predistorter design are discussed. At the end of the chapter, a comparison of all the existing and proposed adaptation algorithms is presented.

In Chapter 5, the proposed adaptation algorithms are utilized for predistortion of some specific nonlinear systems, *e.g.*, the parallel Wiener-type system and memory polynomial system.

The contributions of chapter 5 have been previously presented in the following publications:

- L. Gan and E. Abd-Elrady: Digital Predistortion of Memory Polynomial Systems using Direct and Indirect Learning Architectures. In *Proceedings of the IASTED Conference on Signal and Image Processing (SIP'09)*, Honolulu, Hawaii, USA, August 17-19, 2009. [44]

- L. Gan, E. Abd-Elrady and G. Kubin: Nonlinear distortion compensation for parallel Wiener-type systems using predistorter and direct learning architecture. In *Proceedings of the IEEE Digital Signal Processing Workshop (DSP'09)*, Marco Island, FL, USA, January 4-7, 2009, pp. 72-77. [45]

- L. Gan and E. Abd-Elrady: Digital predistortion of parallel Wiener-type systems using the RPEM and NFxLMS algorithms. In *Proceedings of the IEEE International Conference on Signal Processing (ICSP'08)*, Beijing, China, October 26-29, 2008, pp. 149-152. [46]

These publications were based on L. Gan's idea and simulation results, under Dr. E. Abd-Elrady and/or Prof. G. Kubin's supervision.

Furthermore, the author contributed to the simulation section of a publication which is beyond the scope of this thesis, which is

- E. Abd-Elrady and L. Gan: Identification of Hammerstein and Wiener Models using Spectral Magnitude Matching. In *Proceedings of the IFAC World Congress on Automatic Control*, Seoul, Korea, July 6-11, 2008, pp. 6440-6445. [47]

Chapter 2

Predistortion Using the Direct Learning Architecture (DLA)

In this chapter the nonlinear physical system is modeled as Volterra, Wiener and Hammerstein systems, respectively, and the adaptation algorithms based on the Direct Learning Architecture (DLA) approach are presented to estimate the coefficients of the predistorter. The existing adaptation algorithms, such as the Nonlinear Filtered-x Least Mean Squares (NFxLMS) and the Nonlinear Filtered-x Recursive Least Squares (NFxRLS) algorithms, are first reviewed. Then new adaptation algorithms are developed in this chapter, such as the Nonlinear Filtered-x Prediction Error Method (NFxPEM), the NFxLMS with Initial Subsystem Estimates (NFxLMS-ISE), the NFxPEM with Initial Subsystem Estimates (NFxPEM-ISE) algorithms, and the Spectral Magnitude Matching (SMM) method.

This chapter is based on the publications [34, 35, 36, 37, 38, 39, 40, 48], which are edited and refined in order to fit the current style of the thesis.

2.1. Introduction

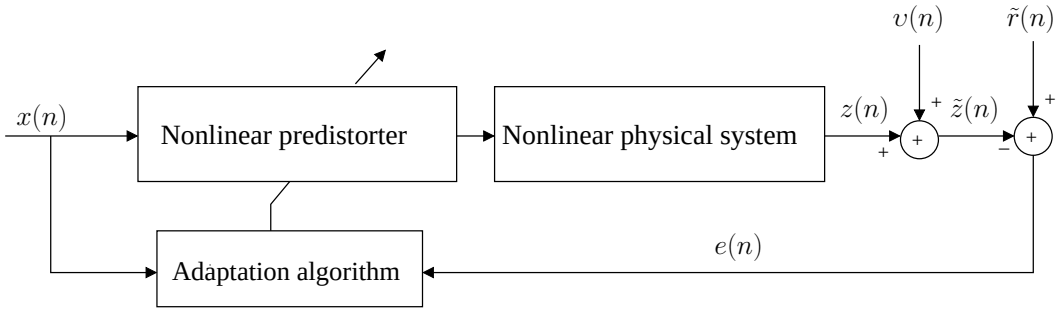

Figure 2.1 Predistortion using the DLA approach.

In the DLA approach, see Fig. 2.1, the nonlinear predistorter is connected tandemly with the nonlinear physical system. The coefficients of the predistorter are estimated directly using the feedback error signal $e(n)$ and adaptation algorithms. The error signal $e(n)$ is the difference between the reference signal $r(n)$ and the noisy measurement of the system output

$\tilde{z}(n)$, defined as

$$\tilde{z}(n) = z(n) + \upsilon(n) \tag{2.1}$$

where $z(n)$ is the clean output of the nonlinear physical system and $\upsilon(n)$ is zero-mean Additive White Gaussian Noise (AWGN). The reference signal $\tilde{r}(n)$ is usually defined as $\tilde{r}(n) = x(n - \tau)$, where τ is the time delay caused by the overall system consisting of the predistorter and the nonlinear physical system.

Remark 2.1: The delay time τ equals zero in case the linear subsystem of the nonlinear physical system is minimum phase [13, 23].

Therefore, the error signal can be written as

$$e(n) = \tilde{r}(n) - \tilde{z}(n) = \tilde{r}(n) - \upsilon(n) - z(n). \tag{2.2}$$

Since $\tilde{r}(n) - \upsilon(n) = \tilde{r}(n) + \tilde{\upsilon}(n)$ where $\tilde{\upsilon}(n) = -\upsilon(n)$ is also AWGN, for convenience, we redefine the reference signal as

$$r(n) = \tilde{r}(n) + \tilde{\upsilon}(n). \tag{2.3}$$

Consequently, the error signal $e(n)$ becomes the difference between the new defined reference signal $r(n)$ and the clean system output $z(n)$, written as

$$e(n) = r(n) - z(n). \tag{2.4}$$

Several adaptation algorithms using the DLA approach have been proposed [23, 24, 25, 26], and all of these algorithms are time domain adaptation algorithms. In [23], the nonlinear physical system is modeled as a Volterra system, the predistorter is also modeled as a Volterra system and the NFxLMS algorithm is proposed to estimate the coefficients of the predistorter. In [24], the NFxLMS algorithm is derived for predistortion of Wiener systems, where the nonlinear physical system is modeled as a Wiener system and the predistorter is modeled as a Hammerstein system. In [25], the NFxLMS algorithm is applied for predistortion of Power Amplifiers (PAs) modeled as Wiener systems in wireless communication systems.

The NFxLMS algorithm estimates the coefficients of the predistorter by minimizing the Mean Square Error (MSE) defined as

$$\mathrm{E}\{e^2(n)\} = \mathrm{E}\{(r(n) - z(n))^2\} \tag{2.5}$$

where $\mathrm{E}\{.\}$ denotes the Expectation. The NFxLMS algorithm is an extension of the Filtered-x LMS (FxLMS) algorithm, which is widely used in active noise control. The FxLMS algorithm is first developed in [49] and the performance of the FxLMS algorithm has been studied in [50, 51, 52, 53, 54]. In [53], the nonlinear physical system is described using a scaled error function. It is shown that the steady-state MSE of the FxLMS algorithm highly depends on the degree of nonlinearity of the nonlinear physical system. The error signal $e(n)$ in steady state increases with the degree of nonlinearity. Therefore, the NFxLMS algorithm is expected to provide inaccurate estimates of the predistorter. Also, Least Mean Squares (LMS) type algorithms usually have slow convergence since increasing the *step size* parameter leads to instability problems [55]. In order to speed up the convergence of the adaptation, the NFxRLS algorithm is proposed for predistortion of Volterra systems in [26]. The NFxRLS algorithm is derived using the property that the output of the Volterra predistorter is linear in the parameters and the nonlinear physical system is a weakly nonlinear system. Hence, it

is difficult to derive the NFxRLS algorithm for other predistorter models, where the output is not linear in the parameters.

In this chapter, the NFxPEM algorithm is proposed where the coefficients of the predistorter are estimated using the Recursive Prediction Error Method (RPEM) algorithm [56,57]. The NFxPEM algorithm is expected to speed up the convergence, reduce the steady-state MSE and hence minimize the total nonlinear distortion at the output of the nonlinear physical system. The NFxPEM algorithm can also be derived for predistortion of Wiener and Hammerstein systems.

In order to implement the NFxLMS and NFxPEM algorithms, accurate system identification of the nonlinear physical system is needed. The effect of inaccurate estimation of the system transfer function on the FxLMS algorithm has been studied in [58, 59], the authors concluded that if the phase error in the estimate of the system transfer function does not exceed $\pm 90^\circ$, stable convergence of the FxLMS algorithm can be guaranteed. This is known as the Strictly Positive Real (SPR) condition [60]. However, the effect of the phase error between these bounds is very difficult to predict [59]. In order to relax the requirement for accurate system identification, the NFxLMS-ISE and NFxPEM-ISE algorithms are proposed for predistortion of Wiener systems, due to the effect that the ISE method is simple, fast and does not require high computational complexity, compared to the full system identification of Wiener systems.

Furthermore, a frequency domain predistortion technique is also introduced in this chapter. The basic idea is an extension from [61], where a frequency domain method is introduced for estimating the telephone handset nonlinearity by matching the spectral magnitude of the distorted signal to the output of a nonlinear model. The nonlinear model is chosen as a Wiener-Hammerstein cascade system with a static nonlinearity described by a finite-order polynomial. The coefficients of the nonlinear model are estimated using the generalized Newton iteration algorithm [49, 62] that minimizes a cost function of the sum squared error between the spectral magnitudes - evaluated over a number of short-time time frames - of the measured distorted signal and the output signal of the nonlinear model. In this chapter, this nonlinear system identification [61], namely the Spectral Magnitude Matching (SMM) method, is extended to predistortion of nonlinear systems. The coefficients of the predistorter are estimated recursively using the generalized Newton iteration algorithm to minimize the sum squared error between the spectral magnitudes of the system output $z(n)$ in Fig. 2.1 and the new defined reference signal $r(n)$ in (2.3). The SMM approach is a general DLA approach for any kind of predistorter model and does not require the nonlinear physical system to be identified as required for the time domain approaches.

Two methods are used to measure the performance of the adaptation algorithms in this chapter. First, we define the normalized Mean Square Distortion (MSD) of the linearized system as

$$E_D = 10 \log_{10} \left(\frac{\widehat{\mathrm{E}}\{e^2(n)\}}{\widehat{\mathrm{E}}\{r^2(n)\}} \right) \tag{2.6}$$

where $\widehat{\mathrm{E}}\{.\}$ is the mean obtained by a number of independent experiments. From E_D we can observe whether the error signal $e(n)$ is well reduced - hence $z(n)$ is well approximating $r(n)$. Second, the mean Power Spectral Density (PSD) of the output signal $z(n)$ over a number of independent experiments is considered to check whether the spectral regrowth is effectively reduced.

2.2. Predistortion of Volterra Systems

In this section, the adaptation algorithms for predistortion of Volterra systems using the DLA approach are introduced. The NFxLMS and NFxRLS algorithms are first reviewed, and then the NFxPEM algorithm is derived. Simulation results for comparison of these algorithms are given at the end of this section.

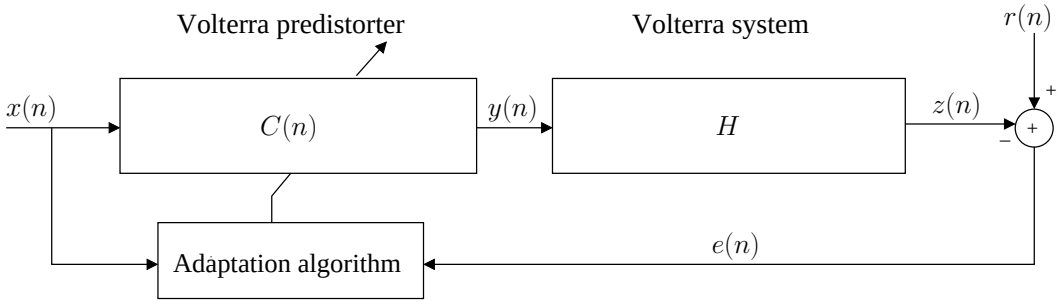

Figure 2.2 The DLA approach for predistortion of Volterra systems.

The DLA approach for predistortion of Volterra systems is shown in Fig. 2.2. The nonlinear physical system H is a qth-order Volterra series with input and output signals $y(n)$ and $z(n)$, respectively. The output signal $z(n)$ is given by

$$\begin{aligned} z(n) &= \boldsymbol{h}^T \boldsymbol{y}(n) = \sum_{k=1}^{q} \boldsymbol{h}_k^T \boldsymbol{y}_k(n) \\ &= \sum_{k=1}^{q} \left(\sum_{i_1=0}^{M_k-1} \cdots \sum_{i_k=0}^{M_k-1} h_k(i_1, \cdots, i_k) y(n-i_1) \cdots y(n-i_k) \right) \end{aligned} \tag{2.7}$$

where M_k is the memory length of the kth order kernel. The kernel vector $\boldsymbol{h}$ is defined as

$$\boldsymbol{h} = \begin{pmatrix} \boldsymbol{h}_1^T & \cdots & \boldsymbol{h}_q^T \end{pmatrix}^T, \tag{2.8}$$

$$\boldsymbol{h}_k = \begin{pmatrix} \boldsymbol{h}_k(0, \cdots, 0) \\ \vdots \\ \boldsymbol{h}_k(i_1, \cdots, i_k) \\ \vdots \\ \boldsymbol{h}_k(M_k-1, \cdots, M_k-1) \end{pmatrix}, \quad k = 1, \cdots, q, \tag{2.9}$$

and the input vector $\boldsymbol{y}(n)$ is defined as

$$\boldsymbol{y}(n) = \begin{pmatrix} \boldsymbol{y}_1^T(n) & \cdots & \boldsymbol{y}_q^T(n) \end{pmatrix}^T, \tag{2.10}$$

$$\boldsymbol{y}_k(n) = \begin{pmatrix} y^k(n) \\ \vdots \\ y(n-i_1) \cdots y(n-i_k) \\ \vdots \\ y^k(n-M_k+1) \end{pmatrix}, \quad k = 1, \cdots, q. \tag{2.11}$$

Similarly, the relation between the input and output of the adaptive Volterra predistorter $C(n)$ is given by

$$\begin{aligned} y(n) &= \boldsymbol{c}^T(n)\boldsymbol{x}(n) = \sum_{k=1}^{p} \boldsymbol{c}_k^T(n)\boldsymbol{x}_k(n) \\ &= \sum_{k=1}^{p} \left(\sum_{i_1=0}^{\widehat{M}_k-1} \cdots \sum_{i_k=0}^{\widehat{M}_k-1} c_k(i_1,\cdots,i_k;n)x(n-i_1)\cdots x(n-i_k) \right) \end{aligned} \tag{2.12}$$

where $\widehat{M}_k$ is the memory length. The kernel vector $\boldsymbol{c}(n)$ is defined as

$$\boldsymbol{c}(n) = \begin{pmatrix} \boldsymbol{c}_1^T(n) & \cdots & \boldsymbol{c}_p^T(n) \end{pmatrix}^T, \tag{2.13}$$

$$\boldsymbol{c}_k(n) = \begin{pmatrix} \boldsymbol{c}_k(0,\cdots,0;n) \\ \vdots \\ \boldsymbol{c}_k(i_1,\cdots,i_k;n) \\ \vdots \\ \boldsymbol{c}_k(\widehat{M}_k-1,\cdots,\widehat{M}_k-1;n) \end{pmatrix}, \quad k=1,\cdots,p, \tag{2.14}$$

and the input vector $\boldsymbol{x}(n)$ is defined as

$$\boldsymbol{x}(n) = \begin{pmatrix} \boldsymbol{x}_1^T(n) & \cdots & \boldsymbol{x}_p^T(n) \end{pmatrix}^T, \tag{2.15}$$

$$\boldsymbol{x}_k(n) = \begin{pmatrix} x^k(n) \\ \vdots \\ x(n-i_1)\cdots x(n-i_k) \\ \vdots \\ x^k(n-\widehat{M}_k+1) \end{pmatrix}, \quad k=1,\cdots,p. \tag{2.16}$$

Note that kernel vector $\boldsymbol{c}(n)$ of the adaptive Volterra predistorter shows an explicit dependence on the time index n, where the kernel vector $\boldsymbol{h}$ of the nonlinear physical system is considered time-invariant.

According to the pth-order inverse theory [13], the Volterra predistorter $C(n)$ can remove nonlinearities up to pth-order provided that the inverse of the first-order kernel of the Volterra system H is causal and stable. The kernels of $C(n)$ can be estimated using different adaptation algorithms, which are the topics of the next sections.

2.2.1. The NFxLMS algorithm

The NFxLMS algorithm in [23] is obtained by applying the stochastic gradient algorithm, see [56, 63], as

$$\boldsymbol{c}(n+1) = \boldsymbol{c}(n) - \frac{\mu}{2}\boldsymbol{\Delta}^T(n) \tag{2.17}$$

where μ, usually defined as the *step-size* parameter, is a small positive constant that controls stability and rate of convergence of the adaptation algorithm. The gradient vector $\boldsymbol{\Delta}(n)$ is defined as

$$\boldsymbol{\Delta}(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)} e^2(n). \tag{2.18}$$

Taking into consideration of (2.5), we have

$$\nabla_{\boldsymbol{c}(n)} e^2(n) = -2e(n)\nabla_{\boldsymbol{c}(n)} z(n) \tag{2.19}$$

where $\nabla_{\boldsymbol{c}(n)} z(n)$ can be written as (*cf.* (2.7))

$$\nabla_{\boldsymbol{c}(n)} z(n) = \sum_{r=0}^{M-1} \frac{\partial z(n)}{\partial y(n-r)} \nabla_{\boldsymbol{c}(n)} y(n-r) = \sum_{r=0}^{M-1} g(r;n) \nabla_{\boldsymbol{c}(n)} y(n-r). \tag{2.20}$$

Here $M = \max\{M_1, \cdots, M_q\}$ and $g(r;n)$ is given by

$$\begin{aligned} g(r;n) = \frac{\partial z(n)}{\partial y(n-r)} &= h_1(r) + 2\sum_{i=0}^{M_2-1} h_2(r,i) y(n-i) + \\ & \quad 3\sum_{i_1=0}^{M_3-1}\sum_{i_2=0}^{M_3-1} h_3(r,i_1,i_2) y(n-i_1) y(n-i_2) + \cdots \end{aligned} \tag{2.21}$$

where $h_k(r, i_1, \cdots, i_k), k = 1, \cdots, q$ is equal to 0 when $M_k < r \leq M$. Assuming that μ is chosen sufficiently small so that the kernel vector $\boldsymbol{c}(n)$ is changing slowly [23,26], $\nabla_{\boldsymbol{c}(n)} y(n-r)$ can be approximated as (*cf.* (2.12))

$$\nabla_{\boldsymbol{c}(n)} y(n-r) \approx \nabla_{\boldsymbol{c}(n-r)} y(n-r) = \boldsymbol{x}^T(n-r). \tag{2.22}$$

Substituting (2.20)-(2.22) in (2.19), we have

$$\boldsymbol{\Delta}(n) = \nabla_{\boldsymbol{c}(n)} e^2(n) = -2e(n) \sum_{r=0}^{M-1} g(r;n) \boldsymbol{x}^T(n-r). \tag{2.23}$$

Remark 2.2: In (2.21), it is assumed that the correct kernels of the nonlinear physical system H are known or have been estimated. The problem of estimating Volterra kernels for nonlinear systems is discussed, *e.g.*, in [33].

2.2.2. The NFxRLS algorithm

The NFxRLS algorithm in [26] is derived by minimizing the cost function

$$\xi(n) = \sum_{i=1}^{n} \lambda^{n-i} e^2(i) = \sum_{i=1}^{n} \lambda^{n-i} (r(i) - z(i))^2 \tag{2.24}$$

where $0 < \lambda < 1$ is an exponential forgetting factor and $r(i)$ is defined similarly as in (2.3).

To derive the NFxRLS algorithm we need to solve $\nabla_{\boldsymbol{c}(n)} \xi(n)$, so differentiating both sides of (2.24) w.r.t. $\boldsymbol{c}(n)$ we have

$$\nabla_{\boldsymbol{c}(n)} \xi(n) = -2 \sum_{i=1}^{n} \lambda^{n-i} (r(i) - z(i)) \nabla_{\boldsymbol{c}(n)} z(i). \tag{2.25}$$

Straightforward analysis identical to Section 2.2.1 gives

$$\nabla_{\boldsymbol{c}(n)} z(i) = \sum_{r=0}^{M-1} g(r;i) \boldsymbol{x}^T(i-r) = \boldsymbol{\varphi}^T(i) \tag{2.26}$$

where $M = \max\{M_1, \cdots, M_q\}$ and $g(r;i)$ is given by (2.21). Also, the output of H can be approximated as

$$\begin{aligned} z(i) &\approx \sum_{r=0}^{M-1} g(r;i)y(i-r) = \sum_{r=0}^{M-1} g(r;i)[\boldsymbol{c}^T(n)\boldsymbol{x}(i-r)] \\ &= \boldsymbol{c}^T(n)\sum_{r=0}^{M-1} g(r;i)\boldsymbol{x}(i-r) = \boldsymbol{c}^T(n)\boldsymbol{\varphi}(i). \end{aligned} \tag{2.27}$$

Note that $y(i) = \boldsymbol{c}(n)\boldsymbol{x}(i)$ for all $i = 1, ..., n$ because in RLS algorithm, at time n, there is a unique optimal $\boldsymbol{c}(n)$ held constant over entire optimal window running from $i = 1$ to $i = n$.

Substituting (2.26) and (2.27) in (2.25) gives

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{c}(n)}\xi(n) &\approx -2\sum_{i=1}^{n} \lambda^{n-i}(r(i) - \boldsymbol{c}^T(n)\boldsymbol{\varphi}(i))\boldsymbol{\varphi}^T(i) \\ &= -2\sum_{i=1}^{n} \lambda^{n-i} r(i)\boldsymbol{\varphi}^T(i) + 2\sum_{i=1}^{n} \lambda^{n-i}\boldsymbol{c}^T(n)\boldsymbol{\varphi}(i)\boldsymbol{\varphi}^T(i). \end{aligned} \tag{2.28}$$

Setting the right hand side of (2.28) equal to zero vector yields

$$\boldsymbol{R}_{\varphi\varphi}(n)\boldsymbol{c}(n) = \boldsymbol{r}_{r\varphi}(n) \tag{2.29}$$

where

$$\boldsymbol{R}_{\varphi\varphi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\boldsymbol{\varphi}(i)\boldsymbol{\varphi}^T(i) \tag{2.30}$$

$$\boldsymbol{r}_{r\varphi}(n) = \sum_{i=1}^{n} \lambda^{n-i} r(i)\boldsymbol{\varphi}(i). \tag{2.31}$$

Therefore, the NFxRLS algorithm follows as (*cf.* [57, 63, 64])

$$\begin{aligned} &e(n) = r(n) - z(n) \\ &\boldsymbol{k}(n) = \big(\lambda + \boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\varphi}(n)\big)^{-1}\boldsymbol{P}(n-1)\boldsymbol{\varphi}(n) \\ &\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\big)/\lambda \\ &\boldsymbol{c}(n+1) = \boldsymbol{c}(n) + \boldsymbol{k}(n)e(n). \end{aligned} \tag{2.32}$$

The most common choice for the initial condition of $\boldsymbol{P}(n)$ is $\boldsymbol{P}(0) = \rho\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and ρ is a constant which reflects our trust in the initial kernel vector $\boldsymbol{c}(0)$.

2.2.3. The NFxPEM algorithm

The NFxPEM algorithm is derived by the minimization of the cost function [57]

$$V(\boldsymbol{c}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n)\right] \tag{2.33}$$

where $e(n)$ is the prediction error defined as

$$e(n) = r(n) - z(n). \tag{2.34}$$

The formulation of the NFxPEM algorithm requires the negative gradient of $e(n)$ w.r.t. $\boldsymbol{c}(n)$ which is defined as

$$\boldsymbol{\varphi}^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{c}(n)} e(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)} z(n). \tag{2.35}$$

Straightforward analysis identical to Section 2.2.1 gives

$$\boldsymbol{\varphi}^T(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)} z(n) = \sum_{r=0}^{M-1} g(r;n)\boldsymbol{x}^T(n-r) \tag{2.36}$$

where $M = \max\{M_1, \cdots, M_q\}$ and $g(r;n)$ is given in (2.21). Hence, the NFxPEM algorithm follows as (*cf.* [57, 63])

$$\begin{aligned}
&e(n) = r(n) - z(n)\\
&\lambda(n) = \lambda_0\lambda(n-1) + 1 - \lambda_0\\
&s(n) = \boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\varphi}(n) + \lambda(n)\\
&\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1) - \boldsymbol{P}(n-1)\boldsymbol{\varphi}(n)s^{-1}(n)\boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\big)/\lambda(n)\\
&\boldsymbol{c}(n+1) = \boldsymbol{c}(n) + \boldsymbol{P}(n)\boldsymbol{\varphi}(n)e(n).
\end{aligned} \tag{2.37}$$

Here $\lambda(n)$ is a forgetting factor that grows exponentially to 1 as $n \to \infty$ where the rate λ_0 and the initial value $\lambda(0)$ are design variables. The numerical values $\lambda_0 = 0.99$ and $\lambda(0) = 0.95$ have proven to be useful in many applications [57]. Also, $\boldsymbol{P}(n) = n\boldsymbol{R}^{-1}(n)$ where $\boldsymbol{R}(n)$ is the Hessian approximation in the Gauss-Newton algorithm, see [57, 63]. The most common choice for the initial condition of $\boldsymbol{P}(n)$ is $\boldsymbol{P}(0) = \rho\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and ρ is a constant that reflects our trust in the initial kernel vector $\boldsymbol{c}(0)$. In case of no prior knowledge, $\boldsymbol{c}(0) = \boldsymbol{0}$ and ρ is large to speed up convergence to the true parameter vector.

Remark 2.3: Since the gradient vectors $\boldsymbol{\varphi}(n)$ in (2.26) and (2.36) are equivalent, the NFxRLS algorithms in (2.32) can be obtained exactly from the NFxPEM algorithm in (2.37) by setting $\lambda_0 = 1$ and $\lambda(0) = \lambda$.

2.2.4. Simulation study

In this section, a comparative simulation study of the NFxLMS, NFxRLS and NFxPEM algorithms is given.

The nonlinear physical system H was chosen as a known 2nd-order time-invariant Volterra system with memory length $M_1 = 4$ and $M_2 = 3$. The adaptive predistorter $C(n)$ was also assumed to be a 2nd-order Volterra filter. This means that $q = p = 2$. Also, the number of memory in the adaptive Volterra predistorter was chosen as $\widehat{M}_1 = \widehat{M}_2 = 4$. The input-output relation of H was chosen to be

$$z(n) = \boldsymbol{h}_1^T\boldsymbol{y}_1(n) + \boldsymbol{h}_2^T\boldsymbol{y}_2(n) \tag{2.38}$$

where the first-order kernel vector $\boldsymbol{h}_1$ was

$$\boldsymbol{h}_1 = \begin{pmatrix} 0.5625 & 0.4810 & 0.1124 & -0.1669 \end{pmatrix} \tag{2.39}$$

and the second-order kernel vector $\boldsymbol{h}_2$ was

$$\boldsymbol{h}_2 = (0.0175 \quad 0 \quad 0 \quad 0 \quad 0 \quad -0.0088 \quad 0 \quad -0.0088 \quad 0)\,. \tag{2.40}$$

Here $\boldsymbol{h}_2$ was obtained by vectorizing the second-order kernel matrix

$$\boldsymbol{H}_2 = \begin{pmatrix} 0.0175 & 0 & 0 \\ 0 & 0 & -0.0088 \\ 0 & -0.0088 & 0 \end{pmatrix}. \tag{2.41}$$

The number of independent experiments was 100. In each experiment, the input signal $x(n)$ to the predistorter was chosen as a random signal with uniform distribution over $(-1, 1)$ with data length 2×10^4 and the frequency band was limited by a low-pass filter in order to prevent aliasing at the output of H [13]. The normalized cut-off frequency of this filter is chosen as $\frac{\pi}{4}$, because after the two cascaded 2nd-order nonlinear systems $C(n)$ and H, the bandwidth of $z(n)$ will be 4 times as wide as the bandwidth of $x(n)$. The reference signal $r(n)$ was chosen to be equal to the input signal $x(n)$ without delay since the linear subsystem of H is minimum phase, plus the measurement noise which was AWGN such that a signal to noise ratio (SNR) of 40 dB was achieved.

The MSD comparison between the NFxLMS, NFxRLS and NFxPEM algorithms is given in Fig. 2.3. The step size of the NFxLMS algorithm was $\mu = 0.1$ and the matrix $\mathbf{P}(0) = 100\mathbf{I}$ for the NFxRLS and NFxPEM algorithms. λ, λ_0 and $\lambda(0)$ were chosen as 1, 0.99 and 0.95, respectively. The MSD of the nonlinear physical system without predistorter was about -16 dB. The NFxRLS and NFxPEM algorithms achieve much lower distortion values than the NFxLMS algorithm. On average, the NFxLMS algorithm achieves about -23 dB after 2×10^4 samples but it still does not converge. The NFxRLS algorithm converges after about 4000 samples and achieves about -40 dB. The NFxPEM algorithm converges after about 800 samples and achieves about -40 dB. Obviously, the NFxRLS and NFxPEM algorithms converge much faster than the NFxLMS algorithm.

Figure 2.4 shows the mean PSDs of the output signals of the nonlinear physical system without and with the predistorter after 2×10^4 samples. From this figure, we can see that in the normalized frequency band $(0.30\pi, 0.55\pi)$, the NFxLMS algorithm can only reduce the spectral regrowth by up to 5 dB. As compared to the NFxLMS algorithm, the NFxRLS and NFxPEM algorithms can reduce the spectral regrowth more effectively (up to 30 dB). For all algorithms, there is spectral regrowth in the normalized frequency band $(0.55\pi, 0.75\pi)$, which is caused by the cascaded nonlinear systems $C(n)$ and H. However, this spectral regrowth has no significant impact on the performance of predistortion since it is relatively small (under -80 dB).

2.3. Predistortion of Wiener Systems

The adaptation algorithms for predistortion of Wiener systems using the DLA approach are introduced in this section. The Wiener model structure consists of a linear dynamic system followed by a static nonlinearity, and the linear dynamic system can be modeled as a Finite Impulse Response (FIR) filter or an Infinite Impulse Response (IIR) filter. In this section the predistortion of IIR Wiener systems, where the linear dynamic system is modeled as an IIR filter, is considered - hence the predistortion of FIR Wiener systems can

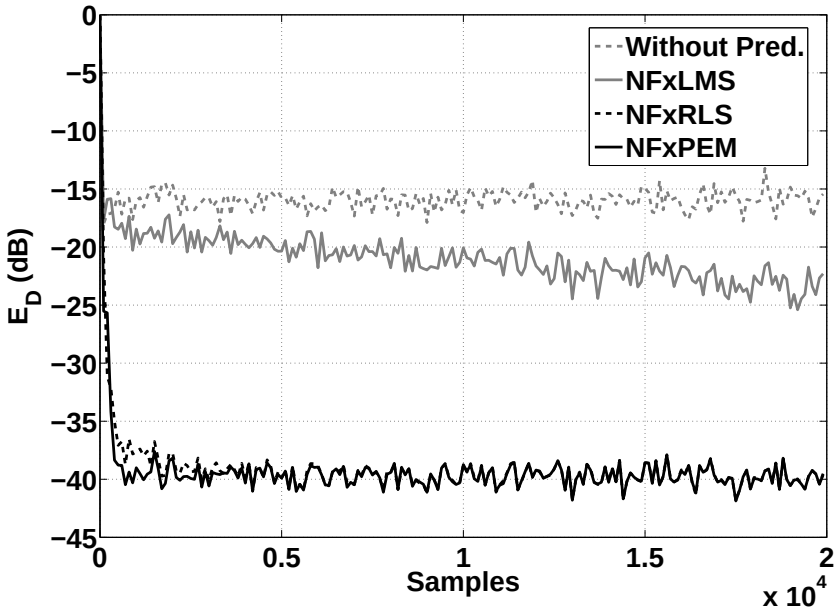

Figure 2.3 MSD E_D for different adaptation algorithms.

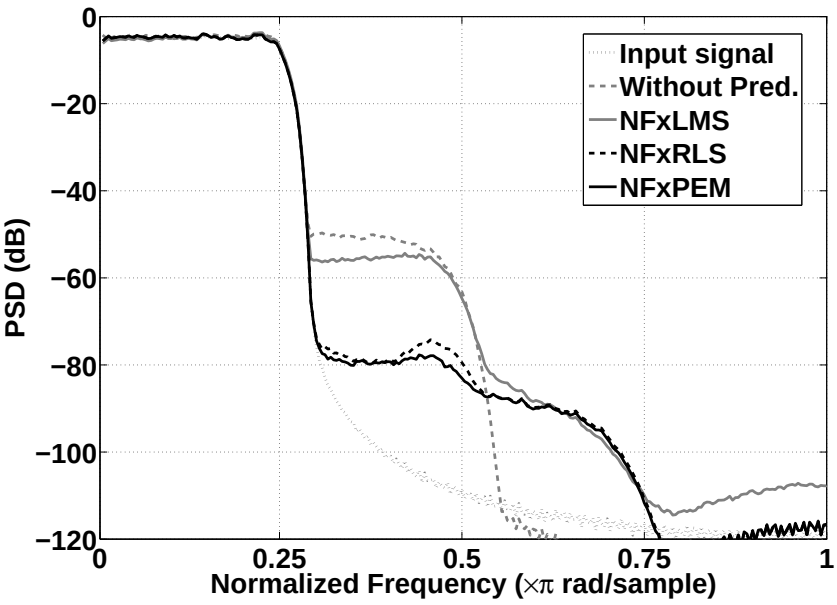

Figure 2.4 Mean PSDs for different adaptation algorithms.

be regarded as a special case. The NFxLMS algorithm is first derived and the NFxLMS-ISE algorithm is proposed, based on using an initial estimate for the linear and nonlinear subsystems as described in [65]. Then, the NFxPEM is derived and by using the initial subsystem estimates, the NFxPEM-ISE algorithms is also proposed. The validity of these algorithms is demonstrated via computer simulation.

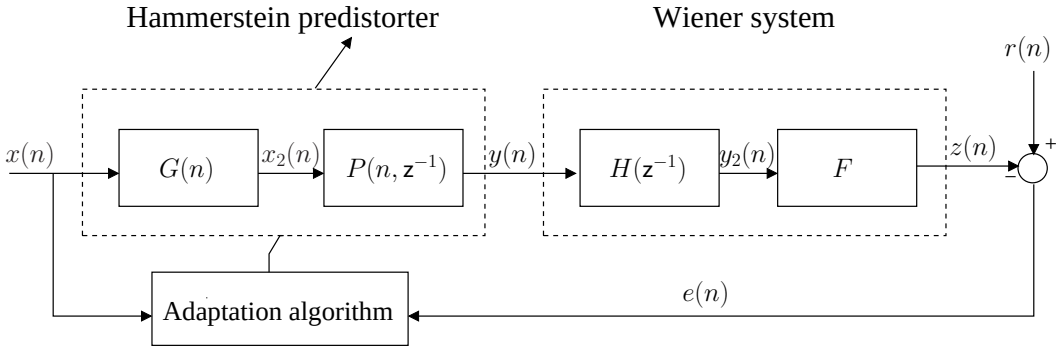

Figure 2.5 The DLA approach for predistortion of Wiener systems.

The DLA approach for predistortion of Wiener systems is shown in Fig. 2.5. The output of the IIR Wiener system is

$$\begin{aligned} z(n) &= f_1 y_2(n) + f_2 y_2^2(n) + \cdots + f_{m_f} y_2^{m_f}(n) \\ &= \boldsymbol{f}^T \boldsymbol{y}_2(n) \end{aligned} \tag{2.42}$$

where $\boldsymbol{f}$ is the parameter vector of the nonlinear subsystem F defined as

$$\boldsymbol{f} = \begin{pmatrix} f_1 & f_2 & \cdots & f_{m_f} \end{pmatrix}^T \tag{2.43}$$

and the corresponding input vector $\boldsymbol{y}_2(n)$ is given by

$$\boldsymbol{y}_2(n) = \begin{pmatrix} y_2(n) & y_2^2(n) & \cdots & y_2^{m_f}(n) \end{pmatrix}^T. \tag{2.44}$$

The intermediate signal $y_2(n)$ is defined as

$$\begin{aligned} y_2(n) &= H(\mathsf{z}^{-1}) y(n) = \frac{B(\mathsf{z}^{-1})}{1 - A(\mathsf{z}^{-1})} y(n) \\ &= \sum_{m=0}^{m_b} b_m y(n-m) + \sum_{m=1}^{m_a} a_m y_2(n-m) \end{aligned} \tag{2.45}$$

where $H(\mathsf{z}^{-1}) = \frac{B(\mathsf{z}^{-1})}{1-A(\mathsf{z}^{-1})}$ is an IIR filter and the polynomials $A(\mathsf{z}^{-1})$ and $B(\mathsf{z}^{-1})$ are defined as

$$\begin{aligned} A(\mathsf{z}^{-1}) &= \sum_{m=1}^{m_a} a_m \mathsf{z}^{-m} \\ B(\mathsf{z}^{-1}) &= \sum_{m=0}^{m_b} b_m \mathsf{z}^{-m}. \end{aligned} \tag{2.46}$$

Here z^{-1} is the delay operator such that $\mathsf{z}^{-m}x(n) = x(n-m)$.

The predistorter of the IIR Wiener system is chosen as an IIR Hammerstein system and the reason to choose Hammerstein predistorter is given in Section 4.2.1. The output of the predistorter is

$$\begin{aligned} y(n) &= P(n,\mathsf{z}^{-1})x_2(n) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}x_2(n) \\ &= \sum_{m=0}^{m_d} d_m(n)x_2(n-m) + \sum_{m=1}^{m_c} c_m(n)y(n-m) \end{aligned} \tag{2.47}$$

where $P(n,\mathsf{z}^{-1}) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}$ is an IIR filter and the polynomials $C(n,\mathsf{z}^{-1})$ and $D(n,\mathsf{z}^{-1})$ are defined as

$$\begin{aligned} C(n,\mathsf{z}^{-1}) &= \sum_{m=1}^{m_c} c_m(n)\mathsf{z}^{-m} \\ D(n,\mathsf{z}^{-1}) &= \sum_{m=0}^{m_d} d_m(n)\mathsf{z}^{-m}. \end{aligned} \tag{2.48}$$

The intermediate signal $x_2(n)$ is defined as

$$\begin{aligned} x_2(n) &= g_1(n)x(n) + g_2(n)x^2(n) + \cdots + g_{m_g}(n)x^{m_g}(n) \\ &= \boldsymbol{g}^T(n)\boldsymbol{x}(n) \end{aligned} \tag{2.49}$$

where $\boldsymbol{g}(n)$ is the parameter vector of the nonlinear subsystem $G(n)$ defined as

$$\boldsymbol{g}(n) = \begin{pmatrix} g_1(n) & g_2(n) & \cdots & g_{m_g}(n) \end{pmatrix}^T \tag{2.50}$$

and the corresponding input vector $\boldsymbol{x}(n)$ is given by

$$\boldsymbol{x}(n) = \begin{pmatrix} x(n) & x^2(n) & \cdots & x^{m_g}(n) \end{pmatrix}^T. \tag{2.51}$$

Here we define the parameter vector $\boldsymbol{\theta}$ of the predistorter as

$$\begin{aligned} \boldsymbol{\theta} &= \begin{pmatrix} \boldsymbol{\theta}_d^T & \boldsymbol{\theta}_c^T & \boldsymbol{\theta}_g^T \end{pmatrix}^T \\ \boldsymbol{\theta}_d &= \begin{pmatrix} d_0 & d_1 & \cdots & d_{m_d} \end{pmatrix}^T \\ \boldsymbol{\theta}_c &= \begin{pmatrix} c_1 & c_2 & \cdots & c_{m_c} \end{pmatrix}^T \\ \boldsymbol{\theta}_g &= \begin{pmatrix} g_1 & g_2 & \cdots & g_{m_g} \end{pmatrix}^T. \end{aligned} \tag{2.52}$$

Different adaptation algorithms for estimating the parameter vector $\boldsymbol{\theta}$ are introduced in the following sections.

2.3.1. The NFxLMS algorithm

The NFxLMS algorithm is obtained by applying the stochastic gradient algorithm [56, 63]:

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) - \frac{\mu}{2}\boldsymbol{\Delta}^T(n) \tag{2.53}$$

where μ is the *step-size* parameter. The gradient vector $\boldsymbol{\Delta}(n)$ is defined as

$$\boldsymbol{\Delta}(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} e^2(n) = -2e(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n). \tag{2.54}$$

Using (2.42)-(2.44), $\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n)$ can be derived as

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n) &= \boldsymbol{f}^T \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} \boldsymbol{y}_2(n) = \boldsymbol{f}^T \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n) \\ \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2^2(n) \\ \vdots \\ \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2^{m_f}(n) \end{pmatrix} \\ &= \boldsymbol{f}^T \begin{pmatrix} 1 \\ 2y_2(n) \\ \vdots \\ m_f y_2^{m_f-1}(n) \end{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n). \end{aligned} \tag{2.55}$$

Since it is not possible to measure the intermediate signal $y_2(n)$, using (2.45) and defining

$$s_1(n) = \boldsymbol{f}^T \begin{pmatrix} 1 \\ 2y_2(n) \\ \vdots \\ m_f y_2^{m_f-1}(n) \end{pmatrix} = \boldsymbol{f}^T \begin{pmatrix} 1 \\ 2\big[H(\mathsf{z}^{-1})y(n)\big] \\ \vdots \\ m_f \big[H(\mathsf{z}^{-1})y(n)\big]^{m_f-1} \end{pmatrix} \tag{2.56}$$

we have

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n) = s_1(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n) \tag{2.57}$$

where

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n) = \sum_{m=0}^{m_b} b_m \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y(n-m) + \sum_{m=1}^{m_a} a_m \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n-m). \tag{2.58}$$

Assuming that $\boldsymbol{\theta}(n)$ changes slowly [23, 26], we have

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y(n-m) &\approx \boldsymbol{\nabla}_{\boldsymbol{\theta}(n-m)} y(n-m), \quad m = 0, 1, \cdots, m_b \\ \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n-m) &\approx \boldsymbol{\nabla}_{\boldsymbol{\theta}(n-m)} y_2(n-m), \quad m = 1, \cdots, m_a. \end{aligned} \tag{2.59}$$

Thus (2.58) becomes

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n) &\approx \sum_{m=0}^{m_b} b_m \boldsymbol{\nabla}_{\boldsymbol{\theta}(n-m)} y(n-m) + \sum_{m=1}^{m_a} a_m \boldsymbol{\nabla}_{\boldsymbol{\theta}(n-m)} y_2(n-m) \\ &= \frac{B(\mathsf{z}^{-1})}{1 - A(\mathsf{z}^{-1})} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y(n). \end{aligned} \tag{2.60}$$

Again using (2.45), (2.60) can be written as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n) = H(\mathsf{z}^{-1}) \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y(n) = H(\mathsf{z}^{-1}) \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} y(n) \end{pmatrix}. \tag{2.61}$$

Differentiating both sides of (2.47) with respect to $d_k(n)$ and $c_k(n)$ gives

$$\begin{aligned}\frac{\partial y(n)}{\partial d_k(n)} &= x_2(n-k) + \sum_{m=1}^{m_c} c_m(n)\frac{\partial y(n-m)}{\partial d_k(n)} \\ \frac{\partial y(n)}{\partial c_k(n)} &= y(n-k) + \sum_{m=1}^{m_c} c_m(n)\frac{\partial y(n-m)}{\partial c_k(n)}.\end{aligned} \tag{2.62}$$

Since the parameter vector $\boldsymbol{\theta}(n)$ is assumed to be changing slowly, we can write

$$\begin{aligned}\frac{\partial y(n-m)}{\partial d_k(n)} &\approx \frac{\partial y(n-m)}{\partial d_k(n-m)}, \quad m = 1, 2, \cdots, m_c \\ \frac{\partial y(n-m)}{\partial c_k(n)} &\approx \frac{\partial y(n-m)}{\partial c_k(n-m)}, \quad m = 1, 2, \cdots, m_c.\end{aligned} \tag{2.63}$$

Hence, (2.62) can be rewritten as

$$\begin{aligned}\frac{\partial y(n)}{\partial d_k(n)} &\approx x_2(n-k) + \sum_{m=1}^{m_c} c_m(n)\frac{\partial y(n-m)}{\partial d_k(n-m)} \\ \frac{\partial y(n)}{\partial c_k(n)} &\approx y(n-k) + \sum_{m=1}^{m_c} c_m(n)\frac{\partial y(n-m)}{\partial c_k(n-m)}\end{aligned} \tag{2.64}$$

or

$$\begin{aligned}\frac{\partial y(n)}{\partial d_k(n)} &\approx \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} x_2(n) \\ &= \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} \left(\boldsymbol{\theta}_g^T(n)\boldsymbol{x}(n)\right), \quad k = 0, 1, \cdots, m_d \\ \frac{\partial y(n)}{\partial c_k(n)} &\approx \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} y(n), \quad k = 1, \cdots, m_c.\end{aligned} \tag{2.65}$$

Similarly, differentiating both sides of (2.47) with respect to $g_k(n)$ gives

$$\frac{\partial y(n)}{\partial g_k(n)} = \sum_{m=0}^{m_d} d_m(n)\frac{\partial x_2(n-m)}{\partial g_k(n)} + \sum_{m=1}^{m_c} c_m(n)\frac{\partial y(n-m)}{\partial g_k(n)}. \tag{2.66}$$

Again because the parameter vector $\boldsymbol{\theta}(n)$ is assumed to be changing slowly, we can write

$$\begin{aligned}\frac{\partial x_2(n-m)}{\partial g_k(n)} &\approx \frac{\partial x_2(n-m)}{\partial g_k(n-m)}, \quad m = 0, \cdots, m_d \\ \frac{\partial y(n-m)}{\partial g_k(n)} &\approx \frac{\partial y(n-m)}{\partial g_k(n-m)}, \quad m = 1, \cdots, m_c.\end{aligned} \tag{2.67}$$

Hence, (2.66) can be rewritten as

$$\begin{aligned}\frac{\partial y(n)}{\partial g_k(n)} &= \sum_{m=0}^{m_d} d_m(n)\frac{\partial x_2(n-m)}{\partial g_k(n-m)} + \sum_{m=1}^{m_c} c_m(n)\frac{\partial y(n-m)}{\partial g_k(n-m)} \\ &= \sum_{m=0}^{m_d} d_m(n)x^k(n-m) + \sum_{m=1}^{m_c} c_m(n)\frac{\partial y(n-m)}{\partial g_k(n-m)} \\ &= \frac{D(n, \mathsf{z}^{-1})}{1 - C(n, \mathsf{z}^{-1})} x^k(n) = P(n, \mathsf{z}^{-1})x^k(n), \quad k = 1, \cdots, m_g.\end{aligned} \tag{2.68}$$

In summary, the gradient vector $\boldsymbol{\Delta}(n)$ can be written as

$$\boldsymbol{\Delta}(n) = -2e(n)s_1(n)H(\mathsf{z}^{-1})\left(\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}y(n) \quad \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}y(n) \quad \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)}y(n)\right) \tag{2.69}$$

where $s_1(n)$ is calculated in (2.56), and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}y(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}y(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)}y(n)$ are given by (2.65) and (2.68), respectively. Predistortion of IIR Wiener system has been treated for the first time in our previous publication [38] and predistortion of FIR Wiener systems using the NFxLMS algorithm in [24,25] is a special case when $A(\mathsf{z}^{-1}) = 0$ in $H(\mathsf{z}^{-1})$ and $C(n, \mathsf{z}^{-1}) = 0$ in $P(n, \mathsf{z}^{-1})$.

2.3.2. The NFxLMS-ISE algorithm

As it is clear from (2.56) and (2.69), the IIR Wiener system should be known or identified before applying the NFxLMS algorithm since the linear subsystem $H(\mathsf{z}^{-1})$ and the nonlinear subsystem F are needed in order to calculate the gradient vector. Notice that in (2.69), $H(\mathsf{z}^{-1})$ performs as a filter for the gradient components. Hence, it could be replaced by any other filter with a similar frequency response. On the other hand, the accuracy of estimating the nonlinear subsystem F is not critical here since this will only contribute to the scalar factor $s_1(n)$ and, therefore, not change the direction of the gradient vector. In [65], the Initial Subsystem Estimates (ISE) method is proposed in order to approximately estimate the linear and nonlinear subsystems of the Wiener systems, and this approximate estimates can be used as the initial parameters for the system identification process. Therefore, the ISE method can be applied as an initial step and then we make use of the estimated subsystems in the NFxLMS algorithm. This will save the effort needed to identify accurate estimates for the linear and nonlinear subsystems.

The ISE method uses Discrete Multitone (DMT) signals to construct the excitation inputs of the system. There are two main advantages to use the DMT signals [65]: first, this signal is periodic which can average out the measurement noise. Second, the periodicity allows us to use a frequency domain representation without leakage problems. Also, as explained in [66], the frequency response function (FRF) of the best linear approximation of a nonlinear system can be better measured by averaging the system responses to the DMT signal than the system responses to the Gaussian signal, in terms of the variance and the bias of the measured FRF of the approximated nonlinear system.

The ISE method follows the following lines:

(1) Generate M different DMT singals $\boldsymbol{y}^{[m]}, 1 < m \leq M$.

(2) For each DMT signal $\boldsymbol{y}^{[m]}$, measure N periods of output signal $\boldsymbol{z}^{[m]}$ in steady state excitation, take the average over these periods to obtain the mean output signal $\bar{\boldsymbol{z}}^{[m]}$ and similarly evaluate the mean input signal $\bar{\boldsymbol{y}}^{[m]}$.

(3) Use the Fast Fourier Transform (FFT) to calculate the spectrum of $\bar{\boldsymbol{z}}^{[m]}$, $\bar{\boldsymbol{Z}}^{[m]}(k)$ and the spectrum of $\bar{\boldsymbol{y}}^{[m]}$, $\bar{\boldsymbol{Y}}^{[m]}(k)$. Then, estimate the frequency response $\widehat{\boldsymbol{H}}^{[m]}(k)$ for each excitation, *i.e.*, $\widehat{\boldsymbol{H}}^{[m]}(k) = \bar{\boldsymbol{Z}}^{[m]}(k)/\bar{\boldsymbol{Y}}^{[m]}(k)$.

(4) Take the average of all frequency response functions: $\widehat{\boldsymbol{H}}(k) = \frac{1}{M}\sum_{m=1}^{M}\widehat{\boldsymbol{H}}^{[m]}(k)$. From $\widehat{\boldsymbol{H}}(k)$, estimate a parametric linear model $\widehat{H}(\mathsf{z}^{-1})$.

(5) For each $\boldsymbol{y}^{[m]}$, compute the intermediate signal $\boldsymbol{y}_2^{[m]}$ using $\widehat{H}(\mathsf{z}^{-1})$. From the signals $\boldsymbol{y}_2^{[m]}$ and $\boldsymbol{z}^{[m]}$, the nonlinear subsystem F can be estimated using different techniques, *e.g.*, Least Squares (LS) fitting [63].

Therefore, the filter $H(\mathsf{z}^{-1})$ can be replaced in (2.69) by the initial linear subsystem estimate $\widehat{H}(\mathsf{z}^{-1})$ and the scalar factor $s_1(n)$ is replaced with

$$\widehat{s}_1(n) = \widehat{\boldsymbol{f}}^T \begin{pmatrix} 1 \\ 2[\widehat{H}(\mathsf{z}^{-1})y(n)] \\ \vdots \\ \widehat{m}_f[\widehat{H}(\mathsf{z}^{-1})y(n)]^{\widehat{m}_f-1} \end{pmatrix} \tag{2.70}$$

where $\widehat{\boldsymbol{f}}$ represents the initial nonlinear subsystem estimate of order $\widehat{m}_f$.

Now, the coefficient adaptation of the NFxLMS-ISE algorithm can be described as

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) - \frac{\mu}{2}\boldsymbol{\Delta}^T(n) \tag{2.71}$$

with

$$\boldsymbol{\Delta}(n) = -2e(n)\widehat{s}_1(n)\widehat{H}(\mathsf{z}^{-1}) \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)}y(n) \end{pmatrix} \tag{2.72}$$

where $\widehat{H}(\mathsf{z}^{-1})$ is the initial linear subsystem estimate, $\widehat{s}_1(n)$ is calculated using the initial nonlinear subsystem estimate $\widehat{\boldsymbol{f}}$ as in (2.70), and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}y(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}y(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)}y(n)$ are given by (2.65) and (2.68), respectively.

2.3.3. The NFxPEM and NFxPEM-ISE algorithms

The NFxPEM algorithm is derived by the minimization of the cost function [57]

$$V(\boldsymbol{\theta}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n)\right] \tag{2.73}$$

where $e(n)$ is the prediction error defined as

$$e(n) = r(n) - z(n). \tag{2.74}$$

The formulation of the NFxPEM algorithm requires the negative gradient of $e(n)$ w.r.t. $\boldsymbol{\theta}(n)$ which is defined as

$$\boldsymbol{\varphi}^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}e(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}z(n). \tag{2.75}$$

Straightforward analysis identical to Section 2.3.1 gives

$$\boldsymbol{\varphi}^T(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}z(n) = s_1(n)H(\mathsf{z}^{-1}) \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)}y(n) \end{pmatrix} \tag{2.76}$$

where $s_1(n)$ is calculated in (2.56), and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}y(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}y(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)}y(n)$ are given by (2.65) and (2.68), respectively. Hence, the NFxPEM algorithm follows as (*cf.* [57, 63])

$$\begin{aligned} &e(n) = r(n) - z(n) \\ &\lambda(n) = \lambda_0\lambda(n-1) + 1 - \lambda_0 \\ &s(n) = \boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\varphi}(n) + \lambda(n) \\ &\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1) - \boldsymbol{P}(n-1)\boldsymbol{\varphi}(n)s^{-1}(n)\boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\big)/\lambda(n) \\ &\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \boldsymbol{P}(n)\boldsymbol{\varphi}(n)e(n). \end{aligned} \tag{2.77}$$

Here, λ_0, $\lambda(0)$ and $\boldsymbol{P}(0)$ are design variables as introduced in Section 2.2.3.

From (2.76), it is clear that the IIR Wiener system should also be known or identified before applying the NFxPEM algorithm since the parameters of the linear subsystem $H(\mathsf{z}^{-1})$ and the nonlinear subsystem F are needed in order to calculate the negative gradient vector $\boldsymbol{\varphi}(n)$. Therefore, the ISE method can also be used to approximately estimate the linear and nonlinear subsystems of the Wiener system as an initial step and then be used for the NFxPEM algorithm. The ISE method follows the same procedure as in Section 2.3.2 and the negative gradient of $e(n)$ w.r.t. $\boldsymbol{\theta}(n)$ in the NFxPEM-ISE algorithm can be described as

$$\begin{aligned}\widehat{\boldsymbol{\varphi}}^T(n) &= \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n) = s_1(n) H(\mathsf{z}^{-1}) \left(\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n) \quad \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n) \quad \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} y(n)\right) \\ &\approx \widehat{s}_1(n) \widehat{H}(\mathsf{z}^{-1}) \left(\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n) \quad \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n) \quad \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} y(n)\right)\end{aligned} \tag{2.78}$$

where $\widehat{H}(\mathsf{z}^{-1})$ is the initial linear subsystem estimate, $\widehat{s}_1(n)$ is calculated using the initial nonlinear subsystem estimate $\widehat{\boldsymbol{f}}$ as in (2.70), and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} y(n)$ are given by (2.65) and (2.68), respectively. Hence, the NFxPEM-ISE algorithm follows as

$$\begin{aligned}&e(n) = r(n) - z(n) \\ &\lambda(n) = \lambda_0 \lambda(n-1) + 1 - \lambda_0 \\ &s(n) = \widehat{\boldsymbol{\varphi}}^T(n) \boldsymbol{P}(n-1) \widehat{\boldsymbol{\varphi}}(n) + \lambda(n) \\ &\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1) - \boldsymbol{P}(n-1) \widehat{\boldsymbol{\varphi}}(n) s^{-1}(n) \widehat{\boldsymbol{\varphi}}^T(n) \boldsymbol{P}(n-1)\big)/\lambda(n) \\ &\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \boldsymbol{P}(n) \widehat{\boldsymbol{\varphi}}(n) e(n).\end{aligned} \tag{2.79}$$

2.3.4. Simulation study

The simulation study for all algorithms regarding predistortion of IIR Wiener systems is given in this section. The following IIR Wiener system was considered:

$$\begin{aligned}z(n) &= y_2(n) + 0.25 y_2^2(n) + 0.125 y_2^3(n) \\ y_2(n) &= \frac{0.72 + 1.51\mathsf{z}^{-1} + 1.04\mathsf{z}^{-2} + 0.26\mathsf{z}^{-3}}{1 + 1.46\mathsf{z}^{-1} + 0.89\mathsf{z}^{-2} + 0.18\mathsf{z}^{-3}} y(n).\end{aligned} \tag{2.80}$$

The order of the linear block of the IIR Hammerstein predistorter was chosen as $m_c = 3$ and $m_d = 3$, in order to avoid model error. The order of the nonlinear block was chosen as $m_g = 9$, since it can achieve the best performance based on large number of experiments for different values. The number of independent experiments was 100 and in each experiment, the input signal was chosen to be a random signal with uniform distribution over $(-1, 1)$ and data length of 10^5 samples. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{5}$. The output measurement noise was considered as AWGN with SNR=40 dB.

The parameter vectors $\boldsymbol{\theta}$ were initialized as

$$\begin{aligned}\boldsymbol{\theta}_d(0) &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T \\ \boldsymbol{\theta}_c(0) &= \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T \\ \boldsymbol{\theta}_g(0) &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}^T.\end{aligned} \tag{2.81}$$

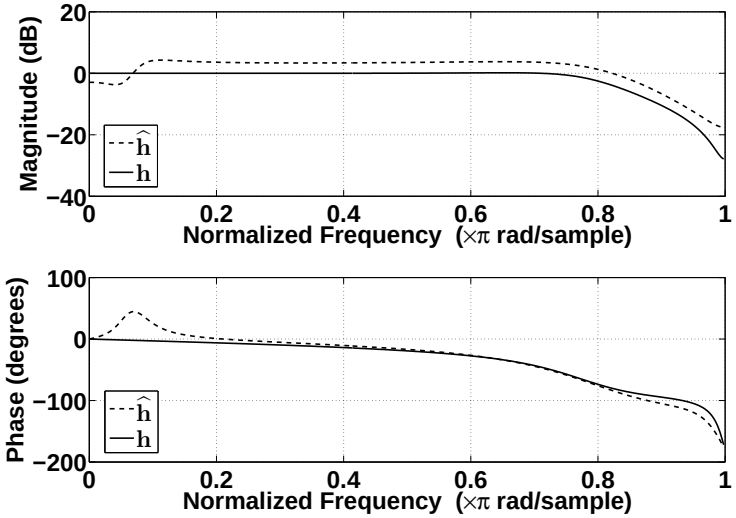

Figure 2.6 Frequency responses of the initial estimates of the linear subsystem $H(\mathsf{z}^{-1})$.

For the initial estimate of the IIR Wiener system, 10 different 64-tones DMT signals with crest factor (peak-to-average ratio) value of 3 and Root Mean Square (RMS) value of 0.25 were used as the system inputs. The constant crest factor of different DMT signals is achieved by using the time-frequency domain swapping algorithm proposed in [67]. The output measurement noise was considered as zero-mean AWGN with SNR=40 dB. $\widehat{H}(\mathsf{z}^{-1})$ was assumed to be an IIR filter with numerator order 5 and denominator order 5, and $\widehat{F}$ was assumed to be a 5th order static nonlinearity. The result of the ISE method $\widehat{H}(\mathsf{z}^{-1})$ is an estimation of the linear subsystem $H(\mathsf{z}^{-1})$ scaled with an unknown constant [65], and Fig. 2.6 gives the frequency responses of the initial estimates of $H(\mathsf{z}^{-1})$. The initial estimates of F were $(1.0001 \quad 0.2502 \quad 0.1246 \quad -0.0007 \quad 0.0005)^T$.

The MSD comparison of the NFxLMS, NFxLMS-ISE, NFxPEM and NFxPEM-ISE algorithms is given in Fig. 2.7. The MSD of the IIR Wiener system without predistorter was about -18 dB in a noise-free scenario. The step sizes of the NFxLMS and NFxLMS-ISE algorithms were $\mu = 0.03$ and the matrix $\mathbf{P}(0) = \mathbf{I}$ for the NFxPEM and NFxPEM-ISE algorithms. λ_0 and $\lambda(0)$ were chosen as 0.99 and 0.95, respectively. These algorithms achieve similar MSD performance in the steady state. On average, the NFxLMS-ISE and NFxLMS algorithms converge after about 8×10^4 samples and both of them achieve about -39 dB. The NFxPEM-ISE and NFxPEM algorithms converge after about 1000 samples and both of them achieve about -40 dB. Obviously, the NFxPEM and NFxPEM-ISE algorithms converge much faster than the NFxLMS and NFxLMS-ISE algorithms.

Figure 2.8 shows the mean PSDs of the output signals of the IIR Wiener system without and with predistorter after 10^5 samples. From this figure, we can see that all of these algorithms can effectively reduce the spectral regrowth in the normalized frequency band $(0.25\pi, 0.60\pi)$. Both the NFxLMS and NFxLMS-ISE algorithms can achieve up to 20 dB reduction. The NFxPEM algorithm has the best performance (up to 40 dB reduction), and the NFxPEM-ISE algorithm has the second best performance (up to 35 dB reduction) but

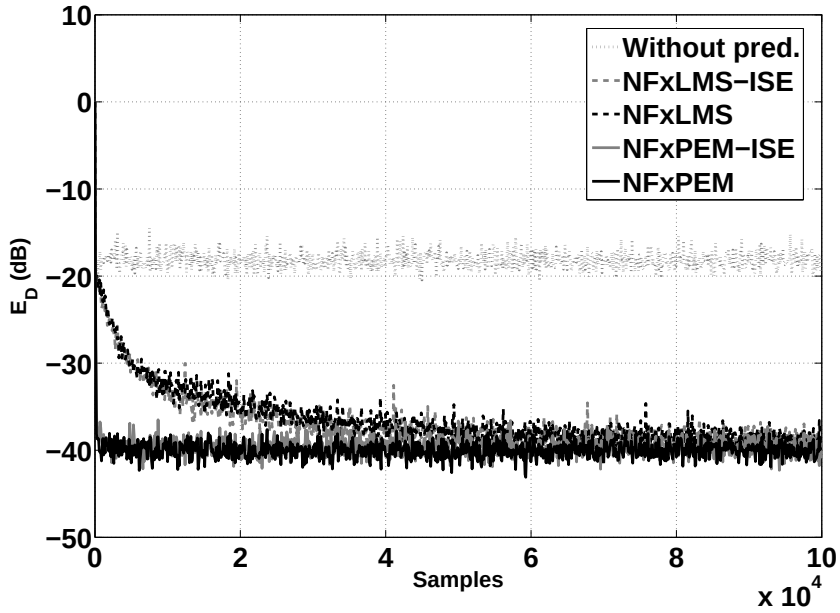

Figure 2.7 MSD E_D for different adaptation algorithms.

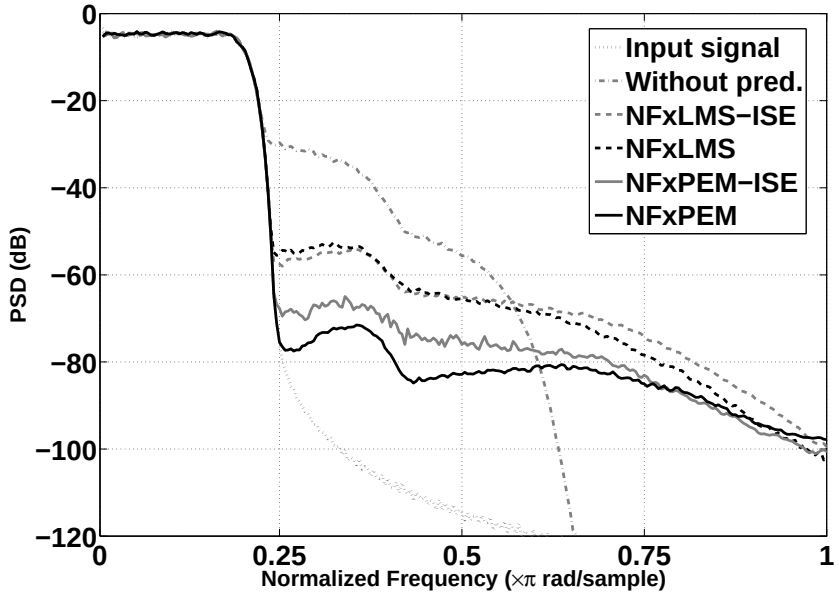

Figure 2.8 Mean PSDs for different adaptation algorithms.

without using the accurate system identification of the IIR Wiener system. For all algorithms, there is spectral regrowth in the normalized frequency band $(0.6\pi, \pi)$, which is caused by the cascaded nonlinear systems. However, this spectral regrowth has no significant impact on the performance of predistortion since it is relatively small (under -70 dB).

2.4. Predistortion of Hammerstein Systems

According to [68], power amplifiers can be modeled as FIR Wiener systems or IIR Hammerstein systems, where the linear dynamic system is modeled as an IIR filter. The authors also concluded that high power amplifiers are better modeled using IIR Hammerstein systems than using FIR Wiener systems, since the IIR Hammerstein models need less parameters, are easier to estimate and present better performance. Therefore, the predistortion of IIR Hammerstein systems using the DLA approach is introduced in this section. The NFxLMS and NFxPEM algorithms are developed for estimating the coefficients of the predistorter. The validity of the algorithms is demonstrated via computer simulation.

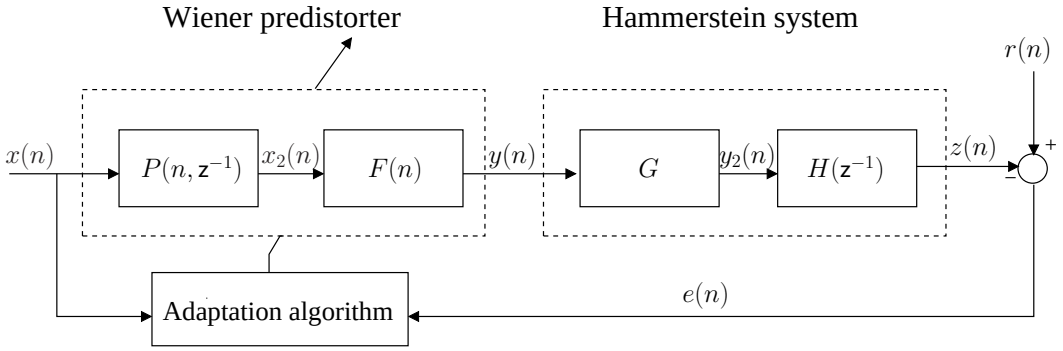

Figure 2.9 The DLA approach for predistortion of Hammerstein systems.

The DLA approach for predistortion of Hammerstein systems is shown in Fig. 2.9. The output of the IIR Hammerstein system is

$$\begin{aligned} z(n) &= H(\mathsf{z}^{-1})y_2(n) = \frac{B(\mathsf{z}^{-1})}{1 - A(\mathsf{z}^{-1})}y_2(n) \\ &= \sum_{m=0}^{m_b} b_m y_2(n-m) + \sum_{m=1}^{m_a} a_m z(n-m) \end{aligned} \tag{2.82}$$

where $H(\mathsf{z}^{-1}) = \frac{B(\mathsf{z}^{-1})}{1-A(\mathsf{z}^{-1})}$ is an IIR filter and the polynomials $A(\mathsf{z}^{-1})$ and $B(\mathsf{z}^{-1})$ are defined as

$$\begin{aligned} A(\mathsf{z}^{-1}) &= \sum_{m=1}^{m_a} a_m \mathsf{z}^{-m} \\ B(\mathsf{z}^{-1}) &= \sum_{m=0}^{m_b} b_m \mathsf{z}^{-m}. \end{aligned} \tag{2.83}$$

Here z^{-1} is the delay operator such that $\mathsf{z}^{-m}x(n) = x(n-m)$. The intermediate signal $y_2(n)$ is defined as

$$\begin{aligned} y_2(n) &= g_1 y(n) + g_2 y^2(n) + \cdots + g_{m_g} y^{m_g}(n) \\ &= \boldsymbol{g}^T \boldsymbol{y}(n) \end{aligned} \tag{2.84}$$

where $\boldsymbol{g}$ is the parameter vector of the nonlinear subsystem G defined as

$$\boldsymbol{g} = \begin{pmatrix} g_1 & g_2 & \cdots & g_{m_g} \end{pmatrix}^T \tag{2.85}$$

and the corresponding input vector $\boldsymbol{y}(n)$ is given by

$$\boldsymbol{y}(n) = \begin{pmatrix} y(n) & y^2(n) & \cdots & y^{m_g}(n) \end{pmatrix}^T. \tag{2.86}$$

The predistorter of the IIR Hammerstein system is chosen as an IIR Wiener system. The output of the predistorter is

$$\begin{aligned} y(n) &= f_1(n)x_2(n) + f_2(n)x_2^2(n) + \cdots + f_{m_f}x_2^{m_f}(n) \\ &= \boldsymbol{f}^T(n)\boldsymbol{x}_2(n) \end{aligned} \tag{2.87}$$

where $\boldsymbol{f}(n)$ is the parameter vector of the nonlinear subsystem $F(n)$ defined as

$$\boldsymbol{f}(n) = \begin{pmatrix} f_1(n) & f_2(n) & \cdots & f_{m_f}(n) \end{pmatrix}^T \tag{2.88}$$

and the corresponding input vector $\boldsymbol{x}_2(n)$ is given by

$$\boldsymbol{x}_2(n) = \begin{pmatrix} x_2(n) & x_2^2(n) & \cdots & x_2^{m_f}(n) \end{pmatrix}^T. \tag{2.89}$$

The intermediate signal $x_2(n)$ is given by

$$\begin{aligned} x_2(n) &= P(n,\mathsf{z}^{-1})x(n) = \frac{D(n,\mathsf{z}^{-1})}{1 - C(n,\mathsf{z}^{-1})}x(n) \\ &= \sum_{m=0}^{m_d} d_m(n)x(n-m) + \sum_{m=1}^{m_c} c_m(n)x_2(n-m). \end{aligned} \tag{2.90}$$

where $P(n,\mathsf{z}^{-1}) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}$ is an IIR filter and the polynomials $C(n,\mathsf{z}^{-1})$ and $D(n,\mathsf{z}^{-1})$ are defined as

$$\begin{aligned} C(n,\mathsf{z}^{-1}) &= \sum_{m=1}^{m_c} c_m(n)\mathsf{z}^{-m} \\ D(n,\mathsf{z}^{-1}) &= \sum_{m=0}^{m_d} d_m(n)\mathsf{z}^{-m}. \end{aligned} \tag{2.91}$$

Let us define the parameter vector $\boldsymbol{\theta}$ of the predistorter as follows

$$\begin{aligned} \boldsymbol{\theta} &= \begin{pmatrix} \boldsymbol{\theta}_f^T & \boldsymbol{\theta}_d^T & \boldsymbol{\theta}_c^T \end{pmatrix}^T \\ \boldsymbol{\theta}_f &= \begin{pmatrix} f_1 & f_2 & \cdots & f_{m_f} \end{pmatrix}^T \\ \boldsymbol{\theta}_d &= \begin{pmatrix} d_0 & d_1 & \cdots & d_{m_d} \end{pmatrix}^T \\ \boldsymbol{\theta}_c &= \begin{pmatrix} c_1 & c_2 & \cdots & c_{m_c} \end{pmatrix}^T. \end{aligned} \tag{2.92}$$

The NFxLMS and NFxPEM algorithms for estimating the parameter vector $\boldsymbol{\theta}$ have been developed for the first time in our publication [35, 39] and they will be introduced in the following sections.

2.4.1. The NFxLMS algorithm

The NFxLMS algorithm is obtained by applying the stochastic gradient algorithm [56, 63]:

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) - \frac{\mu}{2}\boldsymbol{\Delta}^T(n) \tag{2.93}$$

where μ is the *step-size* parameter. Also, $\boldsymbol{\Delta}(n)$ is the gradient vector which is defined as

$$\boldsymbol{\Delta}(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} e^2(n) = -2e(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n). \tag{2.94}$$

Using (2.82), $\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n)$ can be derived as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n) = \sum_{m=0}^{m_b} b_m \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n-m) + \sum_{m=1}^{m_a} a_m \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n-m). \tag{2.95}$$

Assuming that the parameter vector $\boldsymbol{\theta}(n)$ changes slowly [23,26], the following approximations can be made:

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n-m) &\approx \boldsymbol{\nabla}_{\boldsymbol{\theta}(n-m)} y_2(n-m), \quad m = 0, 1, \cdots, m_b \\ \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n-m) &\approx \boldsymbol{\nabla}_{\boldsymbol{\theta}(n-m)} z(n-m), \quad m = 1, 2, \cdots, m_a. \end{aligned} \tag{2.96}$$

Consequently, (2.95) can be written as

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n) &\approx \sum_{m=0}^{m_b} b_m \boldsymbol{\nabla}_{\boldsymbol{\theta}(n-m)} y_2(n-m) + \sum_{m=1}^{m_a} a_m \boldsymbol{\nabla}_{\boldsymbol{\theta}(n-m)} z(n-m) \\ &= \frac{B(\mathsf{z}^{-1})}{1 - A(\mathsf{z}^{-1})} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n) = H(\mathsf{z}^{-1}) \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n). \end{aligned} \tag{2.97}$$

From (2.84)-(2.86), we have

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y_2(n) = \boldsymbol{g}^T \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} \boldsymbol{y}(n) = \boldsymbol{g}^T \boldsymbol{\nabla}_{y(n)} \boldsymbol{y}(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y(n) = s_1(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y(n) \tag{2.98}$$

where

$$s_1(n) = \boldsymbol{g}^T \boldsymbol{\nabla}_{y(n)} \boldsymbol{y}(n) = \boldsymbol{g}^T \begin{pmatrix} 1 \\ 2y(n) \\ \vdots \\ m_g y^{m_g - 1}(n) \end{pmatrix}. \tag{2.99}$$

Using (2.92) and (2.98), (2.97) becomes

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n) &= H(\mathsf{z}^{-1}) s_1(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} y(n) \\ &= H(\mathsf{z}^{-1}) s_1(n) \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n) \end{pmatrix}. \end{aligned} \tag{2.100}$$

Considering (2.87), (2.89) and (2.90), $\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} y(n)$ can be derived as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} y(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} \left(\boldsymbol{\theta}_f^T(n) \boldsymbol{x}_2(n) \right) = \boldsymbol{x}_2^T(n) = \begin{pmatrix} P(n, \mathsf{z}^{-1}) x(n) \\ \left[P(n, \mathsf{z}^{-1}) x(n) \right]^2 \\ \vdots \\ \left[P(n, \mathsf{z}^{-1}) x(n) \right]^{m_f} \end{pmatrix}^T. \tag{2.101}$$

Note that the intermediate signal $x_2(n)$ should be estimated since it is usually not measurable. Again, using (2.87), (2.89) and (2.90), $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n)$ can be derived as

$$\begin{aligned}\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n) &= \boldsymbol{\theta}_f^T(n) \boldsymbol{\nabla}_{x_2(n)} \boldsymbol{x}_2(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} x_2(n) = s_2(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} x_2(n) \\ \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n) &= \boldsymbol{\theta}_f^T(n) \boldsymbol{\nabla}_{x_2(n)} \boldsymbol{x}_2(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} x_2(n) = s_2(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} x_2(n)\end{aligned} \tag{2.102}$$

where

$$\begin{aligned} s_2(n) &= \boldsymbol{\theta}_f^T(n) \boldsymbol{\nabla}_{x_2(n)} \boldsymbol{x}_2(n) = \boldsymbol{\theta}_f^T(n) \begin{pmatrix} 1 \\ 2x_2(n) \\ \vdots \\ m_f x_2^{m_f-1}(n) \end{pmatrix} \\ &= \boldsymbol{\theta}_f^T(n) \begin{pmatrix} 1 \\ 2\left[P(n, \mathsf{z}^{-1}) x(n)\right] \\ \vdots \\ m_f \left[P(n, \mathsf{z}^{-1}) x(n)\right]^{m_f-1} \end{pmatrix}. \end{aligned} \tag{2.103}$$

Now, it remains to derive $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} x_2(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} x_2(n)$. Differentiating both sides of (2.90) with respect to $d_k(n)$ and $c_k(n)$ gives

$$\begin{aligned} \frac{\partial x_2(n)}{\partial d_k(n)} &= x(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial x_2(n-m)}{\partial d_k(n)} \\ \frac{\partial x_2(n)}{\partial c_k(n)} &= x_2(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial x_2(n-m)}{\partial c_k(n)}. \end{aligned} \tag{2.104}$$

Since the parameter vector $\boldsymbol{\theta}(n)$ is assumed to be changing slowly, we can write

$$\begin{aligned} \frac{\partial x_2(n-m)}{\partial d_k(n)} &\approx \frac{\partial x_2(n-m)}{\partial d_k(n-m)}, \quad m = 1, 2, \cdots, m_c \\ \frac{\partial x_2(n-m)}{\partial c_k(n)} &\approx \frac{\partial x_2(n-m)}{\partial c_k(n-m)}, \quad m = 1, 2, \cdots, m_c. \end{aligned} \tag{2.105}$$

Hence, (2.104) can be rewritten as

$$\begin{aligned} \frac{\partial x_2(n)}{\partial d_k(n)} &\approx x(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial x_2(n-m)}{\partial d_k(n-m)} \\ \frac{\partial x_2(n)}{\partial c_k(n)} &\approx x_2(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial x_2(n-m)}{\partial c_k(n-m)} \end{aligned} \tag{2.106}$$

or

$$\begin{aligned} \frac{\partial x_2(n)}{\partial d_k(n)} &\approx \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} x(n), \quad k = 0, 1, \cdots, m_d \\ \frac{\partial x_2(n)}{\partial c_k(n)} &\approx \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} x_2(n) \\ &= \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} \left[P(n, \mathsf{z}^{-1}) x(n)\right], \quad k = 1, \cdots, m_c. \end{aligned} \tag{2.107}$$

Now, we have completely derived the components of $\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n)$ in (2.100) and the gradient vector $\boldsymbol{\Delta}(n)$ can be written as

$$\boldsymbol{\Delta}(n) = -2e(n)H(\mathsf{z}^{-1})s_1(n)\begin{pmatrix}\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} y(n) & s_2(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} x_2(n) & s_2(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} x_2(n)\end{pmatrix} \quad (2.108)$$

where $s_1(n)$ and $s_2(n)$ are calculated in (2.99) and (2.103), respectively, and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} y(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} x_2(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} x_2(n)$ are given by (2.101) and (2.107), respectively.

2.4.2. The NFxPEM algorithm

The NFxPEM algorithm is derived by the minimization of the cost function [57]

$$V(\boldsymbol{\theta}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n)\right] \quad (2.109)$$

where $e(n)$ is the prediction error which is defined as

$$e(n) = r(n) - z(n). \quad (2.110)$$

The formulation of the NFxPEM algorithm requires the negative gradient of $e(n)$ w.r.t. $\boldsymbol{\theta}(n)$ which is defined as

$$\boldsymbol{\varphi}^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} e(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} z(n). \quad (2.111)$$

Straightforward analysis identical to Section 2.4.1 gives

$$\boldsymbol{\varphi}^T(n) = H(\mathsf{z}^{-1})s_1(n)\begin{pmatrix}\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} y(n) & s_2(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} x_2(n) & s_2(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} x_2(n)\end{pmatrix} \quad (2.112)$$

where $s_1(n)$ and $s_2(n)$ are calculated as in (2.99) and (2.103), respectively, and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} y(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} x_2(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} x_2(n)$ are given by (2.101) and (2.107), respectively. Hence, the NFxPEM algorithm follows as (*cf.* [57, 63])

$$\begin{aligned}
&e(n) = r(n) - z(n)\\
&\lambda(n) = \lambda_0\lambda(n-1) + 1 - \lambda_0\\
&s(n) = \boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\varphi}(n) + \lambda(n)\\
&\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1) - \boldsymbol{P}(n-1)\boldsymbol{\varphi}(n)s^{-1}(n)\boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\big)/\lambda(n)\\
&\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \boldsymbol{P}(n)\boldsymbol{\varphi}(n)e(n).
\end{aligned} \quad (2.113)$$

Here, λ_0, $\lambda(0)$ and $\boldsymbol{P}(0)$ are design variables as introduced in Section 2.2.3.

2.4.3. Simulation study

In this simulation study, the following IIR Hammerstein system was considered:

$$\begin{aligned}
z(n) &= \frac{0.72 + 1.51\mathsf{z}^{-1} + 1.04\mathsf{z}^{-2} + 0.26\mathsf{z}^{-3}}{1 + 1.46\mathsf{z}^{-1} + 0.89\mathsf{z}^{-2} + 0.18\mathsf{z}^{-3}} y_2(n)\\
y_2(n) &= y(n) + 0.25y^2(n) + 0.125y^3(n).
\end{aligned} \quad (2.114)$$

The order of the linear block of the IIR Wiener predistorter was chosen as $m_c = 3$ and $m_d = 3$, in order to avoid model error. The order of the nonlinear block was chosen as $m_g = 9$, since it can achieve the best performance based on large number of experiments for different values. The number of independent experiments was 100 and in each experiment, the input signal was a random signal with uniform distribution over $(-1, 1)$ and data length of 2×10^5 samples. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{5}$. The output measurement noise was considered as AWGN with SNR=40 dB.

The parameter vectors $\boldsymbol{\theta}$ were initialized as

$$\begin{aligned} \boldsymbol{\theta}_f(0) &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}^T \\ \boldsymbol{\theta}_d(0) &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T \\ \boldsymbol{\theta}_c(0) &= \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T. \end{aligned} \tag{2.115}$$

The MSD comparison between the NFxLMS and NFxPEM algorithms is given in Fig. 2.10. The MSD of the IIR Hammerstein system without predistorter was about -18 dB in noise-free scenario. The step size of the NFxLMS algorithm was $\mu = 0.05$ and the matrix $\mathbf{P}(0) = \mathbf{I}$ for the NFxPEM algorithms. λ_0 and $\lambda(0)$ were chosen as 0.99 and 0.95, respectively. The predistorters using these algorithms achieve similar MSD performance in the steady state. On average, the NFxLMS algorithm converges after about 1.5×10^5 samples and achieves about -39 dB. The NFxPEM algorithm converges after about 1000 samples and achieves about -40 dB. On the other hand, the NFxPEM algorithm converges much faster than the NFxLMS algorithm.

Figure 2.11 shows the mean PSDs of the output signals of the IIR Hammerstein system without and with predistorter after 2×10^5 samples. From this figure, we can see that both algorithms can effectively reduce the spectral regrowth in the normalized frequency band $(0.25\pi, 0.65\pi)$. The NFxLMS algorithm can achieve up to 30 dB reduction. The NFxPEM algorithm has even better performance (up to 60 dB reduction) compared to the NFxLMS algorithm. For all algorithms, there is spectral regrowth in the normalized frequency band $(0.65\pi, \pi)$, which is caused by the cascaded nonlinear systems. However, this spectral regrowth has no significant impact on the performance of predistortion since it is relatively small (under -70 dB).

2.5. Predistortion Using the SMM Method

The SMM method is first proposed in [61] to identify the telephone handset nonlinearity by matching the spectral magnitude of the distorted signal to the output of a nonlinear model. This method is introduced in this section for predistortion of nonlinear systems. The SMM method is first described in a general formulation and then implemented for predistortion of Volterra, Wiener and Hammerstein systems, respectively. The validity of the method is demonstrated via computer simulation.

The DLA approach using the SMM method is shown in Fig. 2.12. The reference signal $r(n)$ is defined as

$$r(n) = x(n - \tau) + \upsilon(n) \tag{2.116}$$

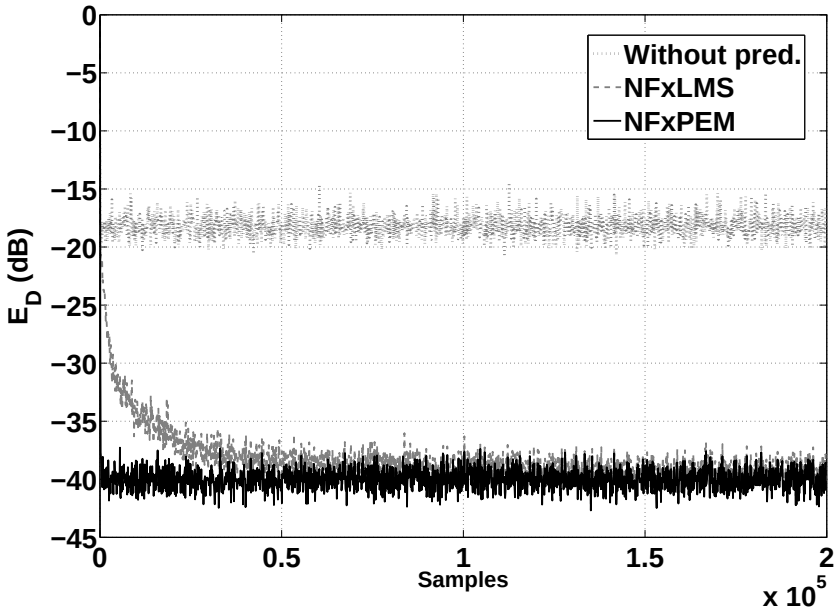

Figure 2.10 MSD E_D for different adaptation algorithms.

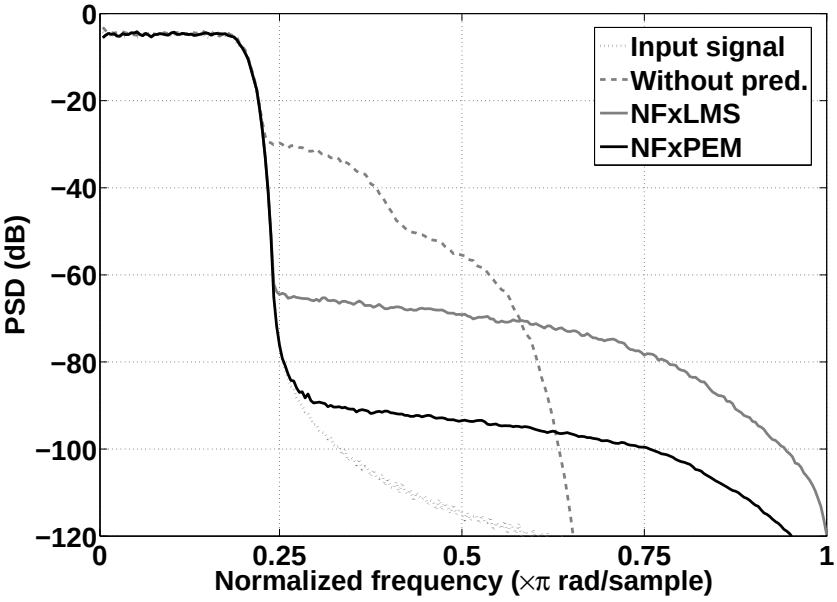

Figure 2.11 Mean PSDs for different adaptation algorithms.

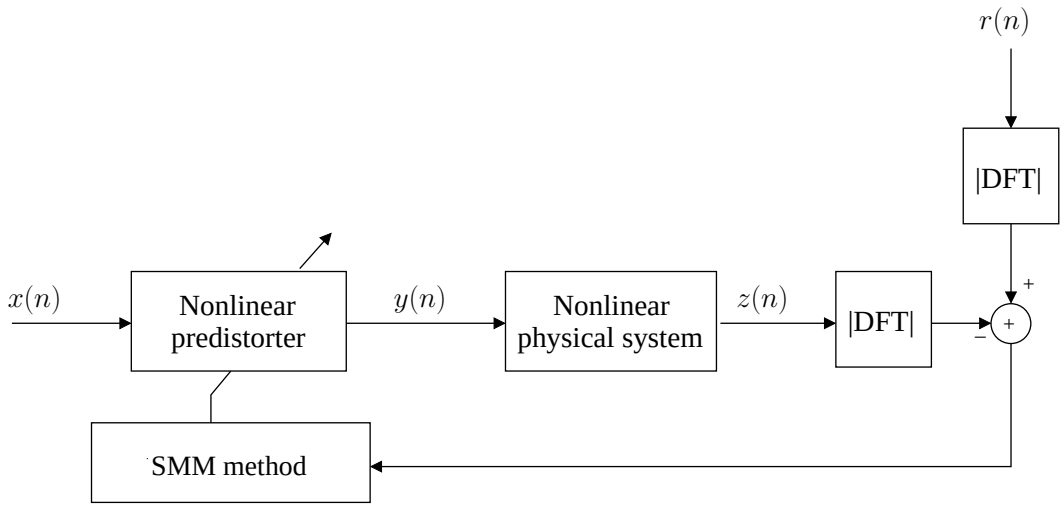

Figure 2.12 The DLA approach using the SMM method.

where τ is the time delay and $\upsilon(n)$ is AWGN. The nonlinear physical system here is a baseband nonlinear system and the input signal is a real value signal. The predistorter intends to reduce the spectral magnitude regrowth of the signal at the output of the nonlinear physical system. Therefore, by defining $\boldsymbol{\theta}$ as the parameter vector of the adaptive predistorter, the goal of the SMM method is to estimate the parameter vector $\boldsymbol{\theta}$ by minimizing the sum squared error between the spectral magnitude of the reference signal $r(n)$ and the spectral magnitude of the received output signal $z(n)$ through the following cost function:

$$V_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} \big[|R(l,k)| - |Z(l,k,\boldsymbol{\theta})|\big]^2 \tag{2.117}$$

where $R(l,k)$ and $Z(l,k,\boldsymbol{\theta})$ are the short-time DFTs of $r(n)$ and $z(n)$, respectively. K is the number of uniformly-spaced short-time frames and L is the DFT length. Note that the phase of the DFT is neglected since reducing the phase distortion here is less important than reducing the spectral regrowth.

$R(l,k)$ can be generated as in Fig. 2.13, where $\boldsymbol{R}(k)$ is the short-time DFT vector defined as

$$\boldsymbol{R}(k) = \begin{pmatrix} R(0,k) & \cdots & R(l,k) & \cdots & R(L-1,k) \end{pmatrix}^T, \quad k = 0, \cdots, K-1. \tag{2.118}$$

$Z(l,k,\boldsymbol{\theta})$ can also be generated using the same structure.

2.5.1. The SMM method

The cost function $V_{\boldsymbol{\theta}}$ can be rewritten as

$$V_{\boldsymbol{\theta}} = \boldsymbol{\Gamma}_{\boldsymbol{\theta}}^T \boldsymbol{\Gamma}_{\boldsymbol{\theta}} \tag{2.119}$$

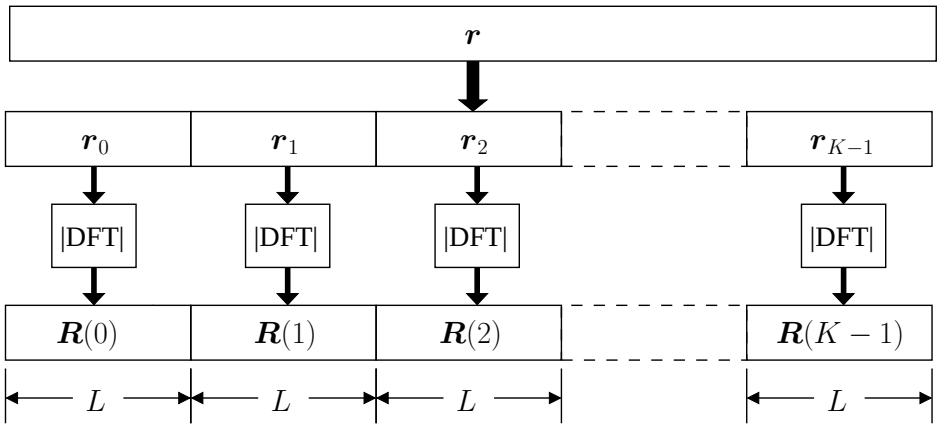

Figure 2.13 Generation of the short-time DFT vectors $\boldsymbol{R}(k)$.

where

$$\boldsymbol{\Gamma}_{\boldsymbol{\theta}} = \begin{pmatrix} \boldsymbol{\gamma}^0(\boldsymbol{\theta}) \\ \boldsymbol{\gamma}^1(\boldsymbol{\theta}) \\ \vdots \\ \boldsymbol{\gamma}^{K-1}(\boldsymbol{\theta}) \end{pmatrix} \tag{2.120}$$

and

$$\boldsymbol{\gamma}^k(\boldsymbol{\theta}) = \begin{pmatrix} |R(0,k)| - |Z(0,k,\boldsymbol{\theta})| \\ \vdots \\ |R(L-1,k)| - |Z(L-1,k,\boldsymbol{\theta})| \end{pmatrix}, \quad k = 0, \cdots, K-1. \tag{2.121}$$

In the SMM method, the parameter vector $\boldsymbol{\theta}$ that minimizes the cost function $V_{\boldsymbol{\theta}}$ can be estimated using the generalized Newton iteration [49,62]. In each iteration, the output signal of the nonlinear physical system with the predistorter is generated using the same series of input signals. The parameter vector $\boldsymbol{\theta}$ is updated by

$$\boldsymbol{\theta}(m+1) = \boldsymbol{\theta}(m) + \mu\ \boldsymbol{\Delta}(m) \tag{2.122}$$

where m is the iteration index, μ is the *step size*, and $\boldsymbol{\Delta}(m)$ is given by [57,61,63]

$$\boldsymbol{\Delta}(m) = -\left[\boldsymbol{\nabla}^2_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}\right]^{-1} \left[\boldsymbol{\nabla}_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}\right] |_{\boldsymbol{\theta}=\boldsymbol{\theta}(m)} = -\left(\boldsymbol{J}^T(m)\boldsymbol{J}(m)\right)^{-1} \boldsymbol{J}^T(m)\ \boldsymbol{\Gamma}_{\boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}(m)}. \tag{2.123}$$

Here $\boldsymbol{J}(m)$ is the Jacobian matrix of first derivative of $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ with respect to $\boldsymbol{\theta}(m)$, *i.e.*,

$$\boldsymbol{J}(m) = \boldsymbol{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\Gamma}_{\boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}(m)} = \begin{pmatrix} \boldsymbol{J}^0(m) \\ \boldsymbol{J}^1(m) \\ \vdots \\ \boldsymbol{J}^{K-1}(m) \end{pmatrix} \tag{2.124}$$

where

$$\boldsymbol{J}^k(m) = \boldsymbol{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\gamma}^k(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}(m)} = -\begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}} |Z(0,k,\boldsymbol{\theta})| \\ \vdots \\ \boldsymbol{\nabla}_{\boldsymbol{\theta}} |Z(L-1,k,\boldsymbol{\theta})| \end{pmatrix} |_{\boldsymbol{\theta}=\boldsymbol{\theta}(m)}, \quad k = 0, \cdots, K-1. \tag{2.125}$$

Due to the fact that there is no closed form expression for $\boldsymbol{\Delta}(m)$, an approximate gradient was evaluated in [61] by finite element approximation. The same approach is considered here. The approximation follows the following lines:

1. Initialize with a parameter vector $\boldsymbol{\theta}(0)$ and compute the DFT magnitude $|R(l,k)|$.

2. Compute the DFT magnitude $|Z(l,k,\boldsymbol{\theta})|$ based on the current value of the parameter vector $\boldsymbol{\theta}(m)$ and form $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$.

3. Recalculate $z(n,\boldsymbol{\theta})$ for each perturbed component of $\boldsymbol{\theta}(m)$ separately and then compute its DFT magnitude, *i.e.*, the DFT magnitude vector for the kth frame when the jth component is perturbed is $\boldsymbol{Z}(k,\theta_1(m),...,\theta_j(m)+\varepsilon_m,...)$. The (i,j)th element of the matrix element $\boldsymbol{J}^k(m)$, denoted as $J^k_{i,j}(m)$, is evaluated using finite element approximation for each element of $\boldsymbol{\theta}(m)$ as

$$\begin{aligned} J^k_{i,j}(m) = \boldsymbol{\nabla}_{\theta_j}\gamma^k_i(m,\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}(m)} \quad &\approx \quad -\frac{1}{\varepsilon_m}(|Z(i,k,\theta_1(m),\cdots,\theta_j(m)+\varepsilon_m,\cdots)| \\ &\quad -|Z(i,k,\theta_1(m),\cdots,\theta_j(m),\cdots)|) \end{aligned} \tag{2.126}$$

   where $\gamma^k_i(m,\boldsymbol{\theta})$ is the ith element of $\boldsymbol{\gamma}^k(\boldsymbol{\theta})$, $\theta_j(m)$ is the jth element of the parameter vector $\boldsymbol{\theta}(m)$ and ε_m is a small adaptive perturbation evaluated as

$$\varepsilon_m = \frac{V_{\boldsymbol{\theta}}(m)}{V_{\boldsymbol{\theta}}(0)}\varepsilon_0 \tag{2.127}$$

   where ε_0 is the initial perturbation, $V_{\boldsymbol{\theta}}(0)$ is the initial value of $V_{\boldsymbol{\theta}}$, and $V_{\boldsymbol{\theta}}(m)$ is the mth step value of $V_{\boldsymbol{\theta}}$. This means that the perturbation decreases proportionally with the error, *i.e.*, $V_{\boldsymbol{\theta}}(m)\to 0$ when $\varepsilon_m \to 0$.

4. Finally, evaluate the correction term $\boldsymbol{\Delta}(m)$ from (2.123) and update the parameter vector $\boldsymbol{\theta}$ using (2.122).

Remark 2.4: The calculation of $\boldsymbol{\Delta}(m)$ in each iteration only needs to observe the output signals of the nonlinear physical system - $z(n,\boldsymbol{\theta})$ for $\boldsymbol{\theta}(m)$ and for all the component-wise perturbed version of $\boldsymbol{\theta}(m)$. Therefore, the SMM method can be used for predistortion of nonlinear physical systems without knowing the system information before hand. After choosing the model of the predistorter, its coefficients can be estimated using the procedure mentioned before. However, this advantage comes at a huge price because, for each iteration, we need to perform measurement on the physical output of the nonlinear physical system driven by as many perturbed versions of the predistorter.

The flowchart of the SMM method is summarized in Fig. 2.14. Here, N is the number of parameters in the parameter vector $\boldsymbol{\theta}$. The iteration can stop when m has reached the pre-defined iteration number or the parameter vector $\boldsymbol{\theta}$ has converged to the steady state.

2.5.2. Simulation study

In this section, predistortion using the SMM method is applied to different nonlinear physical systems - Volterra, Wiener and Hammerstein systems. Because the SMM method intends to

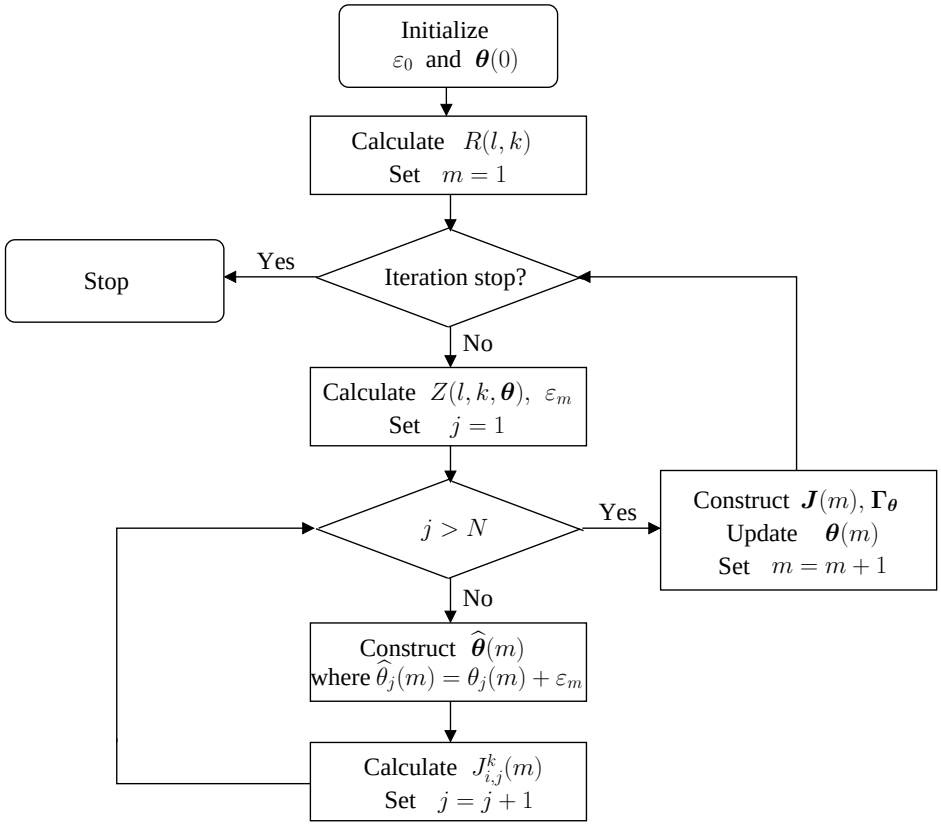

Figure 2.14 Flowchart of the SMM method.

minimize the sum squared error between the spectral magnitudes of $r(n)$ and $z(n)$, we define the normalized spectral magnitude sum squared error, which is

$$E_{SM}(m) = 10\log_{10}\left(\frac{\sum_{k=0}^{K-1}\sum_{l=0}^{L-1}\left[|R(l,k)|-|Z(l,k,\boldsymbol{\theta})|\right]^2}{\sum_{k=0}^{K-1}\sum_{l=0}^{L-1}\left[R(l,k)\right]^2}\right)|_{\boldsymbol{\theta}=\boldsymbol{\theta}(m)}. \tag{2.128}$$

In each independent experiment, E_{SM} is evaluated in each iteration in order to observe whether the spectral magnitude difference between $r(n)$ and $z(n)$ is well reduced. The mean PSD of $z(n)$ is also estimated after the parameter vector $\boldsymbol{\theta}$ converges to the steady state.

Predistortion of Volterra systems

The nonlinear physical system was the 2nd-order Volterra system defined as in Section 2.2.4. The output $z(n)$ was

$$z(n) = \boldsymbol{h}_1^T\boldsymbol{y}_1(n) + \boldsymbol{h}_2^T\boldsymbol{y}_2(n) \tag{2.129}$$

where the first-order kernel vector $\boldsymbol{h}_1$ was

$$\boldsymbol{h}_1 = \begin{pmatrix} 0.5625 & 0.4810 & 0.1124 & -0.1669 \end{pmatrix} \tag{2.130}$$

and the second-order kernel vector $\boldsymbol{h}_2$ was

$$\boldsymbol{h}_2 = (0.0175 \quad 0 \quad 0 \quad 0 \quad 0 \quad -0.0088 \quad 0 \quad -0.0088 \quad 0)\,. \tag{2.131}$$

The adaptive predistorter was also assumed to be a 2nd-order Volterra filter defined identical to Section 2.2.4.

The number of independent experiments was 100 and in each experiment, the input signal was chosen as a random signal with uniform distribution over $(-1, 1)$. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{4}$. For the SMM approach, a data length of 10×2^8 input samples were divided into 10 short-time frames, each with length 2^8 samples. The DFT length L was 2^8. An initial perturbation of $\varepsilon_0 = 0.001$ and adaptation gain of $\mu = 0.5$ were used.

Figure 2.15 demonstrates the E_{SM} evaluated in one experiment. E_{SM} achieves a stable value of about -60 dB after about 20 iterations, which means that the parameter vector $\boldsymbol{\theta}$ converges to the steady state.

The mean PSDs of the output signals of the Volterra system without and with predistorter after 30 iterations are shown in Fig. 2.16. From this figure, we can see that the SMM method can effectively reduce the spectral regrowth in the normalized frequency band $(0.30\pi, 0.55\pi)$. Compared to the simulation result in Fig. 2.4, the SMM method can achieve similar performance as the NFxRLS and NFxPEM algorithms (up to 30 dB reduction).

Predistortion of Wiener systems

The nonlinear physical system was an IIR Wiener system defined as in Section 2.3.4. The output $z(n)$ was

$$\begin{aligned} z(n) &= y_2(n) + 0.25y_2^2(n) + 0.125y_2^3(n) \\ y_2(n) &= \frac{0.72 + 1.51\mathsf{z}^{-1} + 1.04\mathsf{z}^{-2} + 0.26\mathsf{z}^{-3}}{1 + 1.46\mathsf{z}^{-1} + 0.89\mathsf{z}^{-2} + 0.18\mathsf{z}^{-3}} y(n). \end{aligned} \tag{2.132}$$

The predistorter was modeled as an IIR Hammerstein system defined identical to Section 2.3.4. The number of independent experiments was 100 and in each experiment, the input signal was chosen as a random signal with uniform distribution over $(-1, 1)$. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{5}$. For the SMM approach, a data length of 10×2^8 input samples were divided into 20 short-time frames have been used each with length 2^8 samples. The DFT length L was 2^8. An initial perturbation of $\varepsilon_0 = 0.001$ and adaptation gain of $\mu = 0.04$ were used.

Figure 2.17 demonstrates the E_{SM} evaluated in one experiment. E_{SM} achieves a stable value of about -62 dB and the parameter vector $\boldsymbol{\theta}$ converges to the steady state after about 125 iterations.

The mean PSDs of the output signals of the IIR Wiener system without and with predistorter after 200 iterations are shown in Fig. 2.18. From this figure, we can see that the SMM method can effectively reduce the spectral regrowth in the normalized frequency band $(0.25\pi, 0.60\pi)$. Compared to the simulation result in Fig. 2.8, the SMM method can achieve similar performance as the NFxPEM algorithms (up to 40 dB reduction).

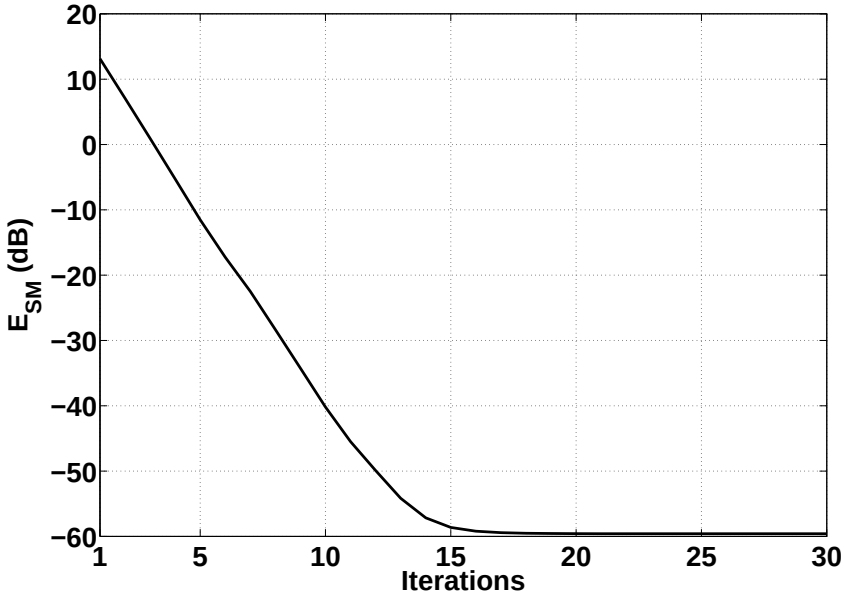

Figure 2.15 E_{SM} for the SMM method in one experiment for predistortion of Volterra system.

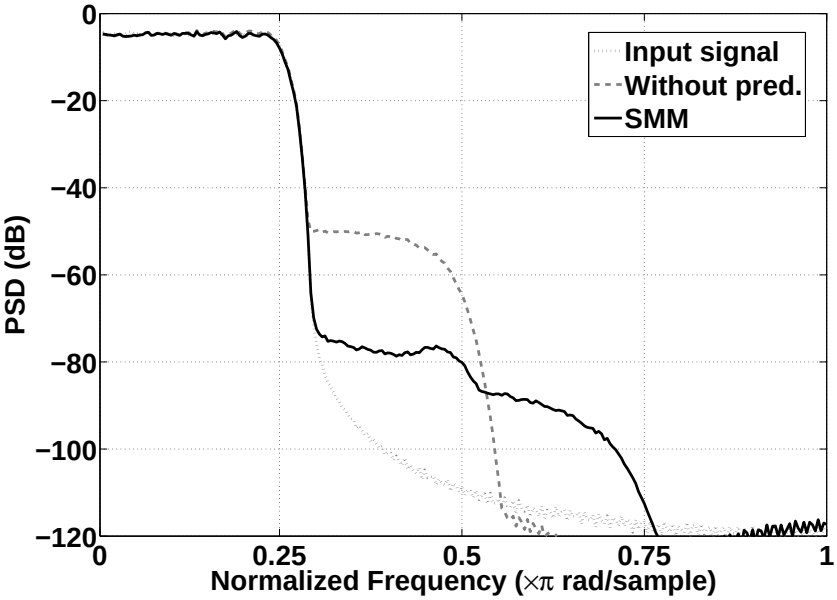

Figure 2.16 Mean PSDs without and with predistorter for predistortion of Volterra system.

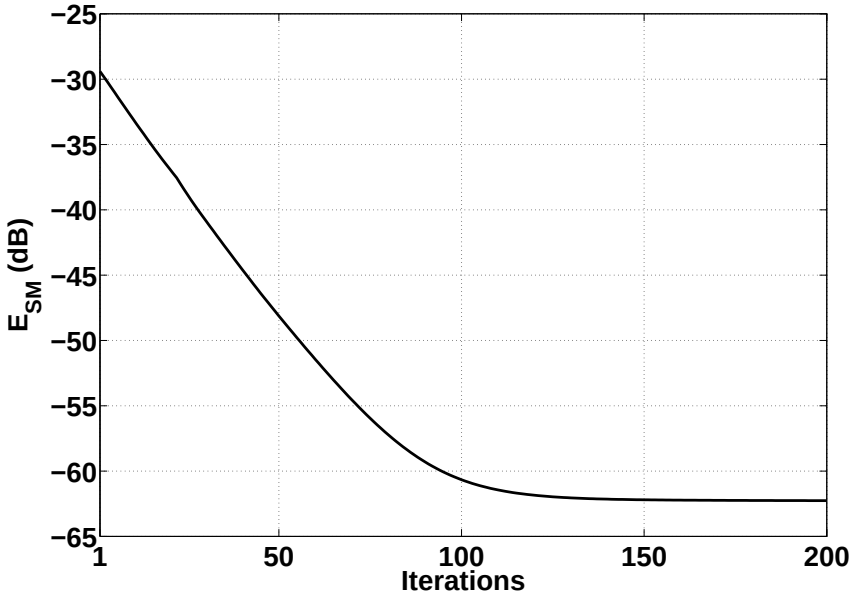

Figure 2.17 E_{SM} for the SMM method in one experiment for predistortion of the Wiener system.

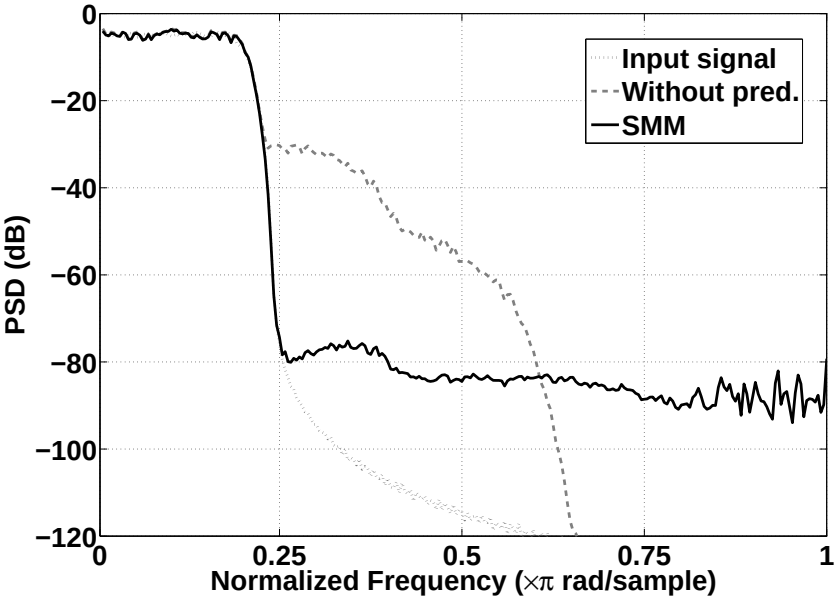

Figure 2.18 Mean PSDs without and with predistorter for predistortion of the Wiener system.

Predistortion of Hammerstein systems

The nonlinear physical system was an IIR Hammerstein system defined as in Section 2.4.3. The output signal $z(n)$ was

$$\begin{aligned} z(n) &= \frac{0.72 + 1.51\mathsf{z}^{-1} + 1.04\mathsf{z}^{-2} + 0.26\mathsf{z}^{-3}}{1 + 1.46\mathsf{z}^{-1} + 0.89\mathsf{z}^{-2} + 0.18\mathsf{z}^{-3}} y_2(n) \\ y_2(n) &= y(n) + 0.25y^2(n) + 0.125y^3(n). \end{aligned} \tag{2.133}$$

The predistorter was modeled as an IIR Wiener system defined identical to Section 2.4.3. The number of independent experiments was 100 and in each experiment, the input signal was chosen as a random signal with uniform distribution over $(-1, 1)$. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{5}$. For the SMM approach, a data length of 10×2^8 input samples divided into 10 short-time frames have been used each with length 2^8 samples. The DFT length L was 2^8. An initial perturbation of $\varepsilon_0 = 0.001$ and adaptation gain of $\mu = 0.04$ were used.

Figure 2.19 demonstrates the E_{SM} evaluated in one experiment. E_{SM} achieves stable value of about -62 dB and the parameter vector $\boldsymbol{\theta}$ converges to the steady state after about 125 iterations.

The mean PSDs of the output signals of the IIR Hammerstein system without and with predistorter after 200 iterations are shown in Fig. 2.20. From this figure, we can see that the SMM method can effectively reduce the spectral regrowth in the normalized frequency band $(0.25\pi, 0.65\pi)$. Compared to the simulation result in Fig. 2.11, the SMM method can achieve similar performance as the NFxPEM algorithms (up to 60 dB reduction).

2.6. Summary

Adaptive predistortion of nonlinear physical systems using the DLA approach is introduced in this chapter. The coefficients of the predistorter can be estimated using the time domain or frequency domain adaptation algorithms.

The NFxLMS algorithm is a fundamental adaptation algorithm in the time domain for predistortion of different nonlinear systems, such as Volterra, Wiener and Hammerstein systems. However, it requires accurate system identification of the nonlinear physical system and usually has very slow convergence speed. The existing NFxRLS algorithm can speed up the convergence but can only be used for predistortion of the nonlinear physical system, where the predistorter output is linear in its coefficients. The NFxPEM algorithm proposed in this chapter can be implemented for predistortion of different nonlinear systems. From the simulation, we can see that compared to the NFxLMS algorithm, the NFxPEM algorithm has very fast convergence speed and is much more efficient in reducing the spectral regrowth caused by the nonlinear physical system. For the predistortion of Wiener systems, the proposed NFxLMS-ISE and NFxPEM-ISE algorithms relax the accurate system identification requirement in NFxLMS and NFxPEM algorithms, by using the ISE instead. The NFxLMS-ISE and NFxPEM-ISE algorithms can achieve similar performance as the NFxLMS and NFxPEM algorithms with accurate system identification, respectively. To relax the accurate system identification requirement for predistortion of Volterra and Hammerstein systems is still an open issue in the future research.

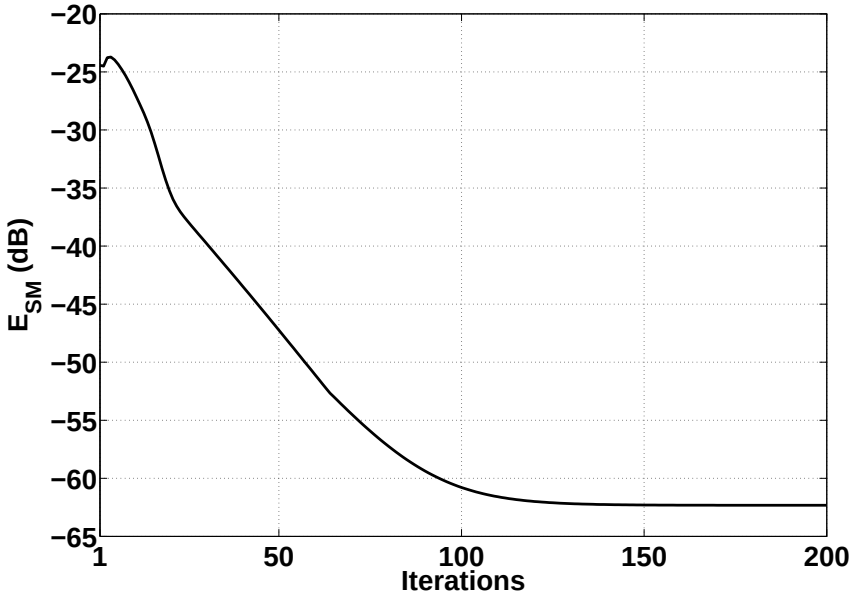

Figure 2.19 E_{SM} for the SMM method in one experiment for predistortion of the Hammerstein system.

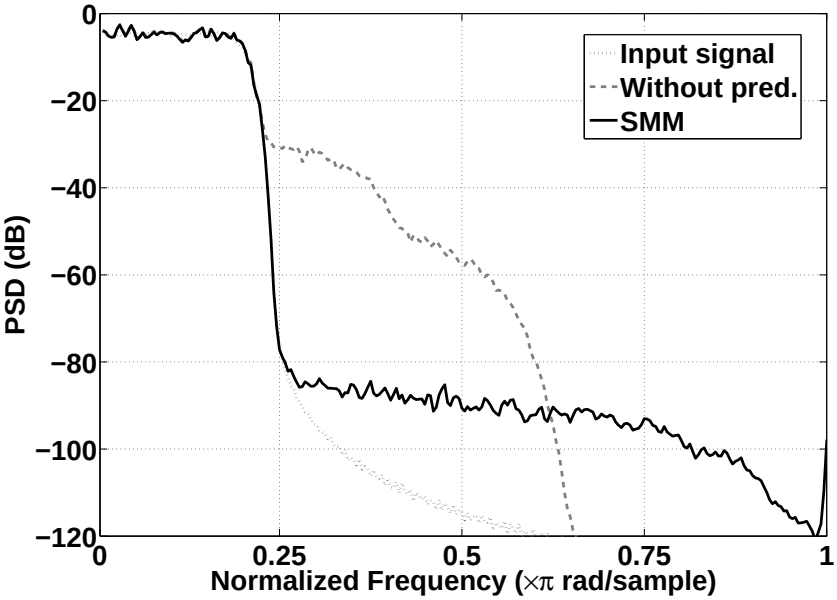

Figure 2.20 Mean PSDs without and with predistorter for predistortion of the Hammerstein system.

The proposed SMM method is a frequency domain technique to estimate the coefficients of the predistorter. It is also a general adaptation algorithm which can be used for predistortion of different nonlinear systems. This method has high computational complexity and requires matrix inversion in each adaptation iteration, and from the simulation, we can see that it has reasonable convergence speed and also good performance in reducing the spectral regrowth.

Chapter 3

Predistortion Using the Indirect Learning Architecture (ILA)

In this chapter the nonlinear physical system is modeled as Volterra, Wiener and Hammerstein systems, respectively, and the adaptation algorithms based on the Indirect Learning Architecture (ILA) approach are presented to estimate the coefficients of the predistorter. The ILA approach can be classified into the ILA-I approach and the ILA-II approach. The existing adaptation algorithms, such as the Recursive Least Squares (RLS) algorithm for predistortion of Volterra systems using the ILA-I approach and the Least Mean Squares (LMS) algorithm in the predistortion of Volterra systems using the ILA-II approach, are first reviewed. Then for the ILA-I approach, the Kalman Filter (KF), Recursive Prediction Error Method (RPEM) algorithms are derived for predistortion of Volterra systems. The RPEM algorithm is also derived for the predistortion of Wiener and Hammerstein systems. For the ILA-II approach, the RPEM algorithm is developed in order to improve the performance of the predistorter, as compared to the LMS algorithm.

This chapter is based on the publications [40, 41, 42, 43, 69], which are edited and refined in order to fit the current style of the thesis.

3.1. Introduction

Predistortion using the ILA approach usually requires extra filters in addition to the predistorter itself: the training filter in the ILA-I approach, and the adaptive linear and nonlinear FIR filters in the ILA-II approach.

In order to estimate the coefficients of the training filter in the ILA-I approach, an adaptation algorithm for predistortion of Volterra systems has been proposed in [27]. Both the training filter and the predistorter are modeled as Volterra systems and the coefficients of the training filter are estimated recursively using the RLS algorithm, see [57, 63]. In this chapter, the KF and RPEM algorithms [56, 57] are derived for predistortion of Volterra systems. Moreover, the RPEM algorithm is also derived for predistortion of Wiener and Hammerstein systems.

Furthermore, the RPEM algorithm is derived to estimate the coefficients of the adaptive linear and nonlinear FIR filters in the ILA-II approach, instead of the suggested LMS algorithm in [9]. The RPEM algorithm gives consistent parameter estimates under weak con-

ditions in case the asymptotic loss function has a unique stationary point which represents the true parameter vector. Therefore, using the RPEM algorithm is expected to provide more accurate estimates and hence to improve the performance of the predistorter.

In order to measure the performance of these approaches, the Mean Square Distortion (MSD) and the mean Power Spectral Density (PSD) as defined in Chapter 2 are used.

3.2. Predistortion of Volterra Systems

In this section, different adaptation algorithms for predistortion of Volterra systems using the ILA approach are introduced. For the ILA-I approach, the RLS algorithm is first reviewed, and then the KF and RPEM algorithms are derived. Also, the ILA-II approach using the LMS algorithm is first reviewed and then the RPEM algorithm is derived. Simulation results are given to verify the proposed algorithms.

3.2.1. The ILA-I approach

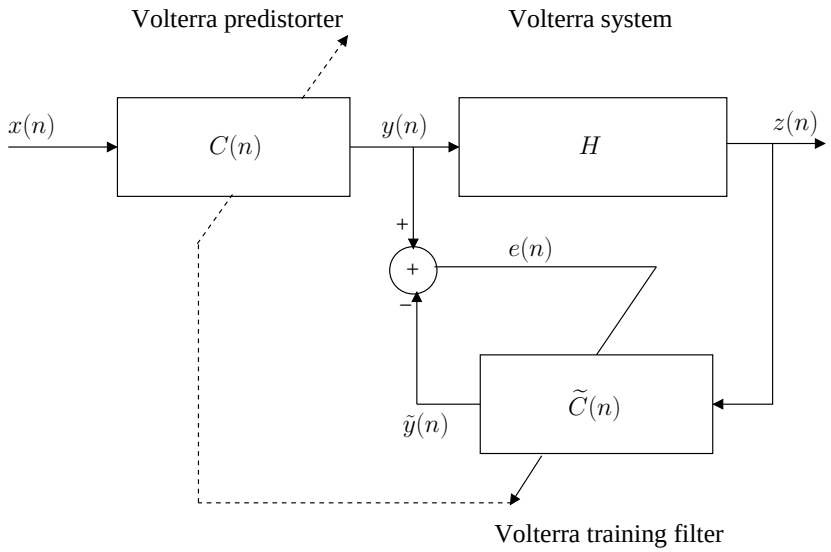

Figure 3.1 The ILA-I approach for predistortion of Volterra systems.

The ILA-I approach for predistortion of Volterra systems is shown in Fig. 3.1. The nonlinear physical system H is a qth-order Volterra system with input $y(n)$ and output $z(n)$. The output signal $z(n)$ is given by

$$\begin{aligned} z(n) &= \boldsymbol{h}^T \boldsymbol{y}(n) = \sum_{k=1}^{q} \boldsymbol{h}_k^T \boldsymbol{y}_k(n) \\ &= \sum_{k=1}^{q} \left(\sum_{i_1=0}^{M_k-1} \cdots \sum_{i_k=0}^{M_k-1} h_k(i_1, \cdots, i_k) y(n-i_1) \cdots y(n-i_k) \right) \end{aligned} \tag{3.1}$$

where M_k is the memory length for the kth order kernel. The kernel vector $\boldsymbol{h}$ is defined as

$$\boldsymbol{h} = \left(\begin{array}{ccc} \boldsymbol{h}_1^T & \cdots & \boldsymbol{h}_q^T \end{array} \right)^T, \tag{3.2}$$

$$\boldsymbol{h}_k = \begin{pmatrix} \boldsymbol{h}_k(0,\cdots,0) \\ \vdots \\ \boldsymbol{h}_k(i_1,\cdots,i_k) \\ \vdots \\ \boldsymbol{h}_k(M_k-1,\cdots,M_k-1) \end{pmatrix}, \quad k=1,\cdots,q, \tag{3.3}$$

and the input vector $\boldsymbol{y}(n)$ is defined as

$$\boldsymbol{y}(n) = \begin{pmatrix} \boldsymbol{y}_1^T(n) & \cdots & \boldsymbol{y}_q^T(n) \end{pmatrix}^T, \tag{3.4}$$

$$\boldsymbol{y}_k(n) = \begin{pmatrix} y^k(n) \\ \vdots \\ y(n-i_1)\cdots y(n-i_k) \\ \vdots \\ y^k(n-M_k+1) \end{pmatrix}, \quad k=1,\cdots,q. \tag{3.5}$$

The input and output relation of the training filter $\tilde{C}(n)$ is given by

$$\begin{aligned} \tilde{y}(n) &= \boldsymbol{c}^T(n)\boldsymbol{z}(n) = \sum_{k=1}^{p} \boldsymbol{c}_k^T(n)\boldsymbol{z}_k(n) \\ &= \sum_{k=1}^{p} \left(\sum_{i_1=0}^{\widehat{M}_k-1} \cdots \sum_{i_k=0}^{\widehat{M}_k-1} c_k(i_1,\cdots,i_k;n) z(n-i_1)\cdots z(n-i_k) \right) \end{aligned} \tag{3.6}$$

where $\widehat{M}_k$ is the memory length corresponding to the kth order kernel. The total kernel vector $\boldsymbol{c}(n)$ is defined as

$$\boldsymbol{c}(n) = \begin{pmatrix} \boldsymbol{c}_1^T(n) & \cdots & \boldsymbol{c}_p^T(n) \end{pmatrix}^T, \tag{3.7}$$

$$\boldsymbol{c}_k(n) = \begin{pmatrix} \boldsymbol{c}_k(0,\cdots,0;n) \\ \vdots \\ \boldsymbol{c}_k(i_1,\cdots,i_k;n) \\ \vdots \\ \boldsymbol{c}_k(\widehat{M}_k-1,\cdots,\widehat{M}_k-1;n) \end{pmatrix}, \quad k=1,\cdots,p, \tag{3.8}$$

and the input vector $\boldsymbol{z}(n)$ to the training filter is defined as

$$\boldsymbol{z}(n) = \begin{pmatrix} \boldsymbol{z}_1^T(n) & \cdots & \boldsymbol{z}_p^T(n) \end{pmatrix}^T, \tag{3.9}$$

$$\boldsymbol{z}_k(n) = \begin{pmatrix} z^k(n) \\ \vdots \\ z(n-i_1)\cdots z(n-i_k) \\ \vdots \\ z^k(n-\widehat{M}_k+1) \end{pmatrix}, \quad k=1,\cdots,p. \tag{3.10}$$

The predistorter is a copy of the training filter, hence the relation between the input and output of the predistorter $C(n)$ is given by

$$\begin{aligned} y(n) &= \boldsymbol{c}^T(n)\boldsymbol{x}(n) = \sum_{k=1}^{p} \boldsymbol{c}_k^T(n)\boldsymbol{x}_k(n) \\ &= \sum_{k=1}^{p} \left(\sum_{i_1=0}^{\widehat{M}_k-1} \cdots \sum_{i_k=0}^{\widehat{M}_k-1} c_k(i_1, \cdots, i_k; n) x(n-i_1) \cdots x(n-i_k) \right) \end{aligned} \tag{3.11}$$

where the kernel vector $\boldsymbol{c}(n)$ is defined as in (3.7) and (3.8). The input vector $\boldsymbol{x}(n)$ to the predistorter is defined as

$$\boldsymbol{x}(n) = \begin{pmatrix} \boldsymbol{x}_1^T(n) & \cdots & \boldsymbol{x}_p^T(n) \end{pmatrix}^T, \tag{3.12}$$

$$\boldsymbol{x}_k(n) = \begin{pmatrix} x^k(n) \\ \vdots \\ x(n-i_1)\cdots x(n-i_k) \\ \vdots \\ x^k(n-\widehat{M}_k+1) \end{pmatrix}, \quad k = 1, \cdots, p. \tag{3.13}$$

Note that the predistorter can copy the coefficients from the training filter in each sample instant n, or copy the coefficients after the training filter has converged, which is usually preferred in real implementation in order to simplify the circuit design.

Let us define the error signal $e(n)$ as

$$e(n) = y(n) - \tilde{y}(n) \tag{3.14}$$

which can be written as (*cf.* (3.6) and (3.11))

$$e(n) = \boldsymbol{c}^T(n)(\boldsymbol{x}(n) - \boldsymbol{z}(n)). \tag{3.15}$$

Therefore, if the Volterra models satisfy the conditions [27]

$$\begin{aligned} \boldsymbol{x}(n) \neq \boldsymbol{z}(n) &\longrightarrow y(n) \neq \tilde{y}(n) \\ \boldsymbol{x}(n) = \boldsymbol{z}(n) &\longrightarrow y(n) = \tilde{y}(n) \end{aligned} \tag{3.16}$$

the error signal $e(n)$ approaches zero, $\boldsymbol{z}(n)$ approaches $\boldsymbol{x}(n)$, and hence the overall output of the system $z(n)$ (input to the training filter $\tilde{C}$) approaches the total system input $x(n)$ (input to the predistorter C) since the outputs of the two Volterra models, *i.e.*, $y(n)$ and $\tilde{y}(n)$, approach each other.

When the kernel vector $\boldsymbol{c}(n)$ has been found and it is believed that the nonlinear physical system characteristics are not changing, the training branch is shut down. The training branch can be reconnected in case of significant change in the characteristics of the nonlinear physical system and hence high nonlinear distortion would appear at the system output. On the other hand, in case of time-varying nonlinear systems, the training branch should always stay connected.

The RLS Algorithm

The kernel vector $\boldsymbol{c}(n)$ can be estimated as done in [27] using the RLS algorithm [57, 63]. The RLS algorithm with exponential forgetting minimizes the cost function $\xi(n)$ given by

$$\xi(n) = \sum_{i=1}^{n} \lambda^{n-i} e^2(i) \tag{3.17}$$

where $\lambda \leq 1$ is the forgetting factor and $e(i)$ is given as in (3.15). The smaller the value of λ, the quicker the information in previous data will be forgotten. Therefor, the choice of λ controls the ability of the algorithm to track time-varying parameters. The RLS algorithm takes the following form [57, 63]:

$$\begin{aligned} e(n) &= \boldsymbol{c}^T(n)(\boldsymbol{x}(n) - \boldsymbol{z}(n)) \\ \boldsymbol{k}(n) &= (\lambda + \boldsymbol{z}^T(n)\boldsymbol{P}(n-1)\boldsymbol{z}(n))^{-1}\boldsymbol{P}(n-1)\boldsymbol{z}(n) \\ \boldsymbol{P}(n) &= \lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{z}^T(n)\boldsymbol{P}(n-1) \\ \boldsymbol{c}(n+1) &= \boldsymbol{c}(n) + \boldsymbol{k}(n)e(n) \end{aligned} \tag{3.18}$$

The most common choice for the initial condition of $\boldsymbol{P}(n)$ is $\boldsymbol{P}(0) = \rho\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and ρ is a constant reflects our trust in the initial kernel vector $\boldsymbol{c}(0)$.

The KF algorithm

The KF algorithm [57,63] is a well studied algorithm that provides the optimal (mean square) estimate of the system state vector and also has the ability to track time-varying parameters. The KF is usually presented for state-space equations whose matrices may be time varying.

In order to construct the state space model for the ILA-I approach, the kernel vector $\boldsymbol{c}$ is modeled as a random walk or a drift. Hence, the training filter has the following state-space model

$$\begin{aligned} \boldsymbol{c}(n+1) &= \boldsymbol{c}(n) + \boldsymbol{v}(n) \\ \tilde{y}(n) &= \boldsymbol{c}^T(n)\boldsymbol{z}(n) \end{aligned} \tag{3.19}$$

where $\mathrm{E}\{\boldsymbol{v}(n)\boldsymbol{v}^T(m)\} = \boldsymbol{R}_1\delta_{n,m}$ and the covariance matrix $\boldsymbol{R}_1$ describes how fast different components of $\boldsymbol{c}$ are expected to vary.

Applying Kalman filter to the state-space model (3.19) gives the following recursive algorithm [57, 63]:

$$\begin{aligned} e(n) &= \boldsymbol{c}^T(n)(\boldsymbol{x}(n) - \boldsymbol{z}(n)) \\ \boldsymbol{k}(n) &= (1 + \boldsymbol{z}^T(n)\boldsymbol{P}(n-1)\boldsymbol{z}(n))^{-1}\boldsymbol{P}(n-1)\boldsymbol{z}(n) \\ \boldsymbol{P}(n) &= \boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{z}^T(n)\boldsymbol{P}(n-1) + \boldsymbol{R}_1 \\ \boldsymbol{c}(n+1) &= \boldsymbol{c}(n) + \boldsymbol{k}(n)e(n) \end{aligned} \tag{3.20}$$

The design variable $\boldsymbol{R}_1$ plays a similar role as the forgetting factor λ in the RLS algorithm (3.18). In case of tracking time variations of the Volterra kernels, λ should be small or $\boldsymbol{R}_1$ should be large. On the other hand, for small variances of time-invariant kernels, λ should be close to 1 or $\boldsymbol{R}_1$ close to zero.

The RPEM algorithm

The RPEM algorithm [57, 63] is derived by the minimization of the cost function

$$V(\boldsymbol{c}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n)\right] \tag{3.21}$$

where $e(n, \boldsymbol{c})$ is the prediction error defined as

$$e(n) = y(n) - \tilde{y}(n). \tag{3.22}$$

The formulation of the RPEM algorithm requires the negative gradient of $e(n)$ w.r.t. $\boldsymbol{c}(n)$. Thus using (3.6) and (3.11), we have

$$\boldsymbol{\varphi}^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{c}(n)} e(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)} \tilde{y}(n) = \boldsymbol{z}^T(n). \tag{3.23}$$

Hence, the RPEM algorithm [57, 63] follows as

$$\begin{aligned}
&e(n) = y(n) - \boldsymbol{c}^T(n)\boldsymbol{z}(n) \\
&\lambda(n) = \lambda_0\lambda(n-1) + 1 - \lambda_0 \\
&s(n) = \boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\varphi}(n) + \lambda(n) \\
&\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1) - \boldsymbol{P}(n-1)\boldsymbol{\varphi}(n)s^{-1}(n)\boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\big)/\lambda(n) \\
&\boldsymbol{c}(n+1) = \boldsymbol{c}(n) + \boldsymbol{P}(n)\boldsymbol{\varphi}(n)e(n).
\end{aligned} \tag{3.24}$$

Here $\lambda(n)$ is a forgetting factor grows exponentially to 1 as $n \to \infty$ where the rate λ_0 and the initial value $\lambda(0)$ and $\boldsymbol{P}(0)$ are design variables.

Remark 3.1: In case of predistortion of time-varying nonlinear systems, the RPEM algorithm of (3.24) can be modified to:

$$\begin{aligned}
&e(n) = y(n) - \boldsymbol{c}^T(n)\boldsymbol{z}(n) \\
&s(n) = \boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\varphi}(n) + r_2 \\
&\boldsymbol{P}(n) = \boldsymbol{P}(n-1) - \boldsymbol{P}(n-1)\boldsymbol{\varphi}(n)s^{-1}(n)\boldsymbol{\varphi}^T(n)\boldsymbol{P}(n-1) + \boldsymbol{R}_3 \\
&\boldsymbol{c}(n+1) = \boldsymbol{c}(n) + \boldsymbol{P}(n)\boldsymbol{\varphi}(n)e(n)
\end{aligned} \tag{3.25}$$

where r_2 and $\boldsymbol{R}_3$ are the gain design variables, see [57]. This modification transforms the algorithm into a special case of the KF algorithm.

Remark 3.2: Since our system model is linear in parameters, the gradient calculation in (3.23) leads to $\boldsymbol{\varphi}(n) = \boldsymbol{z}(n)$. Hence, the two algorithms (3.20) and (3.25) are expected to perform similarly. An identical performance of these two algorithms is obtained in case the design variables are chosen as $r_2 = 1$ and $\boldsymbol{R}_3 = \boldsymbol{R}_1$. Also, the RLS algorithm in (3.19) can be obtained exactly from the RPEM algorithm in (3.25) by setting $\lambda_0 = 1$ and $\lambda(0) = \lambda$.

Simulation Study

In this section, a comparison study of the RLS, KF and RPEM algorithms is given using computer simulations. The nonlinear physical system H is a known second-order Volterra system and the input-output relation of H was chosen identical to Section 2.2.4 as

$$z(n) = H\left[y(n)\right] = \boldsymbol{h}_1^T\boldsymbol{y}_1(n) + \boldsymbol{h}_2^T\boldsymbol{y}_2(n). \tag{3.26}$$

	λ_0	λ	$\boldsymbol{P}(0)$	$\boldsymbol{R}_1$
RLS	1	-	$10^3\boldsymbol{I}$	-
KF	-	-	$10^3\boldsymbol{I}$	$10^{-5}\boldsymbol{I}$
RPEM	0.99	0.95	$10^3\boldsymbol{I}$	-

Table 3.1 Initialization of the RLS, KF and RPEM algorithms.

where the first-order kernel vector $\boldsymbol{h}_1$ was

$$\boldsymbol{h}_1 = \begin{pmatrix} 0.5625 & 0.4810 & 0.1124 & -0.1669 \end{pmatrix} \tag{3.27}$$

and the second-order kernel vector $\boldsymbol{h}_2$ was

$$\boldsymbol{h}_2 = (0.0175 \quad 0 \quad 0 \quad 0 \quad 0 \quad -0.0088 \quad 0 \quad -0.0088 \quad 0)\,. \tag{3.28}$$

The predistorter C and the training filter $\tilde{C}$ are also assumed to be second-order Volterra systems. This means that $q = p = 2$. Also, the number of memory in the training filter and predistorter was chosen as $\widehat{M}_1 = \widehat{M}_2 = 4$.

The number of independent experiments was 100 and in each experiment, the input signal to the predistorter was chosen as a random signal with uniform distribution over $(-1, 1)$ with data length 2×10^3. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{4}$. The reference signal $r(n)$ was chosen to be equal to the input signal $x(n)$ in additive white Gaussian noise (AWGN) such that a signal to noise ratio (SNR) of 40 dB was achieved.

The MSD comparison between the RLS, KF and RPEM algorithms is given in Fig. 3.2. The algorithms were initialized as in Table 3.1. The MSD of the nonlinear physical system without predistorter was about -16 dB. On average, these algorithms converge after about 300 samples and the achieved values of E_D are about -40 dB.

Figure 3.3 shows the mean PSDs of the output signals of the nonlinear physical system without and with predistorter after 2×10^3 samples. From this figure, we can see that with predistorter, there is a significant spectral regrowth reduction in the normalized frequency band $(0.30\pi, 0.55\pi)$. The RPEM algorithm has a little better performance than the RLS and KF algorithms. Therefore, from this simulation, we can say that for predistortion of time-invariant Volterra systems, the RLS, KF and RPEM algorithm can achieve similar performance with similar converge speed and computational complexity. Compared to the simulation results in Figs. 2.4 and 2.16, these algorithms can achieve similar performance as the NFxRLS, NFxPEM algorithms and the SMM method (up to 30 dB reduction).

3.2.2. The ILA-II approach

The ILA-II approach [9] for predistortion of Volterra systems is shown in Fig. 3.4. The nonlinear physical system H is a qth-order Volterra system with M memories, and H can be be divided into two subsystems, one is the purely linear subsystem H_L and the other is the purely nonlinear subsystem H_N. Hence, the output of this system $z(n)$ is given by

$$\begin{aligned} z(n) &= H[y(n)] = H_L[y(n)] + H_N[y(n)] \\ &= \boldsymbol{h}^T\boldsymbol{y}(n) = \boldsymbol{h}_L^T\boldsymbol{y}_L(n) + \boldsymbol{h}_N^T\boldsymbol{y}_N(n) \end{aligned} \tag{3.29}$$

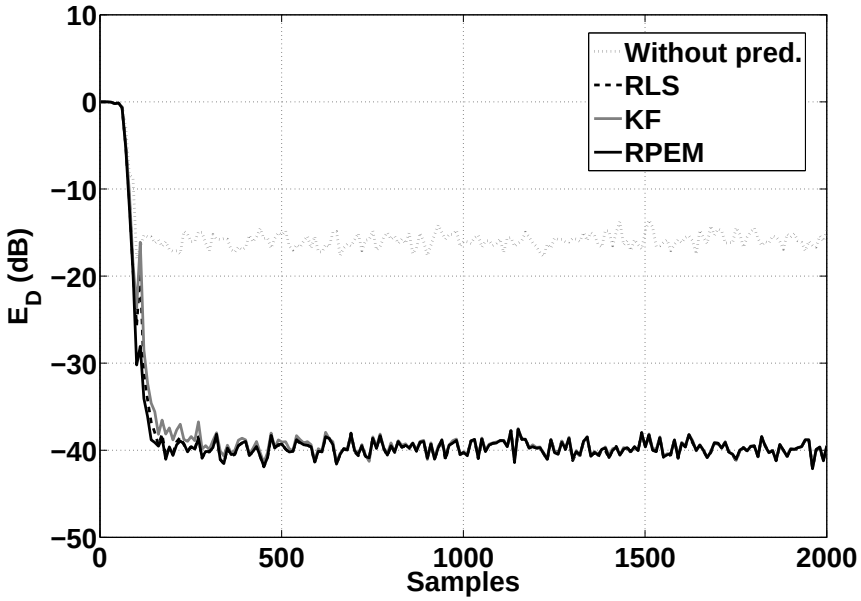

Figure 3.2 MSD E_D for different adaptation algorithms.

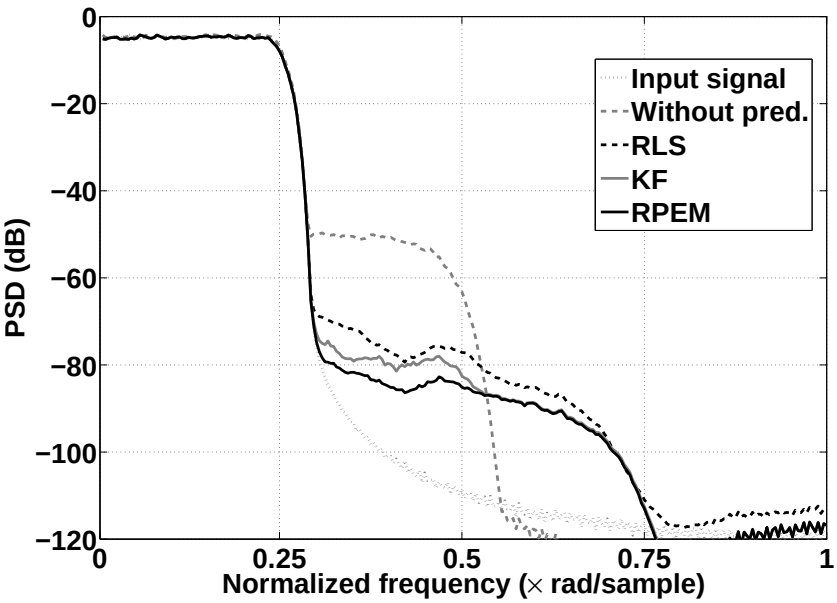

Figure 3.3 Mean PSDs for different adaptation algorithms.

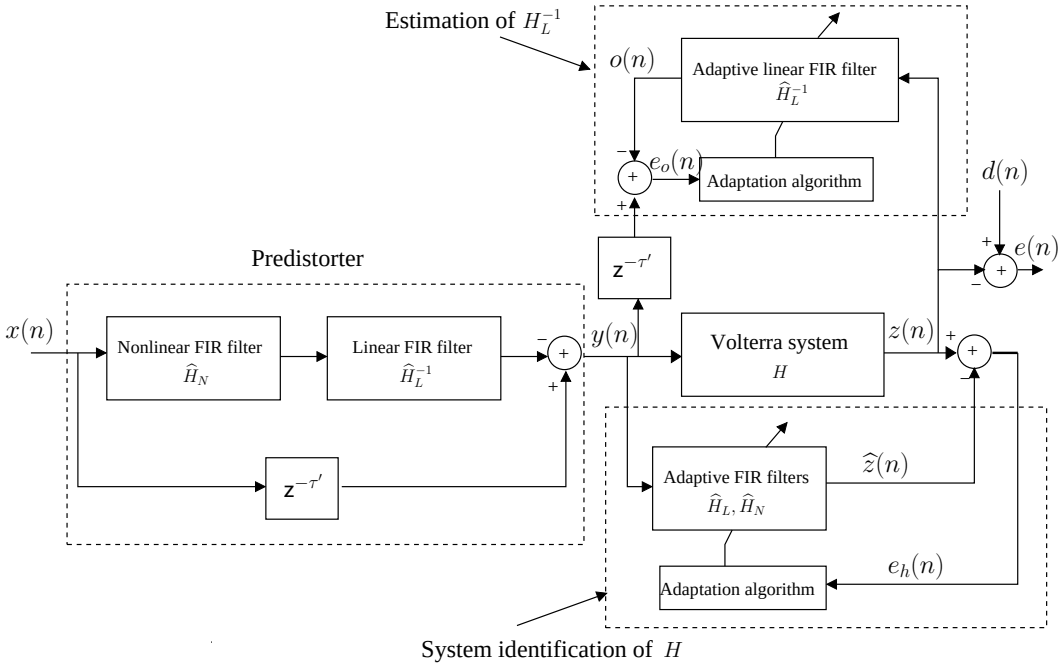

Figure 3.4 The ILA-II approach for predistortion of Volterra systems.

where the kernel vector $\boldsymbol{h}$ is defined as

$$\boldsymbol{h} = \begin{pmatrix} \boldsymbol{h}_L^T & \boldsymbol{h}_N^T \end{pmatrix}^T, \tag{3.30}$$

$$\boldsymbol{h}_L = \begin{pmatrix} h_0 & h_1 & \cdots & h_{M-1} \end{pmatrix}^T, \tag{3.31}$$

$$\boldsymbol{h}_N = \begin{pmatrix} h_{0,0} & h_{0,1} & \cdots & h_{M-1,\cdots,M-1} \end{pmatrix}^T, \tag{3.32}$$

and the corresponding input vector $\boldsymbol{y}(n)$ is given by

$$\boldsymbol{y}(n) = \begin{pmatrix} \boldsymbol{y}_L^T(n) & \boldsymbol{y}_N^T(n) \end{pmatrix}^T, \tag{3.33}$$

$$\boldsymbol{y}_L(n) = \begin{pmatrix} y(n) \\ y(n-1) \\ \vdots \\ y(n-M+1) \end{pmatrix}, \tag{3.34}$$

$$\boldsymbol{y}_N(n) = \begin{pmatrix} y^2(n) \\ y(n)y(n-1) \\ \vdots \\ y^q(n-M+1) \end{pmatrix}. \tag{3.35}$$

Another Volterra system $\widehat{H}$ with pth-order and $\widehat{M}$ memories is used to identify H. Similarly, $\widehat{\boldsymbol{h}}$ can also be divided into the purely linear subsystem $\widehat{H}_L$ and the purely nonlinear subsystem $\widehat{H}_N$. The output signal $\widehat{z}(n)$ can be written as

$$\begin{aligned} \widehat{z}(n) &= \widehat{H}[y(n)] = \widehat{H}_L[y(n)] + \widehat{H}_N[y(n)] \\ &= \widehat{\boldsymbol{h}}^T(n)\widehat{\boldsymbol{y}}(n) = \widehat{\boldsymbol{h}}_L^T(n)\widehat{\boldsymbol{y}}_L(n) + \widehat{\boldsymbol{h}}_N^T(n)\widehat{\boldsymbol{y}}_N(n) \end{aligned} \tag{3.36}$$

where the kernel vector $\widehat{\boldsymbol{h}}(n)$ is defined as

$$\widehat{\boldsymbol{h}}(n) = \begin{pmatrix} \widehat{\boldsymbol{h}}_L^T(n) & \widehat{\boldsymbol{h}}_N^T(n) \end{pmatrix}^T, \tag{3.37}$$

$$\widehat{\boldsymbol{h}}_L(n) = \begin{pmatrix} \widehat{h}_0(n) & \widehat{h}_1(n) & \cdots & \widehat{h}_{\widehat{M}-1}(n) \end{pmatrix}^T, \tag{3.38}$$

$$\widehat{\boldsymbol{h}}_N(n) = \begin{pmatrix} \widehat{h}_{0,0}(n) & \widehat{h}_{0,1}(n) & \cdots & \widehat{h}_{\widehat{M}-1,\cdots,\widehat{M}-1}(n) \end{pmatrix}^T, \tag{3.39}$$

and the corresponding input vector $\widehat{\boldsymbol{y}}(n)$ is given by

$$\widehat{\boldsymbol{y}}(n) = \begin{pmatrix} \widehat{\boldsymbol{y}}_L^T(n) & \widehat{\boldsymbol{y}}_N^T(n) \end{pmatrix}^T, \tag{3.40}$$

$$\widehat{\boldsymbol{y}}_L(n) = \begin{pmatrix} y(n) \\ y(n-1) \\ \vdots \\ y(n-\widehat{M}+1) \end{pmatrix}, \tag{3.41}$$

$$\widehat{\boldsymbol{y}}_N(n) = \begin{pmatrix} y^2(n) \\ y(n)y(n-1) \\ \vdots \\ y^p(n-\widehat{M}+1) \end{pmatrix}. \tag{3.42}$$

Remark 3.3: The choice of an appropriate model order p with $\widehat{M}$ memories is needed for accurate identification of the system H. The ideal values are $p = q$ and $\widehat{M} = M$. Choice of higher or lower model orders leads to over or under-parameterization, respectively. See the parsimony principle in [63].

The inverse of the linear FIR filter, H_L^{-1}, is also required to design the predistorter and it is estimated using an adaptive linear FIR filter $\widehat{H}_L^{-1}$ with K memories. The output signal of this filter can be written as

$$o(n) = \widehat{H}_L^{-1}[z(n)] = \left[\widehat{\boldsymbol{h}}_L^{-1}(n)\right]^T \boldsymbol{z}(n) \tag{3.43}$$

where

$$\widehat{\boldsymbol{h}}_L^{-1}(n) = \begin{pmatrix} \widehat{h}_0^{-1}(n) & \cdots & \widehat{h}_{K-1}^{-1}(n) \end{pmatrix}^T \tag{3.44}$$

$$\boldsymbol{z}(n) = \begin{pmatrix} z(n) & \cdots & z(n-K+1) \end{pmatrix}^T. \tag{3.45}$$

Note that the -1 in $\widehat{h}_j^{-1}(n)$ does not represent the inverse of the coefficient $\widehat{h}_j(n)$, and it just denotes that $\widehat{h}_j^{-1}(n)$ is the element of the vector $\widehat{\boldsymbol{h}}_L^{-1}(n)$.

As suggested in [9], the predistorter is constructed by the copies of $\widehat{H}_N$ and $\widehat{H}_L^{-1}$ after the adaptive filters have been running for a sufficient time T and get close to convergence.

Remark 3.4: During the time T, the signal $y(n)$ is equal to the input $x(n)$. Once the predistorter is constructed, the signal $y(n)$ becomes

$$y(n) = x(n-\tau') - \widehat{H}_L^{-1}\left[\widehat{H}_N[x(n)]\right]. \tag{3.46}$$

Now, from (3.29) and (3.46), we have

$$\begin{aligned} z(n) &= H_L[y(n)] + H_N[y(n)] \\ &= H_L\left[x(n-\tau') - \widehat{H}_L^{-1}\left[\widehat{H}_N[x(n)]\right]\right] \\ &\quad + H_N\left[x(n-\tau') - \widehat{H}_L^{-1}\left[\widehat{H}_N[x(n)]\right]\right]. \end{aligned} \tag{3.47}$$

Assuming that we have obtained accurate estimates for H_N and H_L^{-1}, we have

$$\widehat{H} = H \tag{3.48}$$

$$\widehat{H}_L^{-1} = H_L^{-1} \tag{3.49}$$

where $H_L^{-1}H_L = \mathsf{z}^{-\tau'}$ and $\mathsf{z}^{-\tau'}x(n) = x(n-\tau')$ [9]. Therefore, $z(n)$ can be written as

$$\begin{aligned} z(n) &= H_L\left[x(n-\tau') - H_L^{-1}\left[H_N[x(n)]\right]\right] \\ &\quad + H_N\left[x(n-\tau') - H_L^{-1}\left[H_N[x(n)]\right]\right]. \end{aligned} \tag{3.50}$$

For weakly nonlinear systems, we have [9]

$$\begin{aligned} &|H_L[x(n)]| \gg |H_N[x(n)]|, \\ &|x(n-\tau')| \gg |H_L^{-1}\left[H_N[x(n)]\right]|. \end{aligned} \tag{3.51}$$

Substituting (3.51) in (3.50) gives

$$\begin{aligned} z(n) &\approx H_L\left[x(n-\tau')\right] - H_L\left[H_L^{-1}\left[H_N[x(n)]\right]\right] + H_N\left[x(n-\tau')\right] \\ &= H_L\left[x(n-\tau')\right] - H_N\left[x(n-\tau')\right] + H_N\left[x(n-\tau')\right] \\ &= H_L[x(n-\tau')] = d(n) \end{aligned} \tag{3.52}$$

where $d(n)$ is the desired signal.

Estimation of H_N and H_L^{-1} Using The LMS Algorithm

The parameter vector $\widehat{\boldsymbol{h}}$ is estimated in [9] using the LMS algorithm as follows:

$$\widehat{\boldsymbol{h}}(n+1) = \widehat{\boldsymbol{h}}(n) + \mu e_h(n)\widehat{\boldsymbol{y}}(n) \tag{3.53}$$

where μ is the step size and $e_h(n)$ is the error signal defined as

$$e_h(n) = z(n) - \widehat{z}(n). \tag{3.54}$$

The coefficients of $\widehat{H}_L^{-1}$ are also estimated in [9] using the LMS algorithm as

$$\widehat{\boldsymbol{h}}_L^{-1}(n+1) = \widehat{\boldsymbol{h}}_L^{-1}(n) + \mu e_o(n)\boldsymbol{z}(n) \tag{3.55}$$

where $e_o(n)$ is the error signal defined as

$$e_o(n) = y(n-\tau') - o(n). \tag{3.56}$$

As it is clear from (3.52), the efficiency of the suggested linearization scheme in Fig. 3.2 highly depends on the accuracy of the estimated FIR filters H_N and H_L^{-1}. However, due to the fact that the LMS algorithm provides inaccurate estimates for the nonlinear FIR filters [33], the performance of the linearization scheme is degraded.

Estimation of H_N and H_L^{-1} Using The RPEM Algorithm

In order to estimate the parameter vector $\widehat{\boldsymbol{h}}$ and hence the vector $\widehat{\boldsymbol{h}}_N$ to construct the nonlinear FIR filter $\widehat{H}_N$, the RPEM algorithm is derived by minimizing the cost function [57]

$$V(\widehat{\boldsymbol{h}}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e_h^2(n)\right] \tag{3.57}$$

where $\mathrm{E}\{.\}$ denotes the expectation. Here, $e_h(n)$ is the prediction error defined as

$$e_h(n) = z(n) - \widehat{z}(n). \tag{3.58}$$

The formulation of the RPEM algorithm requires the negative gradient of $e_h(n)$ w.r.t. $\widehat{\boldsymbol{h}}(n)$. Using (3.36), it can be written as

$$\boldsymbol{\varphi}_h^T(n) = -\boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}(n)} e_h(n) = \boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}(n)} \widehat{z}(n) = \widehat{\boldsymbol{y}}^T(n) \tag{3.59}$$

where $\widehat{\boldsymbol{y}}(n)$ is defined by (3.40)-(3.42). Hence, the RPEM algorithm follows as (*cf.* [56, 57])

$$\begin{aligned}
&e_h(n) = z(n) - \widehat{z}(n)\\
&\lambda(n) = \lambda_0\lambda(n-1) + 1 - \lambda_0\\
&s_h(n) = \boldsymbol{\varphi}_h^T(n)\boldsymbol{P}_h(n-1)\boldsymbol{\varphi}_h(n) + \lambda(n)\\
&\boldsymbol{P}_h(n) = \big(\boldsymbol{P}_h(n-1) - \boldsymbol{P}_h(n-1)\boldsymbol{\varphi}_h(n)s_h^{-1}(n)\boldsymbol{\varphi}_h^T(n)\boldsymbol{P}_h(n-1)\big)/\lambda(n)\\
&\widehat{\boldsymbol{h}}(n+1) = \widehat{\boldsymbol{h}}(n) + \boldsymbol{P}_h(n)\boldsymbol{\varphi}_h(n)e_h(n).
\end{aligned} \tag{3.60}$$

Here λ_0, the initial value $\lambda(0)$ and $\boldsymbol{P}_h(0)$ are design variables.

Also, the RPEM algorithm can be derived similarly to estimate the coefficients of H_L^{-1} by minimizing

$$V(\widehat{\boldsymbol{h}}_L^{-1}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e_o^2(n)\right] \tag{3.61}$$

where $e_o(n)$ the prediction error defined as

$$e_o(n) = y(n-\tau') - o(n). \tag{3.62}$$

The negative gradient of $e_o(n)$ w.r.t. $\widehat{\boldsymbol{h}}_L^{-1}(n)$ is (*cf.* (3.43))

$$\boldsymbol{\varphi}_l^T(n) = -\boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}_L^{-1}(n)} e_o(n) = \boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}_L^{-1}(n)} o(n) = \boldsymbol{z}^T(n). \tag{3.63}$$

Hence, similarly to (3.60), the RPEM algorithm follows as:

$$\begin{aligned}
&e_o(n) = y(n-\tau') - o(n)\\
&\lambda(n) = \lambda_0\lambda(n-1) + 1 - \lambda_0\\
&s_l(n = \boldsymbol{\varphi}_l^T(n)\boldsymbol{P}_l(n-1)\boldsymbol{\varphi}_l(n) + \lambda(n)\\
&\boldsymbol{P}_l(n) = \big(\boldsymbol{P}_l(n-1) - \boldsymbol{P}_l(n-1)\boldsymbol{\varphi}_l(n)s_l^{-1}(n)\boldsymbol{\varphi}_l^T(n)\boldsymbol{P}_l(n-1)\big)/\lambda(n)\\
&\widehat{\boldsymbol{h}}_L^{-1}(n+1) = \widehat{\boldsymbol{h}}_L^{-1}(n) + \boldsymbol{P}_l(n)\boldsymbol{\varphi}_l(n)e_o(n).
\end{aligned} \tag{3.64}$$

Simulation Study

In this section, a simulation study for the performance of the predistortion using the ILA-II approach in case of using the LMS algorithm or the RPEM algorithm is given.

The nonlinear physical system H was a known second-order Volterra system and the first-order kernel vector $\boldsymbol{h}_1$ and the second-order kernel vector $\boldsymbol{h}_2$ of H were defined identical to Section 2.2.4. The adaptive filter $\widehat{H}$ was chosen as a 2nd-order Volterra system with memory length $\widehat{M} = 4$. $\widehat{H}_L^{-1}$ was chosen as an adaptive linear FIR filter with memory length $K = 8$, which can achieve the best performance based on large number of experiments for different values.

The input signal to the predistorter $x(n)$ was chosen to be a random signal with uniform distribution over $(-1, 1)$ and data length of 2×10^4. The system should start to copy the coefficients of $\widehat{H}_N$ and $\widehat{H}_L^{-1}$ into the predistorter after they have converged. The bandwidth of the input signal $x(n)$ was limited by a low-pass filter. Before the copy process, the predistorter is just a through connection. Thus, $y(n) = x(n)$ and $z(n)$ is the output of the 2nd-order nonlinear system H. In this case, the normalized cut-off frequency of the low-pass filter filter is chosen as $\frac{\pi}{2}$ thus the nonlinear physical system is fully triggered for the system identification purpose. When the copy process starts, the predistorter is connected to the nonlinear physical system and $z(n)$ becomes the output of two cascaded 2nd-order nonlinear systems. Therefore, the cut-off frequency of the low-pass filter is now chosen as $\frac{\pi}{4}$ in order to prevent aliasing [13]. In another word, the sampling rate of the whole system should be the sampling rate of the system output, which is different before and after the copy process.

As a measure of the performance of the system identification process, the normalized mean square estimation error of the parameter vector $\boldsymbol{h}$ is defined as

$$E_I = 10 \log_{10} \left(\frac{\widehat{\mathrm{E}}\{\|\widehat{\boldsymbol{h}}(n) - \boldsymbol{h}\|_2\}}{\widehat{\mathrm{E}}\{\|\boldsymbol{h}\|_2\}} \right) \tag{3.65}$$

where $\widehat{\mathrm{E}}\{.\}$ is the mean obtained using 100 independent experiments. Moreover, from (3.52) we can see that the desired signal that can be achieved by the ILA-II approach is the input signal filtered by the linear subsystem H_L, which means that the ILA-II approach can only reduce the distortion caused by H_N. Therefore, we define the normalized Mean Square Nonlinear Distortion (MSND) as

$$E_{ND} = 10 \log_{10} \left(\frac{\widehat{\mathrm{E}}\{e^2(n)\}}{\widehat{\mathrm{E}}\{d^2(n)\}} \right) \tag{3.66}$$

where $d(n)$ is given in (3.52) and $e(n)$ is defined as

$$e(n) = d(n) - z(n). \tag{3.67}$$

The comparison of E_I between the LMS and RPEM algorithms is given in Fig. 3.5. The measurement noise was AWGN with SNR=40 dB. The two algorithms were initialized with $\mu = 0.1$, $\lambda_0 = 0.99$, $\lambda(0) = 0.95$, and $\boldsymbol{P}_h(0) = \boldsymbol{P}_l(0) = \mathbf{I}$. After convergence, the RPEM and LMS algorithms achieve about -46 dB and -40 dB, respectively. Therefore, the RPEM algorithm provides much better estimates than the LMS algorithm.

Also, the comparison with respect to the MSND is given in Fig. 3.6. The system started to copy the coefficients of $\widehat{H}_N$ and $\widehat{H}_L^{-1}$ into the predistorter after 1.6×10^4 input samples.

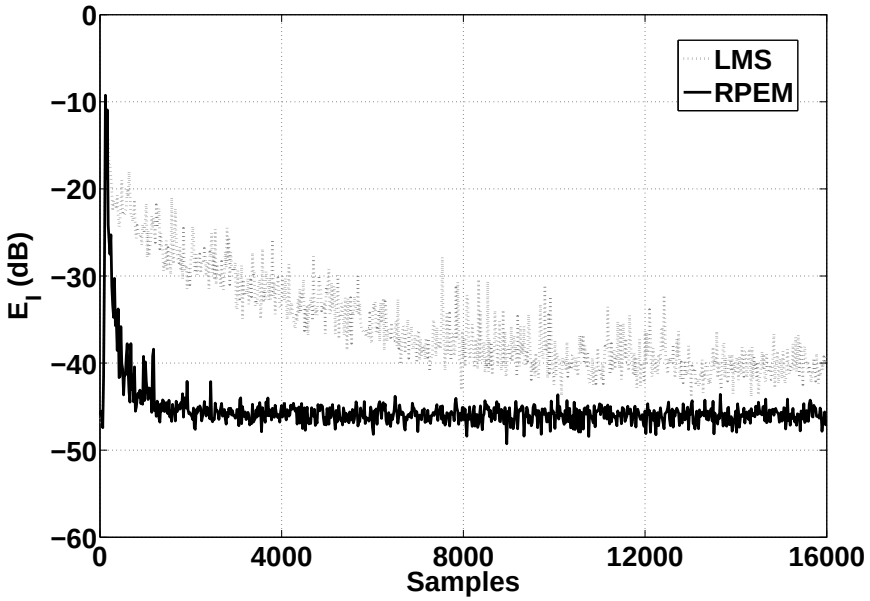

Figure 3.5 E_I comparison between the LMS and RPEM algorithms before copying.

The E_{ND} of the nonlinear physical system without the predistorter was about -46 dB. As from the previous development of [9] and further elaborated in [70], the performance of the ILA-II approach is limited by the fact that the uncompensated residual with predistorter is of the order of the square of the original nonlinear component, *i.e.*, the ideal MSND value that the system with predistorter can achieve should be double of the MSND value of the system without predistorter. As it is shown in Fig. 3.6, the RPEM algorithm gives much lower nonlinear distortion than the LMS algorithm. On average, the RPEM and LMS algorithms achieve about -72 dB and -56 dB, respectively.

Figure 3.7 shows the mean PSDs of the output signals of the nonlinear physical system without and with the predistorter after 2×10^4 samples. From this figure, we can see that the RPEM algorithm reduces spectral regrowth more effectively as compared to the LMS algorithm. Compared to the simulation results in Figs. 2.4, 2.16 and 3.3, the ILA-II approach using the RPEM algorithm can achieve similar performance as the previous algorithms (up to 40 dB reduction).

3.3. Predistortion of Wiener Systems

In this section, the adaptation algorithm for predistortion of Wiener systems using the ILA-I approach is introduced. The RPEM algorithm is derived and simulation results are given to verify the effectiveness of the suggested the algorithm.

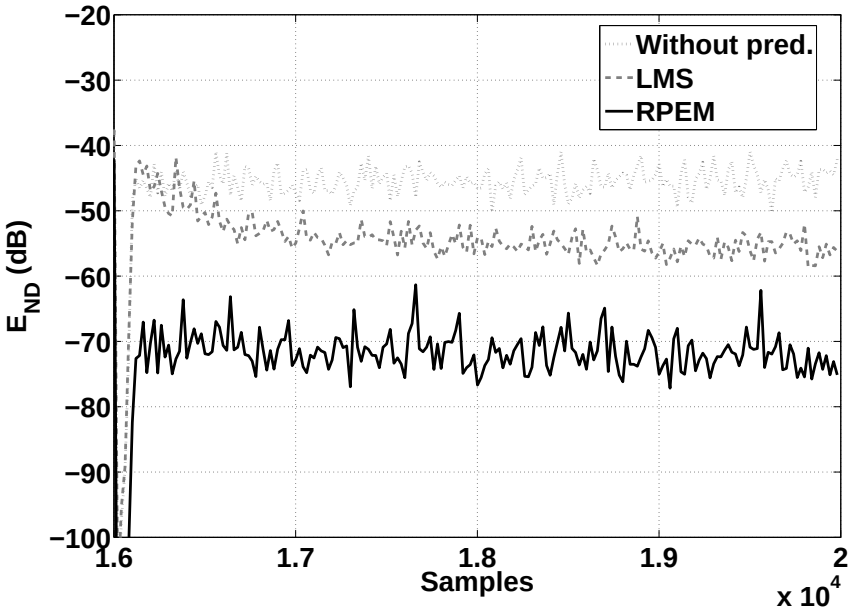

Figure 3.6 E_{ND} for different adaptation algorithms after copying.

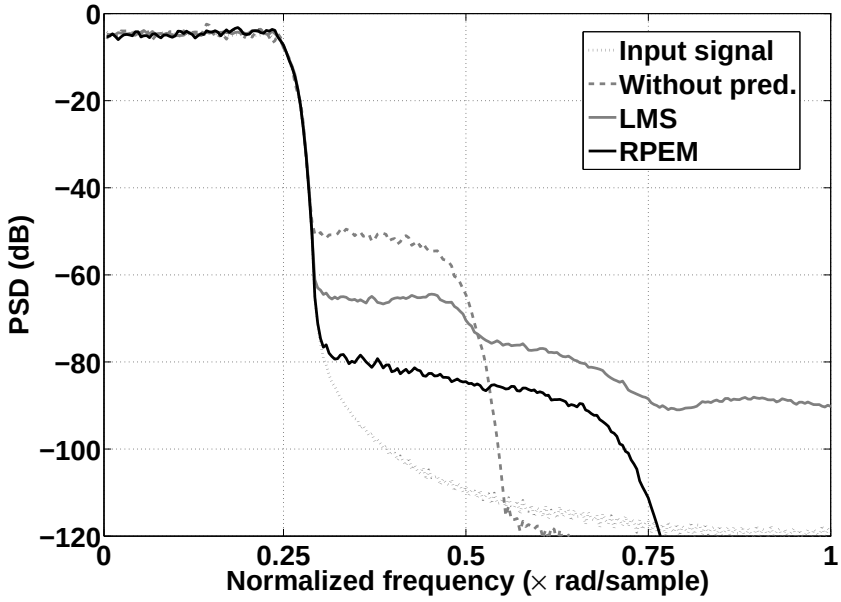

Figure 3.7 Mean PSDs for different adaptation algorithms.

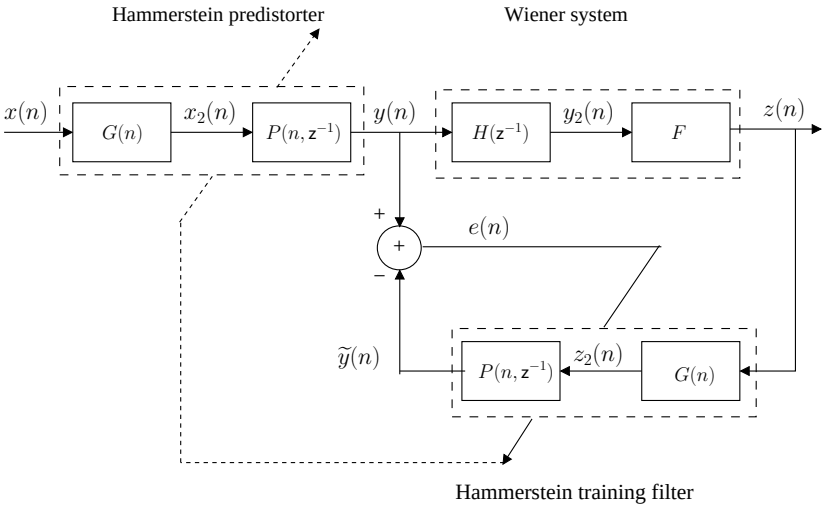

Figure 3.8 The ILA-I approach for predistortion of Wiener systems.

3.3.1. The ILA-I approach

The ILA-I approach for predistortion of Wiener systems is shown in Fig. 3.8. The output of the IIR Wiener system is

$$\begin{aligned} z(n) &= f_1 y_2(n) + f_2 y_2^2(n) + \cdots + f_{m_f} y_2^{m_f}(n) \\ &= \boldsymbol{f}^T \boldsymbol{y}_2(n) \end{aligned} \tag{3.68}$$

where $\boldsymbol{f}$ is the parameter vector of the nonlinear subsystem F defined as

$$\boldsymbol{f} = \begin{pmatrix} f_1 & f_2 & \cdots & f_{m_f} \end{pmatrix}^T \tag{3.69}$$

and the corresponding input vector $\boldsymbol{y}_2(n)$ is given by

$$\boldsymbol{y}_2(n) = \begin{pmatrix} y_2(n) & y_2^2(n) & \cdots & y_2^{m_f}(n) \end{pmatrix}^T. \tag{3.70}$$

The intermediate signal $y_2(n)$ is defined as

$$\begin{aligned} y_2(n) &= H(\mathsf{z}^{-1}) y(n) = \frac{B(\mathsf{z}^{-1})}{1 - A(\mathsf{z}^{-1})} y(n) \\ &= \sum_{m=0}^{m_b} b_m y(n-m) + \sum_{m=1}^{m_a} a_m y_2(n-m) \end{aligned} \tag{3.71}$$

where $H(\mathsf{z}^{-1}) = \frac{B(\mathsf{z}^{-1})}{1-A(\mathsf{z}^{-1})}$ is an IIR filter and the polynomials $A(\mathsf{z}^{-1})$ and $B(\mathsf{z}^{-1})$ are defined as

$$\begin{aligned} A(\mathsf{z}^{-1}) &= \sum_{m=1}^{m_a} a_m \mathsf{z}^{-m} \\ B(\mathsf{z}^{-1}) &= \sum_{m=0}^{m_b} b_m \mathsf{z}^{-m}. \end{aligned} \tag{3.72}$$

Here z^{-1} is the delay operator such that $\mathsf{z}^{-m}x(n) = x(n-m)$.

The training filter is modeled as a Hammerstein system and the output signal $\tilde{y}(n)$ is given by

$$\begin{aligned}\tilde{y}(n) &= P(n,\mathsf{z}^{-1})z_2(n) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}z_2(n) \\ &= \sum_{m=0}^{m_d} d_m(n)z_2(n-m) + \sum_{m=1}^{m_c} c_m(n)\tilde{y}(n-m)\end{aligned} \tag{3.73}$$

where $P(n,\mathsf{z}^{-1}) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}$ is an IIR filter and the polynomials $C(n,\mathsf{z}^{-1})$ and $D(n,\mathsf{z}^{-1})$ are defined as

$$\begin{aligned}C(n,\mathsf{z}^{-1}) &= \sum_{m=1}^{m_c} c_m(n)\mathsf{z}^{-m} \\ D(n,\mathsf{z}^{-1}) &= \sum_{m=0}^{m_d} d_m(n)\mathsf{z}^{-m}.\end{aligned} \tag{3.74}$$

The intermediate signal $z_2(n)$ is defined as

$$\begin{aligned}z_2(n) &= g_1(n)z(n) + g_2(n)z^2(n) + \cdots + g_{m_g}(n)z^{m_g}(n) \\ &= \boldsymbol{g}^T(n)\boldsymbol{z}(n)\end{aligned} \tag{3.75}$$

where $\boldsymbol{g}(n)$ is the parameter vector of the nonlinear subsystem $G(n)$ defined as

$$\boldsymbol{g}(n) = \begin{pmatrix} g_1(n) & g_2(n) & \cdots & g_{m_g}(n) \end{pmatrix}^T \tag{3.76}$$

and the corresponding input vector $\boldsymbol{z}(n)$ is given by

$$\boldsymbol{z}(n) = \begin{pmatrix} z(n) & z^2(n) & \cdots & z^{m_g}(n) \end{pmatrix}^T. \tag{3.77}$$

The predistorter is a copy of the training filter, hence the predistorter is also a Hammerstein model with output signal $y(n)$ given as

$$\begin{aligned}y(n) &= P(n,\mathsf{z}^{-1})x_2(n) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}x_2(n) \\ &= \sum_{m=0}^{m_d} d_m(n)x_2(n-m) + \sum_{m=1}^{m_c} c_m(n)y(n-m)\end{aligned} \tag{3.78}$$

where $P(n,\mathsf{z}^{-1})$ is the IIR filter with parameter vector defined as in (3.74). The intermediate signal $x_2(n)$ is defined as

$$\begin{aligned}x_2(n) &= g_1(n)x(n) + g_2(n)x^2(n) + \cdots + g_{m_g}(n)x^{m_g}(n) \\ &= \boldsymbol{g}^T(n)\boldsymbol{x}(n)\end{aligned} \tag{3.79}$$

where $\boldsymbol{g}(n)$ is the parameter vector defined as in (3.76) and the corresponding input vector $\boldsymbol{x}(n)$ is given by

$$\boldsymbol{x}(n) = \begin{pmatrix} x(n) & x^2(n) & \cdots & x^{m_g}(n) \end{pmatrix}^T. \tag{3.80}$$

Define the parameter vector $\boldsymbol{\theta}$ of the training filter as follows

$$\begin{aligned}\boldsymbol{\theta} &= \begin{pmatrix} \boldsymbol{\theta}_d^T & \boldsymbol{\theta}_c^T & \boldsymbol{\theta}_g^T \end{pmatrix}^T \\ \boldsymbol{\theta}_d &= \begin{pmatrix} d_0 & d_1 & \cdots & d_{m_d} \end{pmatrix}^T \\ \boldsymbol{\theta}_c &= \begin{pmatrix} c_1 & c_2 & \cdots & c_{m_c} \end{pmatrix}^T \\ \boldsymbol{\theta}_g &= \begin{pmatrix} g_1 & g_2 & \cdots & g_{m_g} \end{pmatrix}^T.\end{aligned} \tag{3.81}$$

The RPEM algorithm [57, 63] is derived by minimization of the cost function

$$V(\boldsymbol{\theta}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n)\right] \tag{3.82}$$

where $e(n)$ is the prediction error defined as

$$e(n) = y(n) - \tilde{y}(n). \tag{3.83}$$

The formulation of the RPEM algorithm requires the negative gradient of $e(n)$ with respect to $\boldsymbol{\theta}(n)$. The negative gradient $\boldsymbol{\varphi}(n)$ is defined as

$$\boldsymbol{\varphi}^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} e(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} \tilde{y}(n) = \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} \tilde{y}(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} \tilde{y}(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} \tilde{y}(n) \end{pmatrix}. \tag{3.84}$$

Differentiating both sides of (3.73) with respect to $d_k(n)$ and $c_k(n)$ gives

$$\begin{aligned}\frac{\partial \tilde{y}(n)}{\partial d_k(n)} &= z_2(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial \tilde{y}(n-m)}{\partial d_k(n)}, \quad k = 0, 1, \cdots, m_d \\ \frac{\partial \tilde{y}(n)}{\partial c_k(n)} &= \tilde{y}(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial \tilde{y}(n-m)}{\partial c_k(n)}, \quad k = 1, \cdots, m_c.\end{aligned} \tag{3.85}$$

Since the parameter vector $\boldsymbol{\theta}(n)$ is assumed to be changing slowly, we can write

$$\begin{aligned}\frac{\partial \tilde{y}(n-m)}{\partial d_k(n)} &\approx \frac{\partial \tilde{y}(n-m)}{\partial d_k(n-m)}, \quad m = 1, 2, \cdots, m_c \\ \frac{\partial \tilde{y}(n-m)}{\partial c_k(n)} &\approx \frac{\partial \tilde{y}(n-m)}{\partial c_k(n-m)}, \quad m = 1, 2, \cdots, m_c.\end{aligned} \tag{3.86}$$

Hence, (3.85) can be rewritten as

$$\begin{aligned}\frac{\partial \tilde{y}(n)}{\partial d_k(n)} &\approx z_2(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial \tilde{y}(n-m)}{\partial d_k(n-m)}, \quad k = 0, 1, \cdots, m_d \\ \frac{\partial \tilde{y}(n)}{\partial c_k(n)} &\approx \tilde{y}(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial \tilde{y}(n-m)}{\partial c_k(n-m)}, \quad k = 1, \cdots, m_c\end{aligned} \tag{3.87}$$

or

$$\begin{aligned}\frac{\partial \tilde{y}(n)}{\partial d_k(n)} &\approx \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} z_2(n) \\ &= \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} \left(\boldsymbol{\theta}_g^T(n) \boldsymbol{z}(n)\right), \quad k = 0, 1, \cdots, m_d \\ \frac{\partial \tilde{y}(n)}{\partial c_k(n)} &\approx \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} \tilde{y}(n), \quad k = 1, \cdots, m_c.\end{aligned} \tag{3.88}$$

Similarly, differentiating both sides of (3.73) with respect to $g_k(n)$ gives

$$\frac{\partial \tilde{y}(n)}{\partial g_k(n)} = \sum_{m=0}^{m_d} d_m(n) \frac{\partial z_2(n-m)}{\partial g_k(n)} + \sum_{m=1}^{m_c} c_m(n) \frac{\partial \tilde{y}(n-m)}{\partial g_k(n)}, \quad k = 1, \cdots, m_g. \tag{3.89}$$

Again because the parameter vector $\boldsymbol{\theta}(n)$ is assumed to be changing slowly, we can write

$$\begin{aligned} \frac{\partial z_2(n-m)}{\partial g_k(n)} &\approx \frac{\partial z_2(n-m)}{\partial g_k(n-m)}, \quad m = 0, \cdots, m_d \\ \frac{\partial \tilde{y}(n-m)}{\partial g_k(n)} &\approx \frac{\partial \tilde{y}(n-m)}{\partial g_k(n-m)}, \quad m = 1, \cdots, m_c. \end{aligned} \tag{3.90}$$

Hence, (3.89) can be rewritten as

$$\begin{aligned} \frac{\partial \tilde{y}(n)}{\partial g_k(n)} &= \sum_{m=0}^{m_d} d_m(n) \frac{\partial z_2(n-m)}{\partial g_k(n-m)} + \sum_{m=1}^{m_c} c_m(n) \frac{\partial \tilde{y}(n-m)}{\partial g_k(n-m)} \\ &= \sum_{m=0}^{m_d} d_m(n) z^k(n-m) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial \tilde{y}(n-m)}{\partial g_k(n-m)} \\ &= \frac{D(n, \mathsf{z}^{-1})}{1 - C(n, \mathsf{z}^{-1})} z^k(n) = P(n, \mathsf{z}^{-1}) z^k(n), \quad k = 1, \cdots, m_g. \end{aligned} \tag{3.91}$$

Now, we have completely derived the gradient components and hence $\boldsymbol{\varphi}(n)$ can be written as

$$\boldsymbol{\varphi}^T(n) = \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} \tilde{y}(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} \tilde{y}(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} \tilde{y}(n) \end{pmatrix}. \tag{3.92}$$

where $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} \tilde{y}(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} \tilde{y}(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} \tilde{y}(n)$ are given by (3.88) and (3.91), respectively. The RPEM algorithm [57, 63] follows as

$$\begin{aligned} &e(n) = y(n) - \tilde{y}(n) \\ &\lambda(n) = \lambda_0 \lambda(n-1) + 1 - \lambda_0 \\ &s(n) = \boldsymbol{\varphi}^T(n) \boldsymbol{P}(n-1) \boldsymbol{\varphi}(n) + \lambda(n) \\ &\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1) - \boldsymbol{P}(n-1) \boldsymbol{\varphi}(n) s^{-1}(n) \boldsymbol{\varphi}(n)^T \boldsymbol{P}(n-1)\big)/\lambda(n) \\ &\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \boldsymbol{P}(n) \boldsymbol{\varphi}(n) e(n) \end{aligned} \tag{3.93}$$

where λ_0, the initial value $\lambda(0)$ and $\boldsymbol{P}(0)$ are design variables.

3.3.2. Simulation study

The simulation study for the performance of the RPEM algorithm is given in this section. The IIR Wiener system in Section 2.3.4 was considered:

$$\begin{aligned} z(n) &= y_2(n) + 0.25 y_2^2(n) + 0.125 y_2^3(n) \\ y_2(n) &= \frac{0.72 + 1.51\mathsf{z}^{-1} + 1.04\mathsf{z}^{-2} + 0.26\mathsf{z}^{-3}}{1 + 1.46\mathsf{z}^{-1} + 0.89\mathsf{z}^{-2} + 0.18\mathsf{z}^{-3}} y(n) \end{aligned} \tag{3.94}$$

The orders of the linear and nonlinear blocks of the IIR Hammerstein predistorter were chosen the same as in Section 2.3.4 as $m_c = 3$, $m_d = 3$ and $m_g = 9$. The number of

independent experiments was 100 and in each experiment, the input signal was chosen to be a random signal with uniform distribution over $(-1, 1)$ and data length of 4×10^3 samples. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{5}$. The output measurement noise was considered as AWGN with SNR=40 dB.

The parameter vectors $\boldsymbol{\theta}$ were initialized as

$$\begin{aligned}\boldsymbol{\theta}_d(0) &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T \\ \boldsymbol{\theta}_c(0) &= \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T \\ \boldsymbol{\theta}_g(0) &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}^T.\end{aligned} \tag{3.95}$$

Figure 3.9 gives the MSD of the RPEM algorithm. The MSD of the IIR Wiener system without predistorter was about -18 dB. The matrix $\mathbf{P}(0)$ is chosen as $\mathbf{I}$, $\lambda_0 = 0.99$ and $\lambda(0) = 0.95$ for the RPEM algorithms. On average, the RPEM algorithm converges after about 250 samples and the achieved value of E_D is about -40 dB.

Figure 3.10 shows the mean PSDs of the output signals of the IIR Wiener system without and with predistorter after 4×10^3. From this figure, we can see that using the RPEM algorithm, the spectral regrowth in the normalized frequency band $(0.25\pi, 0.60\pi)$ is significantly reduced (up to 40 dB). Compared to the simulation results in Figs. 2.8 and 2.18, the RPEM algorithm can achieve similar performance as the NFxPEM algorithms and SMM method (up to 40 dB reduction).

3.4. Predistortion of Hammerstein Systems

In this section, the adaptation algorithm for predistortion of Hammerstein systems using the ILA-I approach is introduced. The RPEM algorithm is derived and simulation results are given to verify the algorithm.

3.4.1. The ILA-I approach

The ILA-I approach for predistortion of Hammerstein systems is shown in Fig. 3.11. The output of the Hammerstein system is

$$\begin{aligned} z(n) &= H(\mathsf{z}^{-1})y_2(n) = \frac{B(\mathsf{z}^{-1})}{1 - A(\mathsf{z}^{-1})}y_2(n) \\ &= \sum_{m=0}^{m_b} b_m y_2(n-m) + \sum_{m=1}^{m_a} a_m z(n-m) \end{aligned} \tag{3.96}$$

where $H(\mathsf{z}^{-1}) = \frac{B(\mathsf{z}^{-1})}{1-A(\mathsf{z}^{-1})}$ is an IIR filter and the polynomials $A(\mathsf{z}^{-1})$ and $B(\mathsf{z}^{-1})$ are defined as

$$\begin{aligned} A(\mathsf{z}^{-1}) &= \sum_{m=1}^{m_a} a_m \mathsf{z}^{-m} \\ B(\mathsf{z}^{-1}) &= \sum_{m=0}^{m_b} b_m \mathsf{z}^{-m}. \end{aligned} \tag{3.97}$$

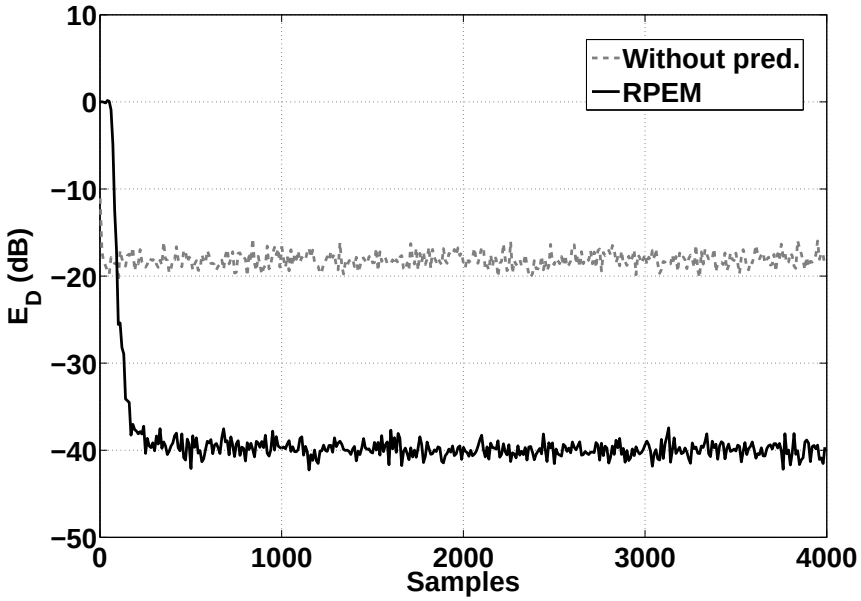

Figure 3.9 MSD E_D: without and with predistorter.

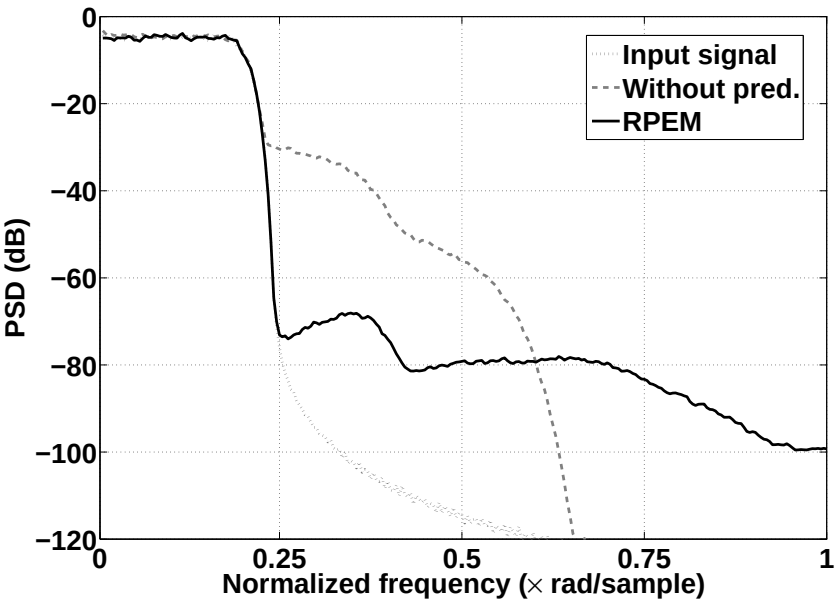

Figure 3.10 Mean PSDs: without and with predistorter.

The intermediate signal $y_2(n)$ is defined as

$$\begin{aligned} y_2(n) &= g_1 y(n) + g_2 y^2(n) + \cdots + g_{m_g} y^{m_g}(n) \\ &= \boldsymbol{g}^T \boldsymbol{y}(n) \end{aligned} \tag{3.98}$$

where $\boldsymbol{g}$ is the parameter vector of the nonlinear subsystem G defined as

$$\boldsymbol{g} = \begin{pmatrix} g_1 & g_2 & \cdots & g_{m_g} \end{pmatrix}^T \tag{3.99}$$

and the corresponding input vector $\boldsymbol{y}(n)$ is given by

$$\boldsymbol{y}(n) = \begin{pmatrix} y(n) & y^2(n) & \cdots & y^{m_g}(n) \end{pmatrix}^T. \tag{3.100}$$

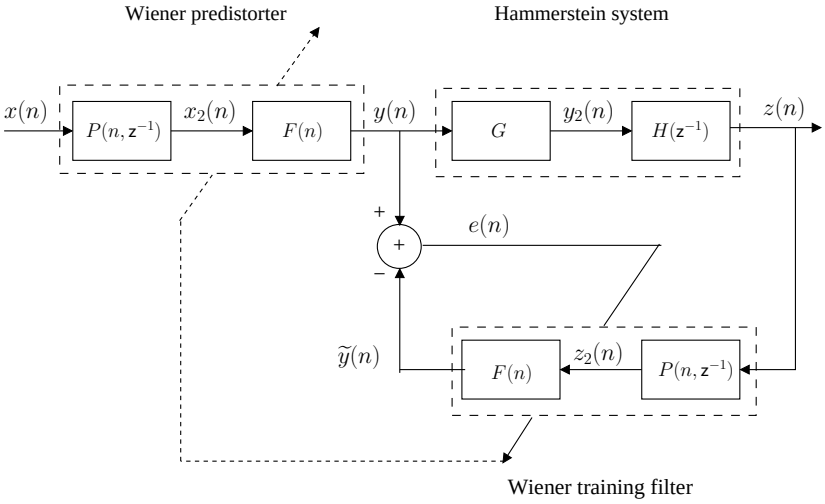

Figure 3.11 The ILA-I approach for predistortion of Hammerstein systems.

The training filter is considered an IIR Wiener system with the output given as

$$\begin{aligned} \tilde{y}(n) &= f_1(n) z_2(n) + f_2(n) z_2^2(n) + \cdots + f_{m_f} z_2^{m_f}(n) \\ &= \boldsymbol{f}^T(n) \boldsymbol{z}_2(n) \end{aligned} \tag{3.101}$$

where $\boldsymbol{f}(n)$ is the parameter vector of the nonlinear subsystem $F(n)$ defined as

$$\boldsymbol{f}(n) = \begin{pmatrix} f_1(n) & f_2(n) & \cdots & f_{m_f}(n) \end{pmatrix}^T \tag{3.102}$$

and the corresponding input vector $\boldsymbol{z}_2(n)$ is given by

$$\boldsymbol{z}_2(n) = \begin{pmatrix} z_2(n) & z_2^2(n) & \cdots & z_2^{m_f}(n) \end{pmatrix}^T. \tag{3.103}$$

The intermediate signal $z_2(n)$ is given by

$$\begin{aligned} z_2(n) &= P(n, \mathsf{z}^{-1}) z(n) = \frac{D(n, \mathsf{z}^{-1})}{1 - C(n, \mathsf{z}^{-1})} z(n) \\ &= \sum_{m=0}^{m_d} d_m(n) z(n-m) + \sum_{m=1}^{m_c} c_m(n) z_2(n-m). \end{aligned} \tag{3.104}$$

where $P(n,\mathsf{z}^{-1}) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}$ is an IIR filter and the polynomials $C(n,\mathsf{z}^{-1})$ and $D(n,\mathsf{z}^{-1})$ are defined as

$$\begin{aligned} C(n,\mathsf{z}^{-1}) &= \sum_{m=1}^{m_c} c_m(n)\mathsf{z}^{-m} \\ D(n,\mathsf{z}^{-1}) &= \sum_{m=0}^{m_d} d_m(n)\mathsf{z}^{-m}. \end{aligned} \tag{3.105}$$

The predistorter is a copy of the training filter, hence the predistorter is also an IIR Wiener model with output signal $y(n)$ given as

$$\begin{aligned} y(n) &= f_1(n)x_2(n) + f_2(n)x_2^2(n) + \cdots + f_{m_f}x_2^{m_f}(n) \\ &= \boldsymbol{f}^T(n)\boldsymbol{x}_2(n) \end{aligned} \tag{3.106}$$

where $\boldsymbol{f}(n)$ is the parameter vector defined as in (3.102) and the corresponding input vector $\boldsymbol{x}_2(n)$ is given by

$$\boldsymbol{x}_2(n) = \begin{pmatrix} x_2(n) & x_2^2(n) & \cdots & x_2^{m_f}(n) \end{pmatrix}^T. \tag{3.107}$$

The intermediate signal $x_2(n)$ is given by

$$\begin{aligned} x_2(n) &= P(n,\mathsf{z}^{-1})x(n) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}x(n) \\ &= \sum_{m=0}^{m_d} d_m(n)x(n-m) + \sum_{m=1}^{m_c} c_m(n)x_2(n-m). \end{aligned} \tag{3.108}$$

where $P(n,\mathsf{z}^{-1}) = \frac{D(n,\mathsf{z}^{-1})}{1-C(n,\mathsf{z}^{-1})}$ is the IIR filter with the polynomials $C(n,\mathsf{z}^{-1})$ and $D(n,\mathsf{z}^{-1})$ defined as in (3.105).

Let us define the parameter vector $\boldsymbol{\theta}$ of the training filter as follows

$$\begin{aligned} \boldsymbol{\theta} &= \begin{pmatrix} \boldsymbol{\theta}_f^T & \boldsymbol{\theta}_d^T & \boldsymbol{\theta}_d^T \end{pmatrix}^T \\ \boldsymbol{\theta}_f &= \begin{pmatrix} f_1 & f_2 & \cdots & f_{m_f} \end{pmatrix}^T \\ \boldsymbol{\theta}_d &= \begin{pmatrix} d_0 & d_1 & \cdots & c_{m_d} \end{pmatrix}^T \\ \boldsymbol{\theta}_c &= \begin{pmatrix} c_1 & c_2 & \cdots & d_{m_c} \end{pmatrix}^T \end{aligned} \tag{3.109}$$

The RPEM algorithm [57, 63] is derived by a minimization of the cost function

$$V(\boldsymbol{\theta}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} E\left[e^2(n)\right] \tag{3.110}$$

where $e(n)$ is the prediction error defined by

$$e(n) = y(n) - \tilde{y}(n). \tag{3.111}$$

The formulation of the RPEM algorithm requires the negative gradient of $e(n)$ with respect to $\boldsymbol{\theta}(n)$. The negative gradient $\boldsymbol{\varphi}(n)$ is defined as

$$\boldsymbol{\varphi}^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}e(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}\tilde{y}(n) = \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)}\tilde{y}(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}\tilde{y}(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}\tilde{y}(n) \end{pmatrix}. \tag{3.112}$$

From (3.101), we have

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)}\tilde{y}(n) = \boldsymbol{z}_2^T(n) = \begin{pmatrix} z_2(n) \\ \vdots \\ z_2^{m_f}(n) \end{pmatrix}^T. \tag{3.113}$$

Since the intermediate signal $z_2(n)$ is not measurable, it can be evaluated using the value of $\boldsymbol{\theta}_d(n)$ and $\boldsymbol{\theta}_c(n)$. Hence, (3.113) becomes

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)}\tilde{y}(n) = \begin{pmatrix} \left[\frac{D(n,z^{-1})}{C(n,z^{-1})}z(n)\right] \\ \vdots \\ \left[\frac{D(n,z^{-1})}{C(n,z^{-1})}z(n)\right]^{m_f} \end{pmatrix}^T = \begin{pmatrix} \left[P(n,\mathsf{z}^{-1})z(n)\right] \\ \vdots \\ \left[P(n,\mathsf{z}^{-1})z(n)\right]^{m_f} \end{pmatrix}^T \tag{3.114}$$

Differentiating both sides of (3.101) with respect to $\boldsymbol{\theta}_c(n)$ and $\boldsymbol{\theta}_d(n)$ gives

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}\tilde{y}(n) &= \boldsymbol{\theta}_f^T(n)\boldsymbol{\nabla}_{z_2(n)}\boldsymbol{z}_2(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}z_2(n) = s_3(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}z_2(n) \\ \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}\tilde{y}(n) &= \boldsymbol{\theta}_f^T(n)\boldsymbol{\nabla}_{z_2(n)}\boldsymbol{z}_2(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}z_2(n) = s_3(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}z_2(n) \end{aligned} \tag{3.115}$$

where

$$\begin{aligned} s_3(n) &= \boldsymbol{\theta}_f^T(n)\boldsymbol{\nabla}_{z_2(n)}\boldsymbol{z}_2(n) = \boldsymbol{\theta}_f^T(n)\begin{pmatrix} 1 \\ 2z_2(n) \\ \vdots \\ m_f z_2^{m_f-1}(n) \end{pmatrix} \\ &= \boldsymbol{\theta}_f^T(n)\begin{pmatrix} 1 \\ 2\left[P(n,\mathsf{z}^{-1})z(n)\right] \\ \vdots \\ m_f\left[P(n,\mathsf{z}^{-1})z(n)\right]^{m_f-1} \end{pmatrix}. \end{aligned} \tag{3.116}$$

Now, it remains to derive $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}z_2(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}z_2(n)$. Differentiating both sides of (3.104) with respect to $d_k(n)$ and $c_k(n)$ gives

$$\begin{aligned} \frac{\partial z_2(n)}{\partial d_k(n)} &= z(n-k) + \sum_{m=1}^{m_c} c_m(n)\frac{\partial z_2(n-m)}{\partial d_k(n)} \\ \frac{\partial z_2(n)}{\partial c_k(n)} &= z_2(n-k) + \sum_{m=1}^{m_c} c_m(n)\frac{\partial z_2(n-m)}{\partial c_k(n)}. \end{aligned} \tag{3.117}$$

Assuming that the parameter vector $\boldsymbol{\theta}(n)$ is changing slowly [23, 26], we can write

$$\begin{aligned} \frac{\partial z_2(n-m)}{\partial d_k(n)} &\approx \frac{\partial z_2(n-m)}{\partial d_k(n-m)}, \quad m = 1, 2, \cdots, m_c \\ \frac{\partial z_2(n-m)}{\partial c_k(n)} &\approx \frac{\partial z_2(n-m)}{\partial c_k(n-m)}, \quad m = 1, 2, \cdots, m_c. \end{aligned} \tag{3.118}$$

Hence, (3.117) can be rewritten as

$$\begin{aligned}\frac{\partial z_2(n)}{\partial d_k(n)} &\approx z(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial z_2(n-m)}{\partial d_k(n-m)} \\ \frac{\partial z_2(n)}{\partial c_k(n)} &\approx z_2(n-k) + \sum_{m=1}^{m_c} c_m(n) \frac{\partial z_2(n-m)}{\partial c_k(n-m)}\end{aligned} \tag{3.119}$$

or

$$\begin{aligned}\frac{\partial z_2(n)}{\partial d_k(n)} &\approx \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} z(n), \quad k = 0, 1, \cdots, m_d \\ \frac{\partial z_2(n)}{\partial c_k(n)} &\approx \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} z_2(n) \\ &= \frac{\mathsf{z}^{-k}}{1 - C(n, \mathsf{z}^{-1})} \left[P(n, \mathsf{z}^{-1}) z(n) \right], \quad k = 1, \cdots, m_c.\end{aligned} \tag{3.120}$$

Now, we have completely derived the components of $\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)} \tilde{y}(n)$ in (3.112) and hence $\boldsymbol{\varphi}(n)$ can be written as

$$\boldsymbol{\varphi}^T(n) = \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} \tilde{y}(n) & s_3(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} z_2(n) & s_3(n) \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} z_2(n) \end{pmatrix} \tag{3.121}$$

where $s_3(n)$ is calculated in (3.116), and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)} \tilde{y}(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} z_2(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} z_2(n)$ are given by (3.114) and (3.120), respectively. The RPEM algorithm [57, 63] follows as

$$\begin{aligned}&e(n) = y(n) - \tilde{y}(n) \\ &\lambda(n) = \lambda_0 \lambda(n-1) + 1 - \lambda_0 \\ &s(n) = \boldsymbol{\varphi}^T(n) \boldsymbol{P}(n-1u) \boldsymbol{\varphi}(n) + \lambda(n) \\ &\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1) - \boldsymbol{P}(n-1) \boldsymbol{\varphi}(n) s^{-1}(n) \boldsymbol{\varphi}(n)^T \boldsymbol{P}(n-1) \big) / \lambda(n) \\ &\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \boldsymbol{P}(n) \boldsymbol{\varphi}(n) e(n).\end{aligned} \tag{3.122}$$

3.4.2. Simulation study

The simulation study for the performance of the RPEM algorithm is given in this section. The IIR Hammerstein system in Section 2.4.3 was considered:

$$\begin{aligned}z(n) &= \frac{0.72 + 1.51\mathsf{z}^{-1} + 1.04\mathsf{z}^{-2} + 0.26\mathsf{z}^{-3}}{1 + 1.46\mathsf{z}^{-1} + 0.89\mathsf{z}^{-2} + 0.18\mathsf{z}^{-3}} y_2(n) \\ y_2(n) &= y(n) + 0.25 y^2(n) + 0.125 y^3(n).\end{aligned} \tag{3.123}$$

The order of the linear and nonlinear blocks of the IIR Wiener predistorter were chosen the same as in Section 2.4.3 as $m_c = 3$, $m_d = 3$ and $m_f = 9$, respectively. The number of independent experiments was 100 and in each experiment, the input signal was a random signal with uniform distribution over $(-1, 1)$ with data length of 5×10^3 samples. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{5}$. The output measurement noise was considered as AWGN with SNR=40 dB.

The parameter vectors $\boldsymbol{\theta}$ were initialized as

$$\begin{aligned}\boldsymbol{\theta}_f(0) &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}^T \\ \boldsymbol{\theta}_d(0) &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T \\ \boldsymbol{\theta}_c(0) &= \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T.\end{aligned} \tag{3.124}$$

Figure 3.12 gives the MSD of the RPEM algorithm. The MSD of the IIR Hammerstein system without predistorter was about -17 dB. The matrix $\mathbf{P}(0)$ is chosen as $\mathbf{I}$, $\lambda_0 = 0.99$ and $\lambda(0) = 0.95$ for the RPEM algorithms. On average, the RPEM algorithm converges after about 250 samples and the achieved value of E_D is about -40 dB.

Figure 3.13 shows the mean PSDs of the output signals of the IIR Hammerstein system without and with predistorter after 5×10^3 samples. From this figure, we can see that using the RPEM algorithm, there is a significant spectral regrowth reduction in the normalized frequency band $(0.25\pi, 0.65\pi)$. Compared to the simulation results in Figs. 2.11 and 2.20, the RPEM algorithm can achieve similar performance as the NFxPEM algorithms and SMM method (up to 60 dB reduction).

3.5. Summary

Adaptive predistortion of nonlinear physical systems using the ILA approach is introduced in this chapter. The coefficients of the predistorter can be estimated using different time domain adaptation algorithms.

The ILA-I approach can be applied for predistortion of different nonlinear systems, such as Volterra, Wiener and Hammerstein systems. For the predistortion of Volterra systems, both the training filter and predistorter are modeled as Volterra systems. The predistorter is a copy of the training filter, and the RLS, KF and RPEM algorithms can be used to estimate the coefficients of the training filter. These algorithms have similar performance and the predistorter can efficiently reduce the spectral regrowth caused by the nonlinear physical system. The RPEM algorithm can also be used for predistortion of Wiener and Hammerstein systems. From the simulation, we can see that the RPEM algorithm has very fast convergence speed, as well as good performance in reducing the nonlinear distortion in the time domain and the spectral regrowth in the frequency domain.

If the nonlinear physical system is a weakly nonlinear system and it can be divided into two subsystems: the sum of a purely linear subsystem and a purely nonlinear subsystem. In this way, the ILA-II approach can be applied to construct the predistorter and estimate its coefficients. The LMS algorithm is a widely used adaptation algorithm to estimate the coefficients for the predistorter, however, the slow convergence speed and inaccurate estimates degrade the performance of the predistorter. The proposed RPEM algorithm has much faster convergence speed and can obtain more accurate estimates - hence greatly improve the performance of the predistorter.

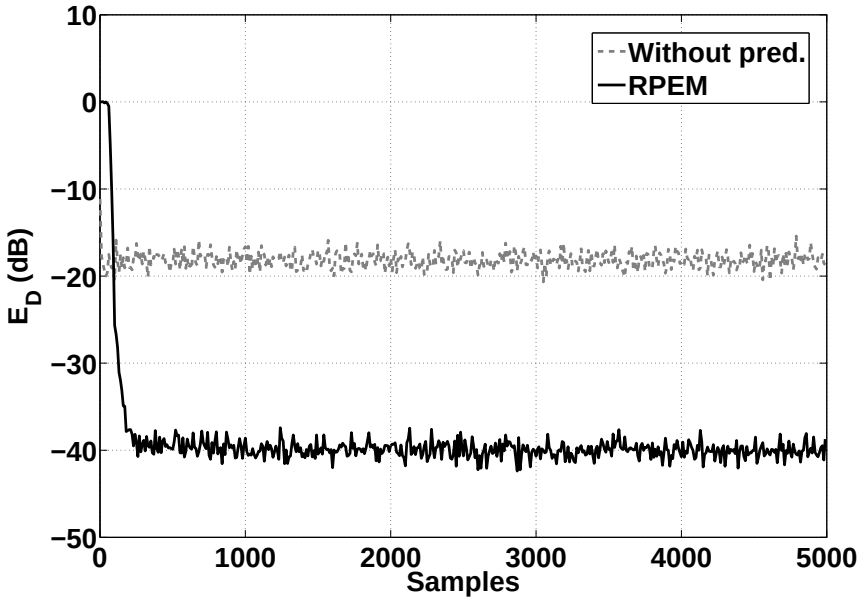

Figure 3.12 MSD E_D: without and with predistorter.

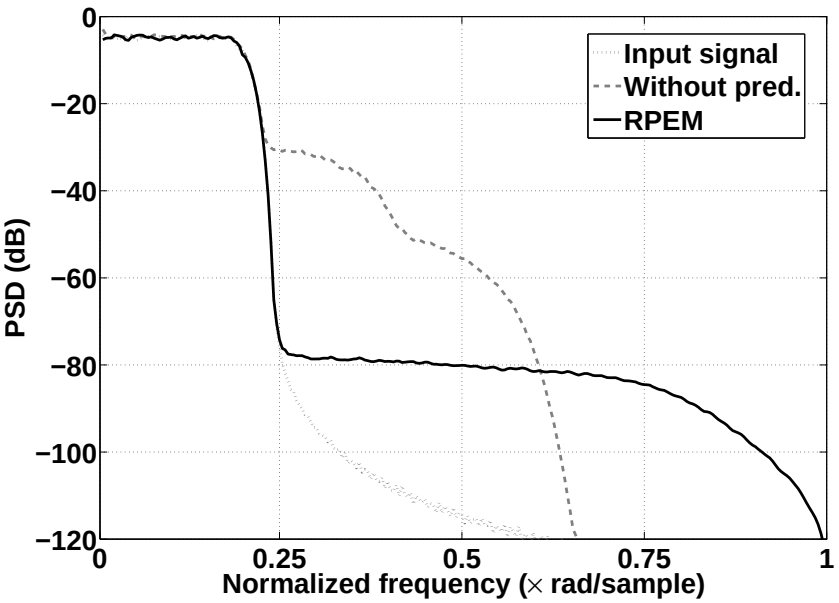

Figure 3.13 Mean PSDs: without and with predistorter.

Chapter 4

Adaptive Predistorter Design

The adaptive predistorter design is discussed in this chapter. First, we propose the General Gradient Calculation Architecture (GGCA) for the adaptive predistortion using different learning architectures and algorithms. Then, some additional aspects of adaptive predistorter design are summarized and comparisons of all adaptation algorithms are given.

4.1. General Gradient Calculation Architecture

As introduced in Chapter 2 and Chapter 3, there are two learning architectures for estimating the coefficients of the predistorter, which are the direct learning architecture (DLA) and the indirect learning architecture (ILA). In this section, we will focus on the DLA and ILA-I approaches.

Several adaptation algorithms based on the DLA and ILA-I approaches have been proposed, see Table 4.1. Normally, the adaptation algorithms for the DLA approach need to calculate the gradient of the output signal of the nonlinear physical system w.r.t. the coefficients of the predistorter, while the adaptation algorithms for the ILA-I approach need to calculate the gradient of the output signal of the training filter w.r.t. its coefficients. However, the common ground of the gradient calculation for all these algorithms has not been considered in the literature before. If a common architecture for the gradient calculation can be defined, then all the corresponding adaptation algorithms can use this architecture directly as a black box, without dealing with the detail design for gradient calculation.

In the following sections, a GGCA is proposed for adaptive predistortion of different nonlinear physical systems, such as Volterra, Wiener and Hammerstein systems, respectively. In each section, the gradient vector required the adaptation algorithms for the DLA and ILA-I approaches are summarized and then the related GGCA is proposed. There are separate out-

Learning architecture	Adaptation algorithms
DLA	NFxLMS, NFxLMS-ISE, NFxRLS, NFxPEM, NFxPEM-ISE
ILA-I	LMS, RLS, KF, RPEM

Table 4.1 Adaptation algorithms for the DLA and ILA-I approaches.

puts in the GGCA, providing the gradient calculation required in the adaptation algorithms for the DLA and ILA-I approaches.

4.1.1. Architecture for predistortion of Volterra systems

The DLA approach

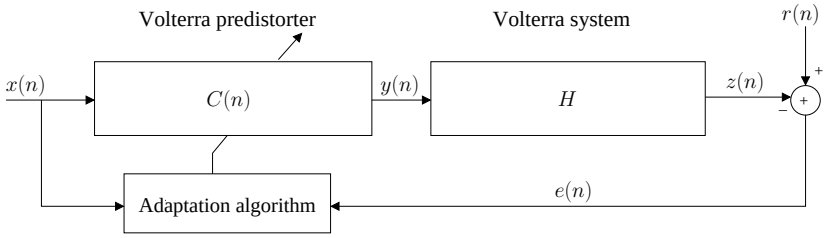

Figure 4.1 The DLA approach for predistortion of Volterra systems.

The DLA approach for predistortion of Volterra systems is shown in Fig. 4.1. The nonlinear physical system H is a qth-order Volterra system, and the predistorter $C(n)$ is a pth-order Volterra system with the same definition as in Section 2.2.

Based on the analysis in Section 2.2.1 to 2.2.3, all the adaptation algorithms, such as NFxLMS, Nonlinear Filtered-x Recursive Least Squares (NFxRLS) and NFxPEM algorithms, need to calculate the gradient of the output signal $z(n)$ w.r.t. the parameter vector of the predistorter $\boldsymbol{c}(n)$. Therefore, we can define the gradient vector $\boldsymbol{\varphi}_g(n)$ as

$$\begin{aligned}\boldsymbol{\varphi}_g^T(n) &= \boldsymbol{\nabla}_{\boldsymbol{c}(n)} z(n) = \sum_{r=0}^{M-1} g(r;n)\boldsymbol{\nabla}_{\boldsymbol{c}(n)} y(n-r) = \sum_{r=0}^{M-1} g(r;n)\boldsymbol{x}^T(n-r) \\ &= G(n,\mathsf{z}^{-1})\boldsymbol{x}^T(n) \end{aligned} \tag{4.1}$$

where $G(n,\mathsf{z}^{-1})$ denotes the Finite Impulse Response (FIR) filter defined as

$$G(n,\mathsf{z}^{-1}) = g(0;n) + g(1;n)\mathsf{z}^{-1} + \cdots + g(M-1;n)\mathsf{z}^{-M+1}. \tag{4.2}$$

Here, z^{-1} is the delay operator such that $\mathsf{z}^{-m}\boldsymbol{x}(n) = \boldsymbol{x}(n-m)$ and $g(r;n)$ is defined as

$$\begin{aligned} g(r;n) = \frac{\partial z(n)}{\partial y(n-r)} &= h_1(r) + 2\sum_{i=0}^{M_2-1} h_2(r,i)y(n-i) + \\ &\quad 3\sum_{i_1=0}^{M_3-1}\sum_{i_2=0}^{M_3-1} h_3(r,i_1,i_2)y(n-i_1)y(n-i_2) + \cdots \end{aligned} \tag{4.3}$$

Note that here we define the gradient vector as $\boldsymbol{\varphi}_g(n)$ in order to distinguish it from the gradient vector $\boldsymbol{\varphi}(n)$ defined in the next section.

The ILA-I approach

The ILA-I approach for predistortion of Volterra systems is shown in Fig. 4.2. The training filter $\tilde{C}(n)$ is defined as in Section 3.2.1.

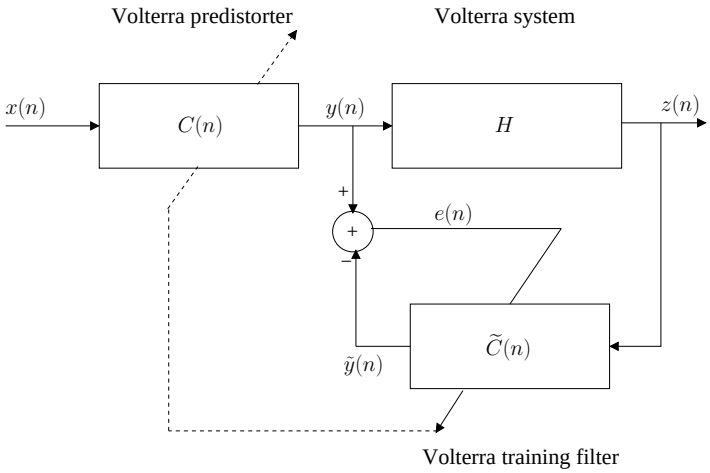

Figure 4.2 The ILA-I approach for predistortion of Volterra systems.

From the analysis in Section 3.2.1, all the adaptation algorithms, such as the Recursive Least Squares (RLS), Kalman Filter (KF) and RPEM algorithms, need the gradient of the signal $\tilde{y}(n)$ w.r.t. the parameter vector of the training filter $\boldsymbol{c}(n)$. The gradient vector $\boldsymbol{\varphi}(n)$ is given by

$$\boldsymbol{\varphi}^T(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)}\tilde{y}(n) = \boldsymbol{z}^T(n). \tag{4.4}$$

The GGCA for predistortion of Volterra systems

From (4.1) and (4.4), we can see that the calculation results of $\boldsymbol{\nabla}_{\boldsymbol{c}(n)}y(n-r)$ and $\boldsymbol{\nabla}_{\boldsymbol{c}(n)}\tilde{y}(n)$ are just data vectors. Also, $\boldsymbol{\varphi}_g(n)$ can be obtained by filtering the data vector using the FIR filter $G(n, \mathsf{z}^{-1})$. Therefore, the GGCA for predistortion of Volterra systems can be given as in Fig. 4.3.

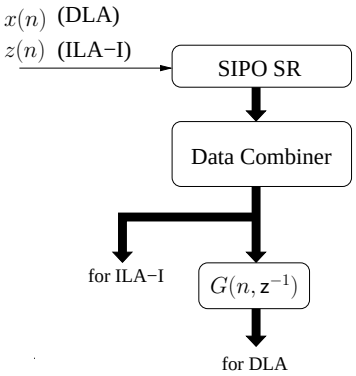

Figure 4.3 The GGCA for predistortion of Volterra systems.

In order to calculate the gradient vector $\boldsymbol{\varphi}(n)$ in the ILA-I approach, the input signal $z(n)$ is first buffered by the Single Input Parallel Output Shift Register (SIPO SR) with memory length M, shown in Fig. 4.4. The output of the SIPO SR is the data vector $\boldsymbol{z}_1(n)$, defined as

$$\boldsymbol{z}_1(n) = \left(z(n) \quad z(n-1) \quad \cdots \quad z(n-M+1)\right).$$

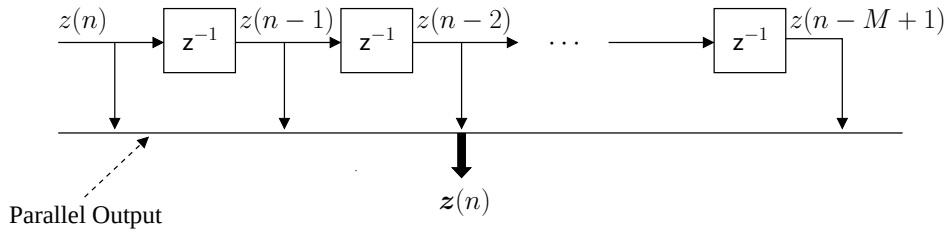

Figure 4.4 SIPO SR.

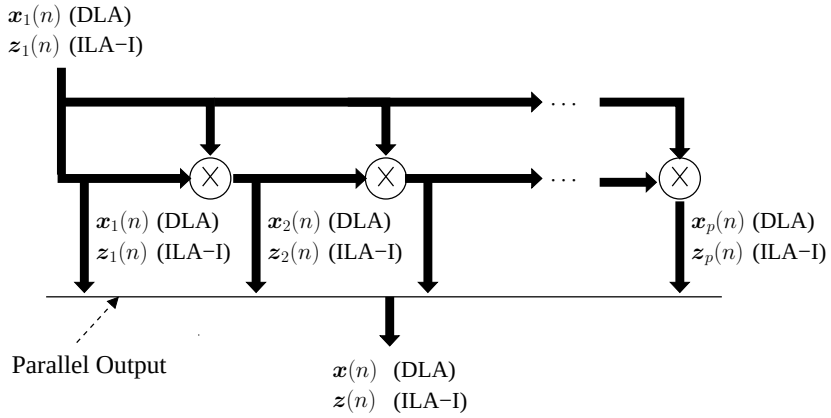

Figure 4.5 Data combiner.

Then, the output of the SIPO SR, $\boldsymbol{z}_1(n)$, is processed by a data combiner in order to construct the input vector $\boldsymbol{z}(n)$ defined in (3.9). The structure of the data combiner is given in Fig. 4.5, where $\boldsymbol{z}_k(n)$ is the input vector for the kth-order Volterra kernel defined in (3.10). Since the gradient vector $\boldsymbol{\varphi}(n) = \boldsymbol{z}(n)$, the data vector $\boldsymbol{z}(n)$ can be used directly in the adaptation algorithms of the ILA-I approach.

Similarly, using the input signal $x(n)$ generates the data vector $\boldsymbol{x}(n)$ defined in (2.15), and the gradient vector $\boldsymbol{\varphi}_g(n)$ in the DLA approach can be obtained by filtering the data vector $\boldsymbol{x}(n)$ using the FIR filter $G(n, \mathsf{z}^{-1})$.

4.1.2. Architecture for predistortion of Wiener systems

The DLA approach

The DLA approach for predistortion of Wiener systems is shown in Fig. 4.6. The IIR Wiener system and the IIR Hammerstein predistorter are defined as in Section 2.3. Also, $\boldsymbol{\theta}$ is defined as the parameter vector of the predistorter, see (2.52).

Based on the analysis in Section 2.3.1 to 2.3.3, we can define the gradient vector $\boldsymbol{\varphi}_h(n)$ as

$$\boldsymbol{\varphi}_h^T(n) = s_1(n) \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} y(n) \end{pmatrix} \tag{4.5}$$

where $s_1(n)$ is calculated in (2.56), and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)} y(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)} y(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)} y(n)$ are given by (2.65) and (2.68), respectively. Here, $\boldsymbol{\varphi}_h(n)$ is required in both NFxLMS and NFxPEM algorithms. Substituting $H(\mathsf{z}^{-1})$ and $\boldsymbol{f}$ by the initial subsystem estimates $\widehat{H}(\mathsf{z}^{-1})$ and $\widehat{\boldsymbol{f}}$,

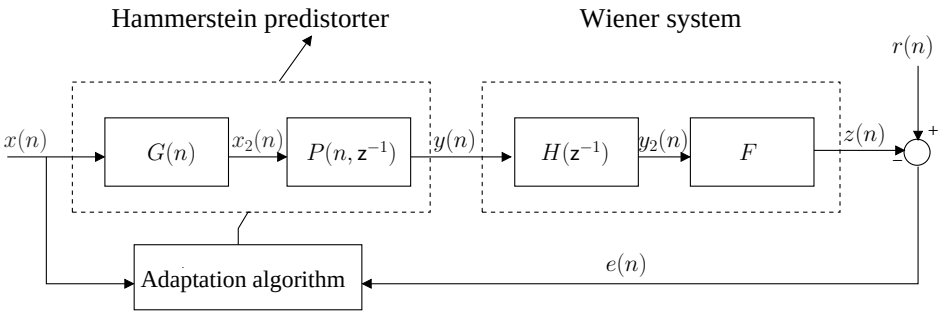

Figure 4.6 The DLA approach for predistortion of Wiener systems.

$\boldsymbol{\varphi}_h(n)$ can also be used in the NFxLMS with Initial Subsystem Estimates (NFxLMS-ISE) and NFxPEM-ISE algorithms.

The ILA-I approach

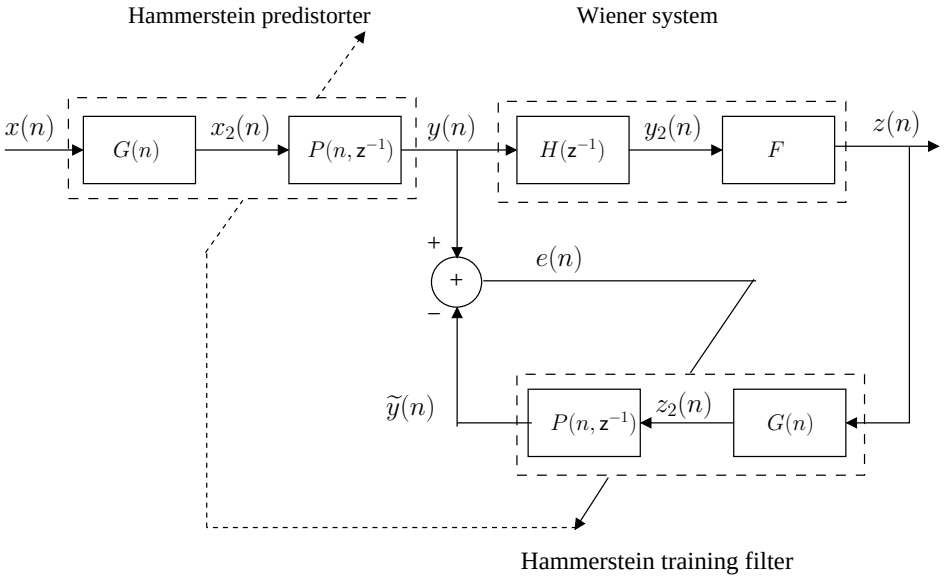

Figure 4.7 The ILA-I approach for predistortion of Wiener systems.

The ILA-I approach for predistortion of Wiener systems is shown in Fig. 4.7. The training filter is modeled as an IIR Hammerstein system defined as in Section 3.3.1. The parameter vector $\boldsymbol{\theta}$ of the training filter is defined as in (3.81).

Based on the analysis in Section 3.3.1, the required negative gradient of $e(n)$ with respect to $\boldsymbol{\theta}(n)$ is equal to the gradient of $\tilde{y}(n)$ with respect to $\boldsymbol{\theta}(n)$, see (3.84). The gradient vector $\boldsymbol{\varphi}(n)$ is given by

$$\boldsymbol{\varphi}^T(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}\tilde{y}(n) = \begin{pmatrix} \boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}\tilde{y}(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}\tilde{y}(n) & \boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)}\tilde{y}(n) \end{pmatrix} \tag{4.6}$$

where $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}\tilde{y}(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}\tilde{y}(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_g(n)}\tilde{y}(n)$ are given by (3.88) and (3.91), respectively.

The GGCA for predistortion of Wiener systems

From (2.65), (2.68) and (3.88), (3.91), we can see that the calculation of $\nabla_{\boldsymbol{\theta}(n)} y(n)$ and $\nabla_{\boldsymbol{\theta}(n)} \tilde{y}(n)$ can be evaluated using similar calculation structures. Also, as quite obvious from (4.5) and (4.6), $\boldsymbol{\varphi}_h(n)$ can be obtained by filtering $\boldsymbol{\varphi}(n)$ using the IIR filter $H(\mathsf{z}^{-1})$. Therefore, the GGCA for predistortion of Wiener systems can be given as in Fig. 4.8.

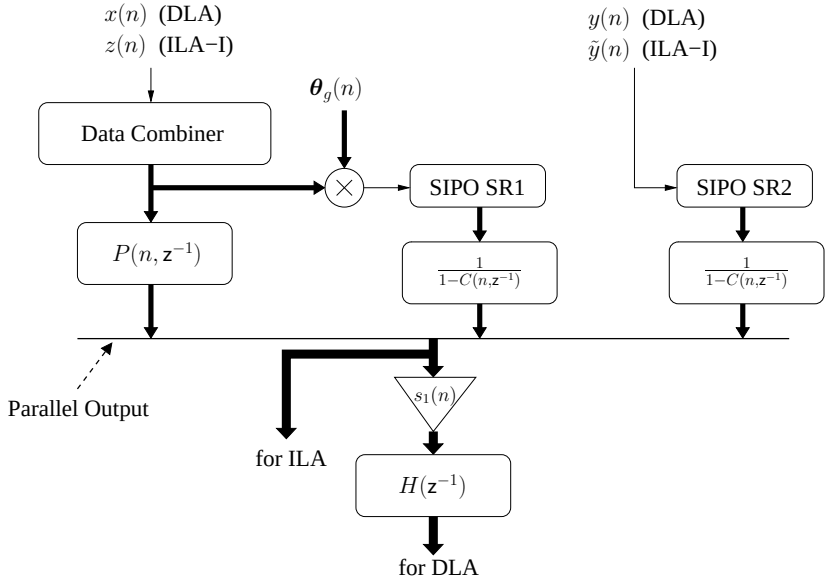

Figure 4.8 The GGCA for predistortion of Wiener systems.

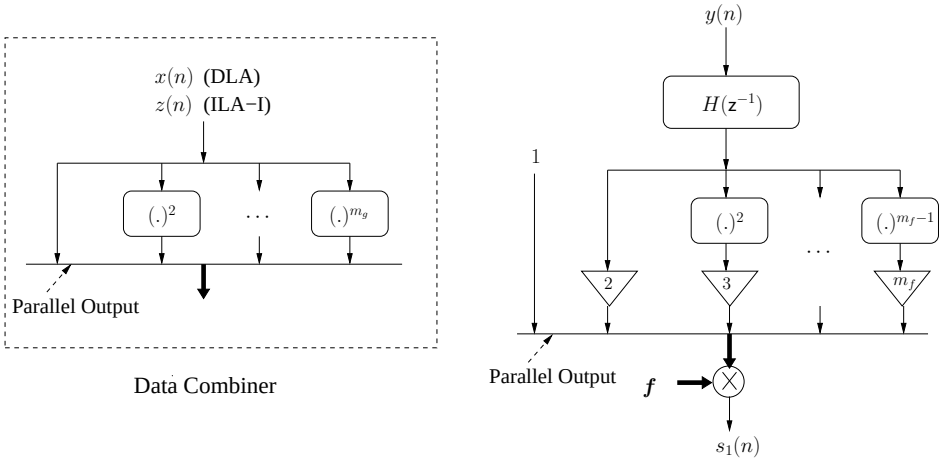

Figure 4.9 Data combiner and the calculation of $s_1(n)$.

When the input signals are chosen as $z(n)$ and $\tilde{y}(n)$, the GGCA calculates the gradient vector $\boldsymbol{\varphi}(n)$ for the ILA-I approach. The data combiner combines the input signal $z(n)$ into data vector $\boldsymbol{z}(n)$ defined in (3.77), and the structure of the data combiner is given in Fig. 4.9. $\boldsymbol{z}(n)$ is filtered by the IIR filter $P(n, \mathsf{z}^{-1})$ to calculate $\nabla_{\boldsymbol{\theta}_g(n)} \tilde{y}(n)$. Also, $\boldsymbol{z}(n)$ is multiplied

with $\boldsymbol{\theta}_g(n)$ to obtain $z_2(n)$, which is buffered in the SIPO SR1 with memory length m_d+1. The output of the SIPO SR1 is then filtered by the IIR filter $\frac{1}{1-C(n,\mathsf{z}^{-1})}$ to obtain $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}\tilde{y}(n)$. Similarly, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}\tilde{y}(n)$ is calculated by buffering $\tilde{y}(n)$ in SIPO SR2 with memory length m_c and filtering the output of the SIPO SR2 by the IIR filter $\frac{1}{1-C(n,\mathsf{z}^{-1})}$.

When the input signals are chosen as $x(n)$ and $y(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}y(n)$ can be obtained by using the same calculation structure. In order to calculate the gradient vector $\boldsymbol{\varphi}_h(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}y(n)$ needs to be multiplied with $s_1(n)$ which is calculated in Fig. 4.9 and then filtered by the IIR filter $H(\mathsf{z}^{-1})$.

4.1.3. Architecture for predistortion of Hammerstein systems

The DLA approach

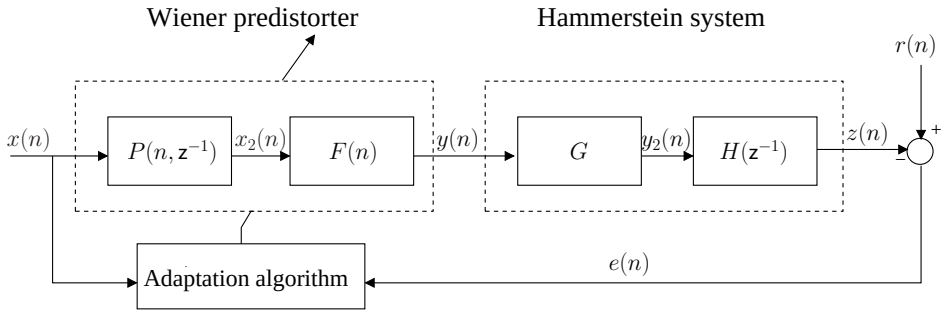

Figure 4.10 The DLA approach for predistortion of Hammerstein systems.

The DLA approach for predistortion of Hammerstein systems is shown in Fig. 4.10. The IIR Hammerstein system and the IIR Wiener predistorter are defined as in Section 2.4. Also, $\boldsymbol{\theta}$ is defined as the parameter vector of the predistorter, see (2.92).

Based on the analysis in Section 2.4.1, we can define the gradient vector $\boldsymbol{\varphi}_h(n)$ as

$$\boldsymbol{\varphi}_h^T(n) = H(\mathsf{z}^{-1})s_1(n)\left(\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)y(n)} \quad s_2(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)x_2(n)} \quad s_2(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)x_2(n)}\right) \tag{4.7}$$

where $s_1(n)$ and $s_2(n)$ are calculated in (2.99) and (2.103), respectively, and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)y(n)}$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)x_2(n)}$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)x_2(n)}$ are given by (2.101) and (2.107), respectively.

The ILA-I approach

The ILA-I approach for predistortion of Hammerstein systems is shown in Fig. 4.11. The training filter is modeled as an IIR Wiener system defined as in Section 3.4.1. The parameter vector $\boldsymbol{\theta}$ of the training filter is defined as in (3.109).

Based on the analysis in Section 3.4.1, the required negative gradient of $e(n)$ with respect to $\boldsymbol{\theta}(n)$ is equal to the gradient of $\tilde{y}(n)$ with respect to $\boldsymbol{\theta}(n)$, see (3.112). The gradient vector $\boldsymbol{\varphi}(n)$ is given by

$$\boldsymbol{\varphi}^T(n) = \boldsymbol{\nabla}_{\boldsymbol{\theta}(n)}\tilde{y}(n) = \left(\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)}\tilde{y}(n) \quad s_3(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}z_2(n) \quad s_3(n)\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}z_2(n)\right). \tag{4.8}$$

where $s_3(n)$ is calculated in (3.116), and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_f(n)}\tilde{y}(n)$, $\boldsymbol{\nabla}_{\boldsymbol{\theta}_d(n)}z_2(n)$ and $\boldsymbol{\nabla}_{\boldsymbol{\theta}_c(n)}z_2(n)$ are given by (3.114) and (3.120), respectively.

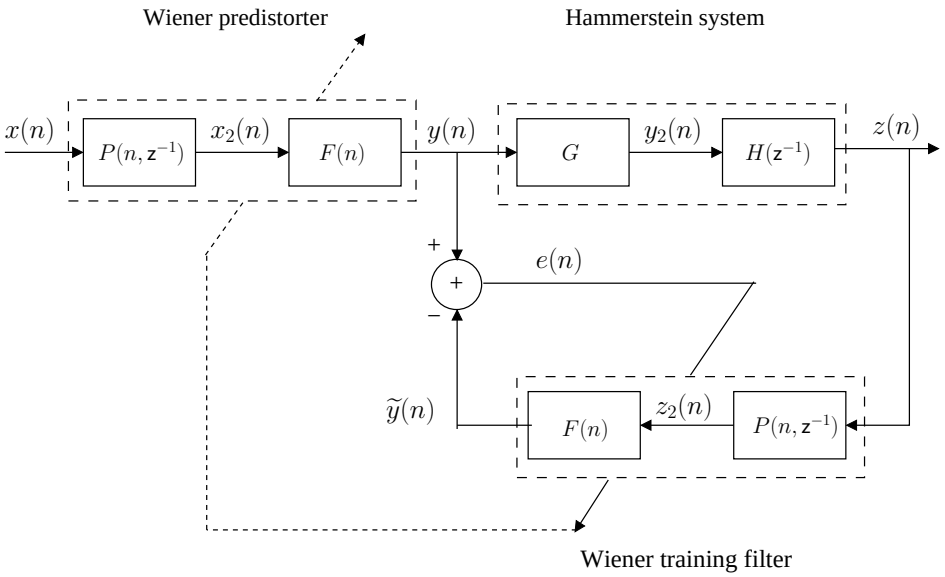

Figure 4.11 The ILA-I approach for predistortion of Hammerstein systems.

The GGCA for predistortion of Hammerstein systems

From (2.101), (2.107) and (3.114), (3.120), we can see that the calculation of $\nabla_{\boldsymbol{\theta}(n)} y(n)$ and $\nabla_{\boldsymbol{\theta}(n)} \tilde{y}(n)$ can be evaluated using similar calculation structures. Also, as quite obvious from (4.7) and (4.8), $\boldsymbol{\varphi}_h(n)$ can be obtained by filtering $\boldsymbol{\varphi}(n)$ using the IIR filter $H(\mathsf{z}^{-1})$. Therefore, the GGCA for predistortion of Hammerstein systems can be given as in Fig. 4.12.

When the input signal is chosen as $z(n)$, the GGCA calculates the gradient vector $\boldsymbol{\varphi}(n)$ for the ILA-I approach. $z(n)$ is first filtered by the IIR filter $P(n,\mathsf{z}^{-1})$, then the filter output goes through the data combiner, see Fig. 4.13, in order to calculate $\nabla_{\boldsymbol{\theta}_f(n)} \tilde{y}(n)$. Also, the filter output is buffered by the SIPO SR2 with memory length m_c, the output of the SIPO SR2 is filtered by the IIR filter $\frac{1}{1-C(n,\mathsf{z}^{-1})}$ and multiplies $s_3(n)$ to obtain $\nabla_{\boldsymbol{\theta}_c(n)} z_2(n)$. Similarly, $\nabla_{\boldsymbol{\theta}_d(n)} z_2(n)$ is obtained by buffering $z(n)$ using the SIPO SR1 with memory length m_d+1, filtering the output of SIPO SR1 by the IIR filter $\frac{1}{1-C(n,\mathsf{z}^{-1})}$ and multiplying $s_3(n)$. The calculation of the scalar $s_3(n)$ is given in Fig. 4.13, too.

When the input signal is chosen as $x(n)$, $\nabla_{\boldsymbol{\theta}(n)} y(n)$ can be obtained by using the same calculation structure. In order to calculate the gradient vector $\boldsymbol{\varphi}_h(n)$, $\nabla_{\boldsymbol{\theta}(n)} y(n)$ needs to be multiplied with $s_1(n)$ which is calculated in Fig. 4.13 and filtered by the IIR filter $H(\mathsf{z}^{-1})$.

4.2. Additional Issues for Adaptive Predistorter Design

In this section, several important issues for adaptive predistorter design are discussed, including predistorter models, learning architectures, adaptation algorithms and computational complexity.

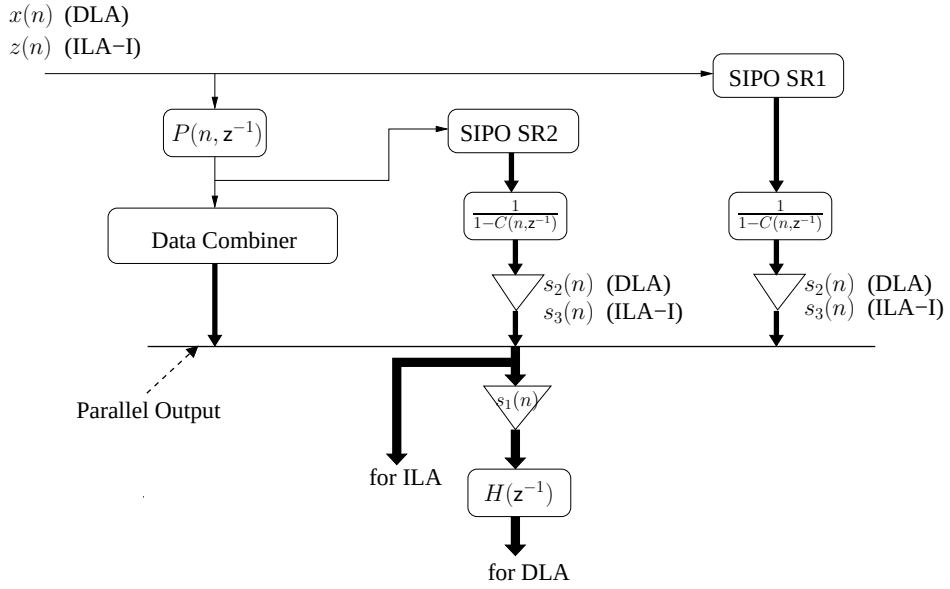

Figure 4.12 The GGCA for predistortion of Hammerstein systems.

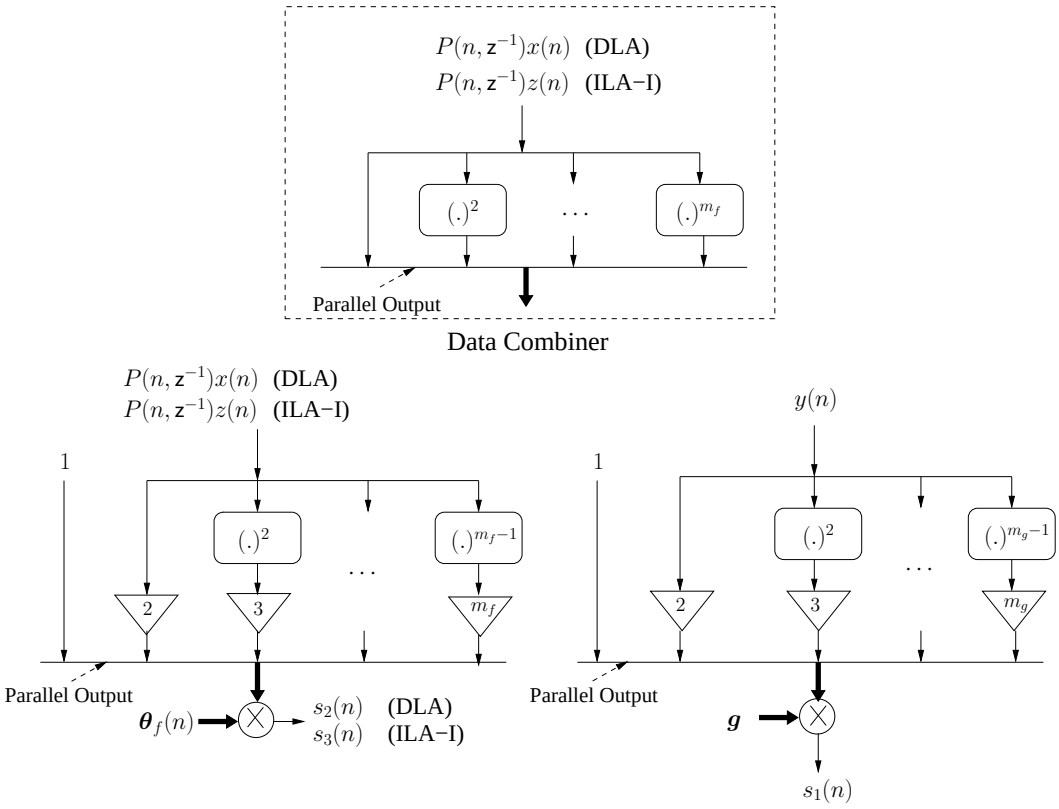

Figure 4.13 Data combiner, the calculation of $s_1(n)$, $s_2(n)$ and $s_3(n)$.

4.2.1. The predistorter model

The nonlinear physical system which we would like to compensate can be modeled using some general nonlinear models, such as Volterra, Wiener or Hammerstein models. The inverse of Volterra systems using Volterra predistorters is discussed in [13]. According to the pth-order inverse theory proposed in [13], the Volterra predistorter can remove nonlinearities up to pth-order provided that the inverse of the first-order Volterra system is causal and stable. Therefore, Volterra series is a proper predistorter model for predistortion of Volterra systems.

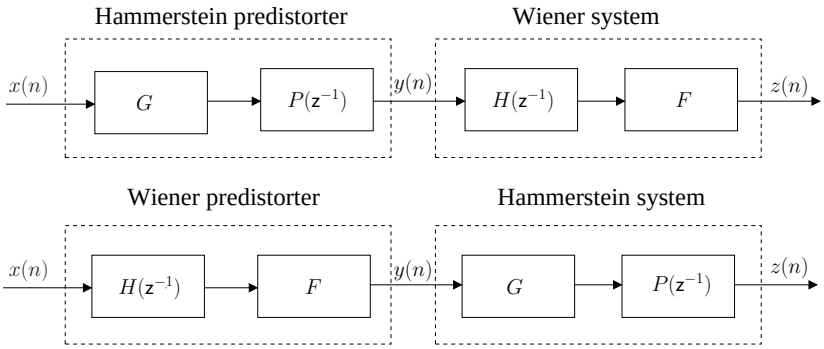

Figure 4.14 The predistorter models for the Wiener and Hammerstein systems.

The Wiener model consists of a linear dynamic system $H(\mathsf{z}^{-1})$ followed by a static nonlinearity F, and in the Hammerstein model the static nonlinearity G precedes the linear dynamic system $P(\mathsf{z}^{-1})$, see Fig. 4.14. When the nonlinear physical system is a Wiener system, assuming that the predistorter is a Hammerstein model, the output of the predistorter $y(n)$ is

$$y(n) = P(\mathsf{z}^{-1})G[x(n)] \tag{4.9}$$

and the output of the Wiener model $z(n)$ can be written as

$$z(n) = F[H(\mathsf{z}^{-1})y(n)] = F\big[H(\mathsf{z}^{-1})P(\mathsf{z}^{-1})G[x(n)]\big]. \tag{4.10}$$

Assuming that $H(\mathsf{z}^{-1})P(\mathsf{z}^{-1}) = \mathsf{z}^{-\tau_1}$ and $F[G] = \mathsf{z}^{-\tau_2}$, where τ_1, τ_2 are time delays and z^{-1} is the delay operator, $z(n)$ can be rewritten as

$$z(n) = F\big[G[x(n-\tau_1)]\big] = x(n-\tau_1). \tag{4.11}$$

Note that here $\tau_2 = 0$ since F and G are static nonlinearities. From (4.11), we can see that the output of the Wiener system with predistorter is equal to the delayed input signal and the linear and nonlinear distortion of the Wiener system is well reduced. Therefore, the Hammerstein model is a proper model for the predistorter. Similarly analysis can be made to show that the Wiener model is a proper predistorter model for predistortion of Hammerstein systems.

The Volterra model can also be used as a predistorter model, since the Wiener and Hammerstein systems are particular cases of the truncated Volterra systems [14]. However, the Volterra predistorter has much more coefficients and higher computational complexity compared to the Hammerstein and Wiener predistorters, in case they have the same order and memory length.

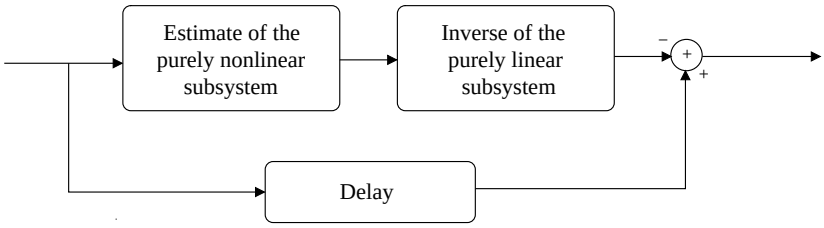

Figure 4.15 The predistorter models for the Wiener and Hammerstein systems.

A general approach to model the predistorter is proposed in [9], under the condition that the nonlinear physical system is a weakly nonlinear system and can be decomposed into the sum of two subsystems: one is the purely linear subsystem and the other is the purely nonlinear subsystem. The predistorter can be constructed as in Fig. 4.15 using the estimate of the purely nonlinear subsystem, the inverse of the purely linear subsystem and the delayed input signal. This type of predistorter is called the "N-L type predistorter". The N-L type predistorter can only reduce the distortion caused by the purely nonlinear subsystem, and the desired output signal of the nonlinear physical system is approximately equal to the delayed input signal filtered by the purely linear subsystem, see the analysis in Section 3.2.2.

Furthermore, the publication [71] proposed the predistorter model for the nonlinear systems described by the input-output relation $y(n) = G[x(n)]H[x(n-1), y(n-1)] + F[x(n-1), y(n-1)]$, where $G[\cdot]$, $H[\cdot,\cdot]$ and $F[\cdot,\cdot]$ are causal, discrete-time and nonlinear operators and the inverse function $G^{-1}[\cdot]$ exists. The exact inverse of such system is then given by $z(n) = G^{-1}\left[\{u(n) - F[z(n-1), u(n-1)]\}/H[z(n-1), u(n-1)]\right]$. Also, the equalizer model for linearization of recursive polynomial systems is discussed in [72].

4.2.2. The learning architectures and adaptation algorithms

As described in previous sections, there are two kinds of learning architectures to estimate the coefficients of the predistorter: the DLA approach and the ILA approach. The ILA approach can still be classified into two categories: the ILA-I approach and the ILA-II approach.

Normally, the time domain adaptation algorithms using the DLA approach, such as the NFxLMS, NFxLMS-ISE, NFxRLS, NFxPEM and NFxPEM-ISE algorithms, need system identification or an initial subsystem estimate of the nonlinear physical system. Hence, the nonlinear physical system has to be identified or initially estimated before implementing the predistortion. Also, the NFxRLS algorithm can only be used in the predistortion of the nonlinear system, where the predistorter output is linear in its coefficients. The NFxLMS-ISE and NFxPEM-ISE algorithms are suitable for the predistortion of Wiener systems, since the initial subsystem estimate of the Wiener system is much simpler and faster than the accurate system identification.

System identification of the nonlinear physical system is also an important step in the ILA-II approach, since the estimates of the nonlinear physical system are used to construct the predistorter. Therefore, extra filters are needed for the system identification of the nonlinear physical system and also for the estimation of the inverse of the purely linear subsystem.

Compared to the DLA approach, the system identification or initial subsystem estimate of the nonlinear physical system is not necessary in the ILA-I approach. However, extra

training filter is needed. The coefficients of the training filter are estimated using adaptation algorithms, such as the RLS, KF and RPEM algorithms, and the predistorter is a copy of the training filter. Here, the RLS and KF algorithms can only be derived under the condition that the output of the training filter is linear in its coefficients.

The Spectral Magnitude Matching (SMM) method is an off-line, frequency domain adaptation algorithm using the DLA approach. Different from the time domain adaptation algorithms using the DLA approach, it does not require the system identification or initial subsystem estimates of the nonlinear physical system.

4.2.3. The computational complexity

Now let us take a look at the Computational Complexity (CC) of the adaptation algorithms.

The gradient vector required in the adaptation algorithms of the DLA and ILA-I approaches is obtained by differentiating the output signal of the nonlinear physical system (or training filter) w.r.t. the parameters of the predistorter (or training filter). After obtaining the gradient vector, the NFxLMS and NFxLMS-ISE algorithms in the DLA approach update the parameter vector directly using the gradient vector. Other algorithms, such as the NFxRLS, NFxPEM and NFxPEM-ISE algorithms in the DLA approach, and the RLS, KF and RPEM algorithms in the ILA-I approach, still need extra calculation to update the parameter vector. Hence, these algorithms have a little higher CC than the NFxLMS and NFxLMS-ISE algorithm in each adaptation step or each input sample. However, they usually converge much faster than the NFxLMS and NFxLMS-ISE algorithms.

In the ILA-II approach, the parameters of the predistorter are obtained by estimating the nonlinear physical system and the inverse of the purely linear subsystem using the LMS or RPEM algorithm. The RPEM algorithm has a little higher CC than the LMS algorithm, but with much faster convergence speed.

The SMM method is an off-line predistortion technique, in each adaptation iteration, the output signal of the nonlinear physical system corresponding to the input signal has to be evaluated and the spectral magnitude is calculated using Discrete Fourier Transform (DFT) for a number of short-time frames. Also, in the gradient calculations, previous steps are repeated for each perturbed parameter. Meanwhile, a matrix inversion is needed, see (2.123). Therefore, the CC of the SMM method is quite high in each adaptation iteration.

4.2.4. Summary

The characteristics of all adaptation algorithms are summarized in Table 4.2. System identification or initial subsystem estimates (SI/ISE) are usually required in the adaptation algorithms of the DLA approach, except the SMM method. System identification is also required in the ILA-II approach since the predistorter is constructed directly by using the identification results.

The CC per sample or iteration is given by comparing these algorithms in a specific application, *i.e.*, predistortion of Volterra systems. Here, we make the following assumptions: the Volterra system H is of order q with memory length m, and the number of the parameters is $\sum_{i=1}^{q} m^i$. The predistorter $C(n)$ is chosen as a pth-order Volterra system with memory n, and the number of the parameters is $\sum_{i=1}^{p} n^i$. The training filter $\tilde{C}(n)$ in the ILA-I approach and the Volterra model $\widehat{H}$ in the ILA-II approach are also pth-order Volterra systems with

Adaptation Algorithms	Needs SI/ISE	Extra Filters	Remarks
NFxLMS (DLA)	yes	no	-
NFxLMS-ISE (DLA)	yes	no	predistortion of Wiener systems
NFxRLS (DLA)	yes	no	predistorter is linear in its coefficients
NFxPEM (DLA)	yes	no	-
NFxPEM-ISE (DLA)	yes	no	predistortion of Wiener systems
RLS (ILA-I)	no	yes	training filter is linear in its coefficients
KF (ILA-I)	no	yes	-
RPEM (ILA-I)	no	yes	-
LMS (ILA-II)	yes	yes	linearization of weakly nonlinear systems
RPEM (ILA-II)	yes	yes	linearization of weakly nonlinear systems
SMM (DLA)	no	no	DFT, matrix inverse

Table 4.2 Summary of all adaptation algorithms.

memory n. The order of the linear FIR filter $\widehat{H}_L^{-1}$ in the ILA-II approach is n_l. For the SMM method, the length of the input signal is r divided into k short-time frames, and the DFT length is l.

The approximated addition per sample (+/Sample) and multiplication per sample (×/Sample) for the online adaptation algorithms are given in Table 4.3. The adaptation algorithms in the ILA-II approach are mainly used for system identification. For the adaptation algorithms using the DLA approach, we assume that the nonlinear physical system has been identified, otherwise the CC of the LMS or RPEM algorithm in the ILA-II approach should also be included.

The approximated addition per iteration (+/Ite) and multiplication per sample (×/Ite) for the off-line adaptation algorithms - the SMM method - are given in Table 4.4. The computation of the DFT and matrix inverse in the SMM method is not considered in this table.

In order to have a direct feeling of the CC comparison, let us consider the predistortion of a 2nd-order Volterra system and assume the follows: $p = q = 2$, $m = n = 4$, $n_l = 8$, $r = 10 \times 2^8$, $k = 10$ and $l = 2^8$. We can conclude the CC of the adaptation algorithms in Table 4.5 based on the simulation results in previous chapters. The overall CC represents the total additions and multiplications required for the convergence of the adaptation algorithm. From this table, we can see that the SMM method has the highest overall CC due to its high CC per iteration, even without considering the computation of the DFT and matrix inverse. The number of the input samples that we store in a buffer to perform this off-line adaptation algorithm is equal to r. In the online adaptation algorithms, the over all CC of the NFxLMS algorithm is also quite high due to its slow convergence. The overall CCs of the other online algorithms are similar, and the LMS algorithm has lower CC per iteration but with slower convergence, compared to the remaining algorithms. The

Adaptation Algorithms	**+/Sample**
NFxLMS (DLA)	$m\sum_{i=1}^{q-1} m^i + (m+1)\sum_{i=1}^{p} n^i$
NFxRLS (DLA)	$(m+2)\sum_{i=1}^{p} n^i + 5(\sum_{i=1}^{p} n^i)^2 + m\sum_{i=1}^{q-1} m^i$
NFxPEM (DLA)	$(m+2)\sum_{i=1}^{p} n^i + 5(\sum_{i=1}^{p} n^i)^2 + m\sum_{i=1}^{q-1} m^i$
RLS (ILA-I)	$5\sum_{i=1}^{p} n^i + 4(\sum_{i=1}^{p} n^i)^2$
KF (ILA-I)	$5\sum_{i=1}^{p} n^i + 5(\sum_{i=1}^{p} n^i)^2$
RPEM (ILA-I)	$5\sum_{i=1}^{p} n^i + 5(\sum_{i=1}^{p} n^i)^2$
LMS (ILA-II)	$2\sum_{i=1}^{p} n^i + 2n_l$
RPEM (ILA-II)	$3\sum_{i=1}^{p} n^i + 5(\sum_{i=1}^{p} n^i)^2 + 3n_l + 5n_l^2$

Adaptation Algorithms	**×/Sample**
NFxLMS (DLA)	$(m+1)\sum_{i=1}^{p}[i\times n^i] + m\sum_{i=1}^{q-1}[i\times m^i] + 3\sum_{i=1}^{p} n^i$
NFxRLS (DLA)	$5(\sum_{i=1}^{p} n^i)^2 + (m+1)\sum_{i=1}^{p}[i\times n^i] + m\sum_{i=1}^{q-1}[i\times m^i]$
NFxPEM (DLA)	$6(\sum_{i=1}^{p} n^i)^2 + (m+1)\sum_{i=1}^{p}[i\times n^i] + m\sum_{i=1}^{q-1}[i\times m^i]$
RLS (ILA-I)	$3\sum_{i=1}^{p} n^i + 5(\sum_{i=1}^{p} n^i)^2 + 2\sum_{i=1}^{p}[i\times n^i]$
KF (ILA-I)	$3\sum_{i=1}^{p} n^i + 5(\sum_{i=1}^{p} n^i)^2 + 2\sum_{i=1}^{p}[i\times n^i]$
RPEM (ILA-I)	$3\sum_{i=1}^{p} n^i + 6(\sum_{i=1}^{p} n^i)^2 + 2\sum_{i=1}^{p}[i\times n^i]$
LMS (ILA-II)	$2\sum_{i=1}^{p}[i\times n^i] + 2\sum_{i=1}^{p} n^i + 2n_l$
RPEM (ILA-II)	$3\sum_{i=1}^{p} n^i + 6(\sum_{i=1}^{p} n^i)^2 + 2\sum_{i=1}^{p}[i\times n^i] + 4n_l + 6n_l^2$

Table 4.3 CC per sample of the online adaptation algorithms for predistortion of Volterra systems.

Adaptation Algorithms	**+/Ite**
SMM (DLA)	$(kl+k+r)(\sum_{i=1}^{p} n^i)^2 + (2kl+k+r)\sum_{i=1}^{p} n^i + r\sum_{i=1}^{q} m^i + kl$

Adaptation Algorithms	**×/Ite**
SMM (DLA)	$r\sum_{i=1}^{p}[i\times n^i] + r\sum_{i=1}^{q}[i\times m^i] + (kl+l)(\sum_{i=1}^{p} n^i)^2$ $+(2kl+1+r\sum_{i=1}^{p}[i\times n^i])\sum_{i=1}^{p} n^i$

Table 4.4 CC per iteration of the off-line adaptation algorithm for predistortion of Volterra systems.

Adaptation Algorithms	**+/Sample**	**×/Sample**	**Convergence (Samples)**	**Overall CC**
NFxLMS (DLA)	180	384	$> 2 \times 10^4$	$> 1.13 \times 10^7$
NFxRLS (DLA)	2200	2324	800	3.62×10^6
NFxPEM (DLA)	2200	2724	800	3.94×10^6
RLS (ILA-I)	1700	2132	300	1.15×10^6
KF (ILA-I)	2100	2132	300	1.27×10^6
RPEM (ILA-I)	2100	2532	300	1.39×10^6
LMS (ILA-II)	56	128	10^4	1.84×10^6
RPEM (ILA-II)	2404	2948	1000	5.36×10^6

Adaptation Algorithms	**+/Ite**	**×/Ite**	**Convergence (Iterations)**	**Overall CC**
SMM (DLA)	2.26×10^6	3.26×10^6	20	1.10×10^8

Table 4.5 CC of the adaptation algorithms for predistortion of a 2nd-order Volterra system.

Chapter 5

Exemplary Applications

In this chapter, several examples of adaptive predistortion applications will be given. Also, some nonlinear models used in practical communication systems, such as the parallel Wiener-type model and the memory polynomial model, are considered. Adaptive predistortion of these nonlinear models is discussed, including the predistorter models, the learning architectures and adaptation algorithms.

5.1. Predistortion of Parallel Wiener-Type Systems

Recently, the parallel Wiener-type model have been proposed in [73, 74, 75] for modeling the Line Driver (LD) in DSL systems. The suggested model can be considered as a generalization of the classic Wiener system. In this section, different nonlinear models for the predistorter are first investigated, then the learning architectures and adaptation algorithms for estimating the coefficients of the predistorter are discussed. A comparison by simulation studies is given at the end of this section.

This section is based on the publications [46, 45, 34], which are edited and refined in order to fit the current style of the thesis.

5.1.1. The predistorter models and learning architectures

The parallel Wiener-type system H has the structure as shown in Fig. 5.1. The output signal $z(n)$ is given by

$$z(n) = \sum_{k=1}^{q} \big(H_k(\mathsf{z}^{-1})y(n)\big)^k = \sum_{k=1}^{q} \big(\boldsymbol{h}_k^T \boldsymbol{y}(n)\big)^k \tag{5.1}$$

where $H_k(\mathsf{z}^{-1})$ denotes the FIR filter in the kth branch and z^{-1} is the delay operator. $\boldsymbol{h}_k$ is the parameter vector of $H_k(\mathsf{z}^{-1})$ defined as

$$\boldsymbol{h}_k = \begin{pmatrix} h_{k,0} & \cdots & h_{k,M-1} \end{pmatrix}^T \tag{5.2}$$

where M is the memory length and the corresponding input vector $\boldsymbol{y}(n)$ is

$$\boldsymbol{y}(n) = \begin{pmatrix} y(n) & \cdots & y(n-M+1) \end{pmatrix}^T. \tag{5.3}$$

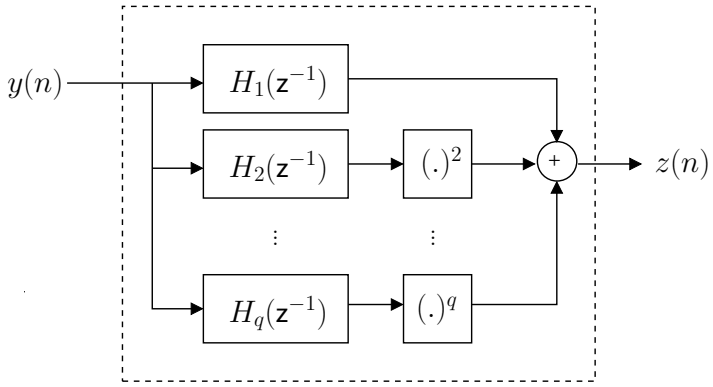

Figure 5.1 The parallel Wiener-type system.

Since the parallel Wiener-type system is a simplified factorisable Volterra system [74], the predistorter can be modeled using Volterra series. Also, considering that the parallel Wiener-type system can be divided into a purely linear subsystem $\boldsymbol{h}_1^T \boldsymbol{y}(n)$ and a purely nonlinear subsystem $\sum_{k=2}^{q} \left(\boldsymbol{h}_k^T \boldsymbol{y}(n)\right)^k$, we can construct the predistorter using the N-L type model as in Fig. 5.2, including the estimate of the purely nonlinear subsystem and the estimate of the inverse of the purely linear subsystem, see Section 4.2.1. Here, $\widehat{H}_k(n, \mathsf{z}^{-1})$ represents the FIR filter estimating $H_k(\mathsf{z}^{-1})$, and $\widehat{H}_1^{-1}(n, \mathsf{z}^{-1})$ represents the FIR filter estimating the inverse of $H_1(\mathsf{z}^{-1})$. τ' is the time delay satisfying $H_1^{-1}(\mathsf{z}^{-1})H_1(\mathsf{z}^{-1}) = \mathsf{z}^{-\tau'}$ ($\mathsf{z}^{-\tau'}x(n) = x(n-\tau')$).

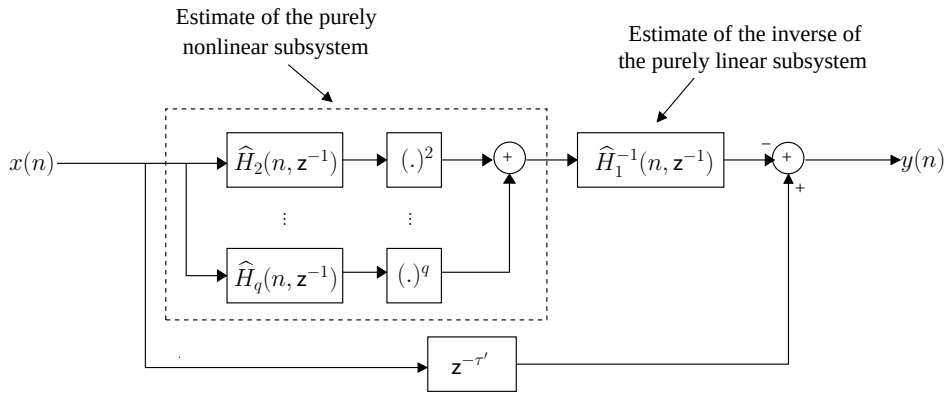

Figure 5.2 The N-L type predistorter model for parallel Wiener-type systems.

5.1.2. Adaptation algorithms using the DLA approach

If the parallel Wiener-type system is known or has been identified, the parameters of the predistorter can be estimated using the DLA approach. The Nonlinear Filtered-x Least Mean Squares (NFxLMS) and Nonlinear Filtered-x Prediction Error Method (NFxPEM) algorithms can be derived.

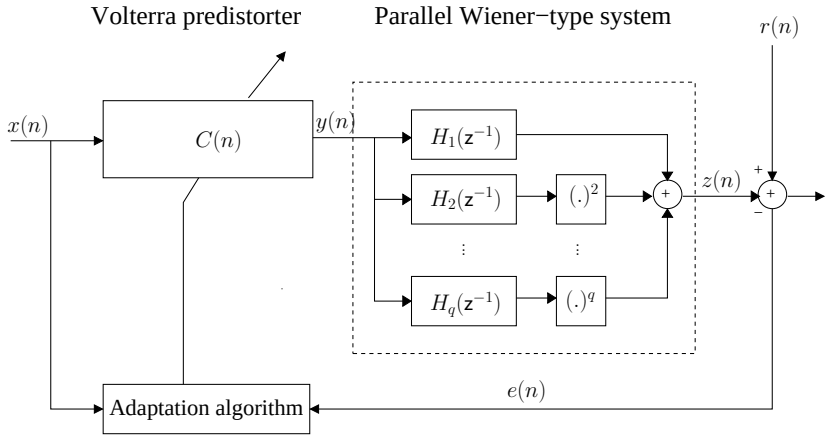

Figure 5.3 The DLA approach using a Volterra predistorter.

The DLA approach using Volterra predistorter

Assuming that the predistorter is a pth-order Volterra system $C(n)$, the relation between the input and output of the predistorter is given by

$$\begin{aligned} y(n) &= \boldsymbol{c}^T(n)\boldsymbol{x}(n) = \sum_{k=1}^{p} \boldsymbol{c}_k^T(n)\boldsymbol{x}_k(n) \\ &= \sum_{k=1}^{p} \left(\sum_{i_1=0}^{\widehat{M}-1} \cdots \sum_{i_k=0}^{\widehat{M}-1} c_k(i_1, \cdots, i_k; n) x(n-i_1) \cdots x(n-i_k) \right) \end{aligned} \tag{5.4}$$

where $\widehat{M}$ is the memory length. The kernel vector $\boldsymbol{c}(n)$ and the corresponding input vector $\boldsymbol{x}(n)$ are defined similarly as in Section 2.2.

The error signal $e(n)$ is

$$e(n) = r(n) - z(n) \tag{5.5}$$

where $r(n)$ is the reference signal defined as

$$r(n) = H_1(\mathsf{z}^{-1})x(n). \tag{5.6}$$

Note that the reference signal is defined as $H_1(\mathsf{z}^{-1})x(n)$ but not $x(n-\tau)$ as in previous chapters, since it is the ideal output of the nonlinear physical system using the N-L type predistorter and we would like to compare the performances of different predistorter models on reducing the distrotion caused by the nonlinear subsystem.

The NFxLMS algorithm is obtained by applying the stochastic gradient algorithm [56, 63], as

$$\boldsymbol{c}(n+1) = \boldsymbol{c}(n) - \frac{\mu}{2}\boldsymbol{\Delta}_{\boldsymbol{c}}^T(n) \tag{5.7}$$

where μ is the *step-size*. The gradient vector $\boldsymbol{\Delta}_{\boldsymbol{c}}(n)$ is defined as

$$\boldsymbol{\Delta}_{\boldsymbol{c}}(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)} e^2(n) = -2e(n)\boldsymbol{\nabla}_{\boldsymbol{c}(n)} z(n). \tag{5.8}$$

Straightforward derivation gives

$$\boldsymbol{\nabla}_{\boldsymbol{c}(n)} z(n) \approx \sum_{k=1}^{q} s_k(n) \boldsymbol{h}_k^T \begin{pmatrix} \boldsymbol{x}^T(n) \\ \vdots \\ \boldsymbol{x}^T(n-M+1) \end{pmatrix} \tag{5.9}$$

where

$$s_k(n) = k \left(\boldsymbol{h}_k^T \boldsymbol{y}(n) \right)^{k-1}. \tag{5.10}$$

Hence, the gradient vector $\boldsymbol{\Delta}_{\boldsymbol{c}}(n)$ can be written as

$$\boldsymbol{\Delta}_{\boldsymbol{c}}(n) = -2e(n) \sum_{k=1}^{q} s_k(n) \boldsymbol{h}_k^T \begin{pmatrix} \boldsymbol{x}^T(n) \\ \vdots \\ \boldsymbol{x}^T(n-M+1) \end{pmatrix}. \tag{5.11}$$

Similarly, the NFxPEM algorithm is derived by the minimization of the cost function [57]

$$V(\boldsymbol{c}) = \lim_{\mathcal{N} \to \infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n) \right] \tag{5.12}$$

where $e(n)$ is the prediction error defined as

$$e(n) = r(n) - z(n). \tag{5.13}$$

The formulation of the NFxPEM algorithm requires the negative gradient of $e(n)$ w.r.t. $\boldsymbol{c}(n)$ which is defined as

$$\boldsymbol{\varphi}^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{c}(n)} e(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)} z(n) \approx \sum_{k=1}^{q} s_k(n) \boldsymbol{h}_k^T \begin{pmatrix} \boldsymbol{x}^T(n) \\ \vdots \\ \boldsymbol{x}^T(n-M+1) \end{pmatrix}. \tag{5.14}$$

Then, the NFxPEM algorithm follows the same formulation as in Chapter 2 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}(0)$.

The DLA approach using the N-L type predistorter

Since the parallel Wiener-type system is assumed known or has been identified, the purely nonlinear subsystem required in the N-L type predistorter is known. Therefore, it remains to estimate the inverse of the purely linear subsystem, see Fig. 5.4. This inverse is modeled as a linear FIR filter denoted as $C_l(n, \mathsf{z}^{-1})$ with memory length M_l and parameter vector defined as

$$\boldsymbol{c}_l(n) = \begin{pmatrix} c_1(n) & \cdots & c_{M_l}(n) \end{pmatrix}^T. \tag{5.15}$$

The output signal of the predistorter $y(n)$ is given by

$$y(n) = x(n-\tau') - C_l(n, \mathsf{z}^{-1}) x_2(n) = x(n-\tau') - \boldsymbol{c}_l^T(n) \boldsymbol{x}_2(n) \tag{5.16}$$

where

$$\boldsymbol{x}_2(n) = \begin{pmatrix} x_2(n) & \cdots & x_2(n-M_l+1) \end{pmatrix}^T. \tag{5.17}$$

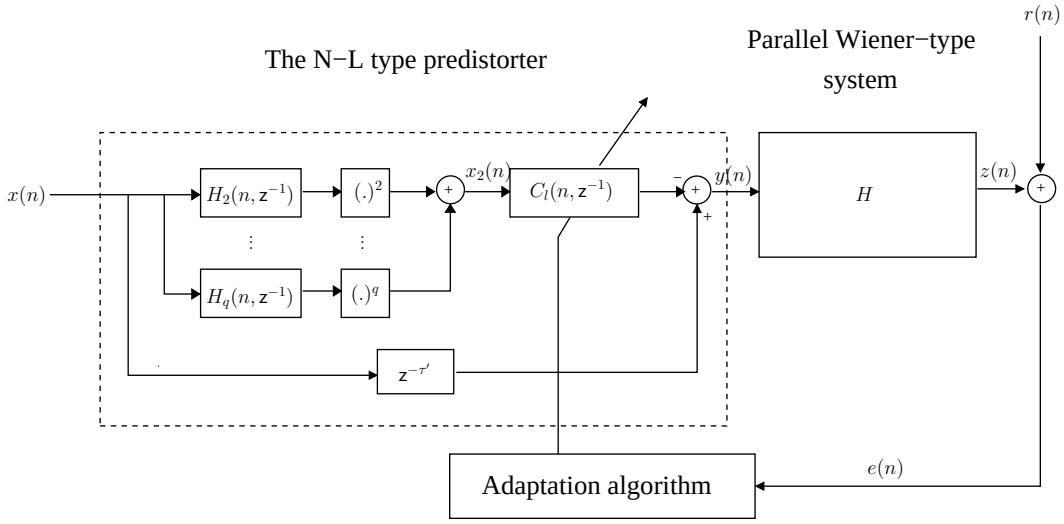

Figure 5.4 The DLA approach using the N-L type predistorter.

The intermediate signal $x_2(n)$ is obtained by

$$x_2(n) = \sum_{k=2}^{q} \left(\boldsymbol{h}_k^T(n) \boldsymbol{x}(n) \right)^k \tag{5.18}$$

where

$$\boldsymbol{x}(n) = \begin{pmatrix} x(n) & \cdots & x(n-M+1) \end{pmatrix}^T. \tag{5.19}$$

The NFxLMS algorithm is obtained by applying the stochastic gradient algorithm [56,63], as

$$\boldsymbol{c}_l(n+1) = \boldsymbol{c}_l(n) - \frac{\mu_l}{2} \boldsymbol{\Delta}_l^T(n) \tag{5.20}$$

and in this case, the gradient vector $\boldsymbol{\Delta}_l(n)$ is given as

$$\boldsymbol{\Delta}_l(n) = 2e(n) \sum_{k=1}^{q} s_k(n) \boldsymbol{h}_k^T \begin{pmatrix} \boldsymbol{x}_2^T(n) \\ \vdots \\ \boldsymbol{x}_2^T(n-M+1) \end{pmatrix} \tag{5.21}$$

where $\boldsymbol{x}_2(n)$ is defined in (5.17) - (5.18) and $s_k(n)$ is defined as

$$s_k(n) = k \left(\boldsymbol{h}_k^T \boldsymbol{y}(n) \right)^{k-1}. \tag{5.22}$$

The NFxPEM algorithm can also be derived similarly as the Volterra predistorter by minimizing the cost function

$$V(\boldsymbol{c}_l) = \lim_{\mathcal{N} \to \infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n) \right]. \tag{5.23}$$

where $e(n)$ is the prediction error defined as

$$e(n) = r(n) - z(n). \tag{5.24}$$

The required negative gradient of the error signal $e(n)$ w.r.t. $\boldsymbol{c}_l(n)$ is given by

$$\boldsymbol{\varphi}_l^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{c}_l(n)} e(n) = \boldsymbol{\nabla}_{\boldsymbol{c}_l(n)} z(n) = -\sum_{k=1}^{q} s_k(n) \boldsymbol{h}_k^T \begin{pmatrix} \boldsymbol{x}_2^T(n) \\ \vdots \\ \boldsymbol{x}_2^T(n-M+1) \end{pmatrix} \tag{5.25}$$

where $s_k(n)$ and $\boldsymbol{x}_2(n)$ are defined as before. Then, the NFxPEM algorithm follows the same formulation as in Chapter 2 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}_l(0)$.

5.1.3. Adaptation algorithms using the ILA approach

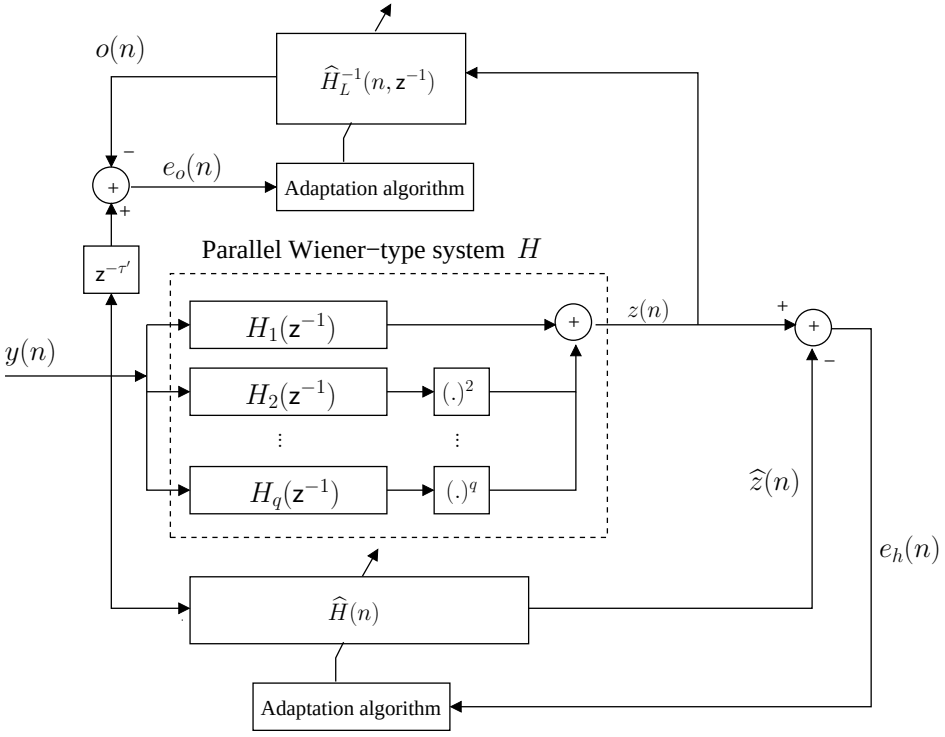

Figure 5.5 System identification in the ILA-II approach.

The system identification scheme for the parallel Wiener-type system and the inverse of the purely linear subsystem is given in Fig. 5.5. Another parallel Wiener-type system $\widehat{H}(n)$ with similar structure is used to identify H. The output signal $\widehat{z}(n)$ can be written as

$$\widehat{z}(n) = \sum_{k=1}^{q} \left(\widehat{\boldsymbol{h}}_k^T(n) \widehat{\boldsymbol{y}}_k(n) \right)^k . \tag{5.26}$$

The parameter vector $\widehat{\boldsymbol{h}}_k(n)$ is defined as

$$\widehat{\boldsymbol{h}}_k(n) = \begin{pmatrix} \widehat{h}_{k,0}(n) & \cdots & \widehat{h}_{k,\widehat{M}_k-1}(n) \end{pmatrix}^T . \tag{5.27}$$

where $\widehat{M}_k$ is the memory length. The corresponding input vector $\widehat{\boldsymbol{y}}_k(n)$ is

$$\widehat{\boldsymbol{y}}_k(n) = \begin{pmatrix} y(n) & \cdots & y(n-\widehat{M}_k+1) \end{pmatrix}^T. \tag{5.28}$$

Let us define the parameter vector $\widehat{\boldsymbol{h}}(n)$ of $\widehat{H}(n)$ as

$$\widehat{\boldsymbol{h}}(n) = \begin{pmatrix} \widehat{\boldsymbol{h}}_1^T(n) & \cdots & \widehat{\boldsymbol{h}}_q^T(n) \end{pmatrix}^T. \tag{5.29}$$

In order to estimate the parameter vector $\widehat{\boldsymbol{h}}$, the RPEM algorithm is derived by minimizing the cost function [57]

$$V(\widehat{\boldsymbol{h}}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e_h^2(n)\right] \tag{5.30}$$

where $e_h(n)$ is the prediction error defined as

$$e_h(n) = z(n) - \widehat{z}(n). \tag{5.31}$$

The negative gradient of $e_h(n)$) w.r.t. $\widehat{\boldsymbol{h}}(n)$ is given as

$$\boldsymbol{\varphi}_h^T(n) = -\boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}(n)} e_h(n) = \boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}(n)} \widehat{z}(n). \tag{5.32}$$

Straightforward derivation gives

$$\boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}(n)} \widehat{z}(n) = \sum_{k=1}^{q} k \left(\widehat{\boldsymbol{h}}_k^T(n)\widehat{\boldsymbol{y}}_k(n)\right)^{k-1} \widehat{\boldsymbol{y}}_k^T(n). \tag{5.33}$$

Then, the RPEM algorithm follows the same formulation as in Chapter 3 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}_h(0)$.

The inverse of the purely linear subsystem is required to model the predistorter and it is estimated using an adaptive FIR filter $\widehat{H}_L^{-1}(n, \mathsf{z}^{-1})$ with K-tap memory. The output signal of this filter can be written as

$$o(n) = \widehat{H}_L^{-1}(n, \mathsf{z}^{-1}) z(n) = \left[\widehat{\boldsymbol{h}}_L^{-1}(n)\right]^T \boldsymbol{z}(n) \tag{5.34}$$

where the parameter vector $\widehat{\boldsymbol{h}}_L^{-1}(n)$ is defined as

$$\widehat{\boldsymbol{h}}_L^{-1}(n) = \begin{pmatrix} \widehat{h}_0^{-1}(n) & \cdots & \widehat{h}_{K-1}^{-1}(n) \end{pmatrix}^T \tag{5.35}$$

and the corresponding input vectors $\boldsymbol{z}(n)$ is

$$\boldsymbol{z}(n) = \begin{pmatrix} z(n) & \cdots & z(n-K+1) \end{pmatrix}^T. \tag{5.36}$$

The gradient vector $\boldsymbol{\varphi}_o(n)$ required in the RPEM algorithm is given by

$$\boldsymbol{\varphi}_o^T(n) = -\boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}_L^{-1}(n)} e_o(n) = \boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}_L^{-1}(n)} o(n) = \boldsymbol{z}^T(n). \tag{5.37}$$

and the RPEM algorithm can follow the same formulation as in Chapter 3 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}_s(0)$.

Predistorter	**Adaptive Algorithms**	μ	μ_l	$\boldsymbol{P}(0)$	$\boldsymbol{P}_l(0)$	$\boldsymbol{P}_h(0)$	$\boldsymbol{P}_s(0)$
Volterra	NFxLMS (DLA)	10^{-3}	-	-	-	-	-
Volterra	NFxPEM (DLA)	-	-	$10^{-3}\boldsymbol{I}$	-	-	-
N-L type	NFxLMS (DLA)	-	0.9	-	-	-	-
N-L type	NFxPEM (DLA)	-	-	-	$10^{2}\boldsymbol{I}$	-	-
N-L type	RPEM (ILA-II)	-	-	-	-	$10^{-2}\boldsymbol{I}$	$10^{-2}\boldsymbol{I}$

Table 5.1 Initialization of all adaptation algorithms for predistortion of the parallel Wiener-type system.

5.1.4. Simulation results

In the simulation study, the following parallel Wiener-type system H is considered:

$$z(n) = \boldsymbol{h}_1^T \boldsymbol{y}_1(n) + (\boldsymbol{h}_3^T \boldsymbol{y}_3(n))^3 \tag{5.38}$$

$$\begin{aligned} \boldsymbol{h}_1 &= \begin{pmatrix} 0.5625 & 0.4810 & 0.1124 & -0.1669 \end{pmatrix}^T \\ \boldsymbol{h}_3 &= \begin{pmatrix} 0.03572 & 0.07796 & 0.06063 & 0.01388 \end{pmatrix}^T. \end{aligned} \tag{5.39}$$

In the DLA approach, the Volterra predistorter is assumed to be a known 3rd-order Volterra filter with memory length $\widehat{M} = 4$, and the memory length of $C_l(n, \mathsf{z}^{-1})$ in the N-L type predistorter is assumed to be $M_l = 6$. In the ILA-II approach, the parameter vectors of $\widehat{H}(n)$ and $\widehat{H}_L^{-1}(n, \mathsf{z}^{-1})$ are chosen as

$$\begin{aligned} \widehat{\boldsymbol{h}}(n) &= \begin{pmatrix} \widehat{\boldsymbol{h}}_1^T(n) & \widehat{\boldsymbol{h}}_3^T(n) \end{pmatrix}^T \\ \widehat{\boldsymbol{h}}_1(n) &= \begin{pmatrix} h_{1,0}(n) & \cdots & h_{1,3}(n) \end{pmatrix}^T \\ \widehat{\boldsymbol{h}}_3(n) &= \begin{pmatrix} h_{3,0}(n) & \cdots & h_{3,3}(n) \end{pmatrix}^T \\ \widehat{\boldsymbol{h}}_L^{-1}(n) &= \begin{pmatrix} h_0^{-1}(n) & \cdots & h_5^{-1}(n) \end{pmatrix}^T. \end{aligned} \tag{5.40}$$

The number of independent experiments is 100. In each experiment, the input signal to the predistorter is chosen to be a random signal with uniform distribution over $(-3, 3)$ and data length of 4×10^4. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{6}$. In the ILA-II approach, the predistorter starts to copy the coefficients after 3.2×10^4 input samples. The adaptation algorithms are initialized as in Table 5.1, where $\boldsymbol{I}$ denotes the identity matrix. Also, the design parameters λ_0 and $\lambda(0)$ in the NFxPEM and RPEM algorithms are chosen as 0.99 and 0.95, respectively.

The Mean Square Distortion (MSD) value of the parallel Wiener-type system without predistorter is about -42 dB. The MSD of different approaches and algorithms is given in Table. 5.2. From the table, we can see that Volterra predistorter using the DLA approach and NFxPEM algorithm achieves the best MSD performance. Compared to the Volterra predistorter, the N-L type predistorter using the DLA approach and NFxPEM algorithm

Predistorter	Adaptive Algorithms	MSD (dB)
Volterra	NFxLMS (DLA)	−71
Volterra	NFxPEM (DLA)	−79
N-L type	NFxLMS (DLA)	−63
N-L type	NFxPEM (DLA)	−76
N-L type	RPEM (ILA-II)	−72

Table 5.2 Comparison of the MSD for predistortion of a parallel Wiener-type system.

achieves the second best performance but with much lower computational complexity since it only needs to update the coefficients of a linear FIR filter in each iteration.

Figure 5.6 shows mean power spectral densities (PSDs) of the output signals of the parallel Wiener-type system without and with the predistorter after 4×10^4 samples. From this figure, we can see that the proposed predistortion techniques can effectively reduce the spectral regrowth. The N-L type predistorter using the ILA-II approach and RPEM algorithm achieves the best performance. The Volterra predistorter using the DLA approach and NFxPEM algorithm can achieve similar performance but without using additional filters. The N-L type predistorter using the DLA approach and NFxPEM algorithm can achieve the second best performance but with lower computational complexity compared to the Volterra predistorter.

5.2. Predistortion of Memory Polynomial Systems

In most cases, the nonlinear behavior of Power Amplifiers (PAs) in wireless communication systems can be modeled using either Volterra series or neural networks [76]. However, the large number of coefficients is the drawback of these two models. Another special case of the Volterra model is the memory polynomial model proposed in [22]. Compared to the Volterra model, the memory polynomial model can capture memory effects with much fewer coefficients.

Predistortion of memory polynomial systems is first considered in [22]. In [77, 78], the predistorter is modeled as another memory polynomial system and the coefficients of the predistorter are estimated based on the ILA-I approach. In this learning architecture, the predistorter is a copy of a training filter. The training filter is connected in cascade with the nonlinear physical system and its coefficients are evaluated using the Least Squares (LS) method.

In this section, the nonlinear models for the predistorter are first investigated, then the learning architectures and adaptation algorithms for estimating the coefficients of the predistorter are discussed. A comparison by simulation studies is given at the end of this section.

This section is based on the publications [44, 34], which are edited and refined in order to fit the current style of the thesis.

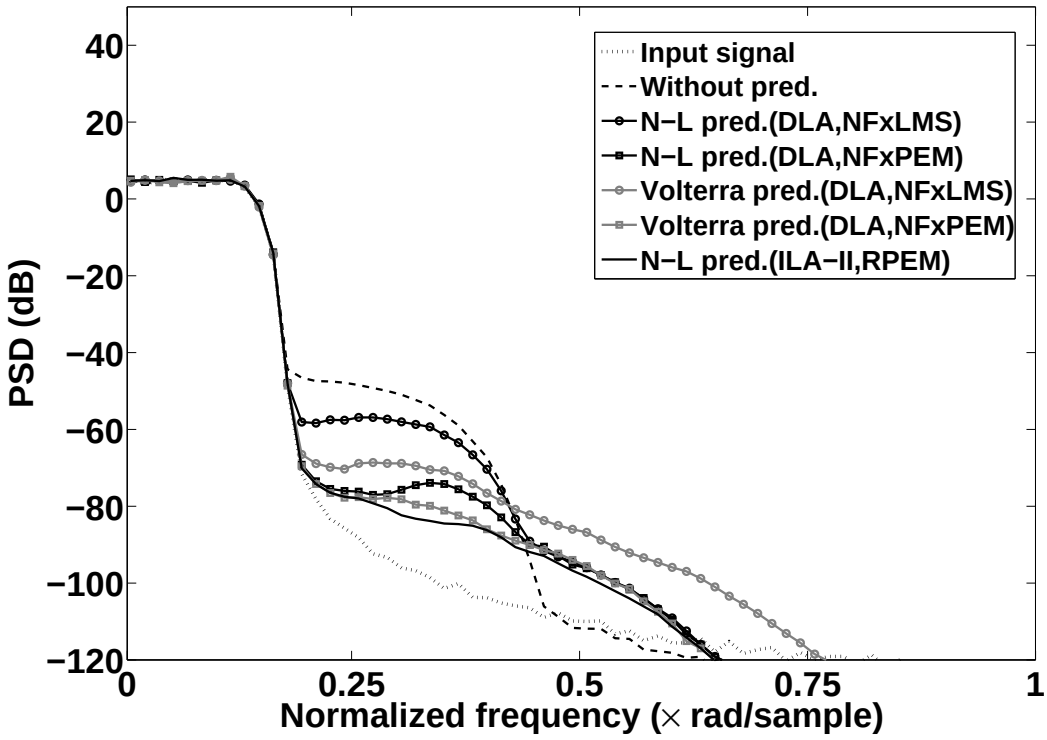

Figure 5.6 Mean PSDs comparison for predistortion of a parallel Wiener-type system.

5.2.1. The predistorter models

The qth-order memory polynomial system H is shown in Fig. 5.7. The output signal $z(n)$ is given by

$$z(n) = \sum_{k=1}^{q} H_k(\mathsf{z}^{-1}) y_k(n) = \sum_{k=1}^{q} \boldsymbol{h}_k^T \boldsymbol{y}_k(n) \tag{5.41}$$

where $H_k(\mathsf{z}^{-1})$ is the FIR filter in the kth branch. The parameter vector $\boldsymbol{h}_k$ of $H_k(\mathsf{z}^{-1})$ is defined as

$$\boldsymbol{h}_k = \begin{pmatrix} h_{k,0} & \cdots & h_{k,M-1} \end{pmatrix}^T. \tag{5.42}$$

where M is the memory length. The corresponding input vector $\boldsymbol{y}_k(n)$ is

$$\boldsymbol{y}_k(n) = \begin{pmatrix} y(n)^k \\ \vdots \\ y(n-M+1)^k \end{pmatrix}. \tag{5.43}$$

In many cases, especially in bandpass systems, odd-order nonlinearities usually dominate the performance [79] and the memory polynomial system is modeled only containing odd-order terms, which is $k = 1, 3, 5, ...$ and q is an odd number. In this section, we will also consider

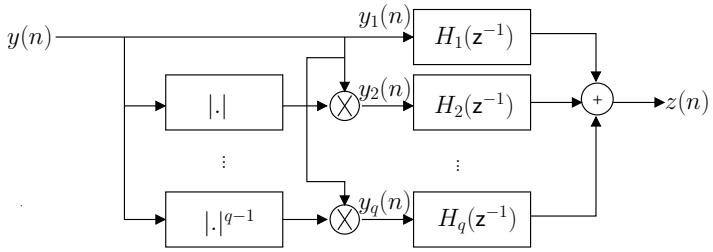

Figure 5.7 The memory polynomial system.

the nonlinear physical system as a memory polynomial system containing odd-order terms, even though the nonlinear physical system is a baseband system.

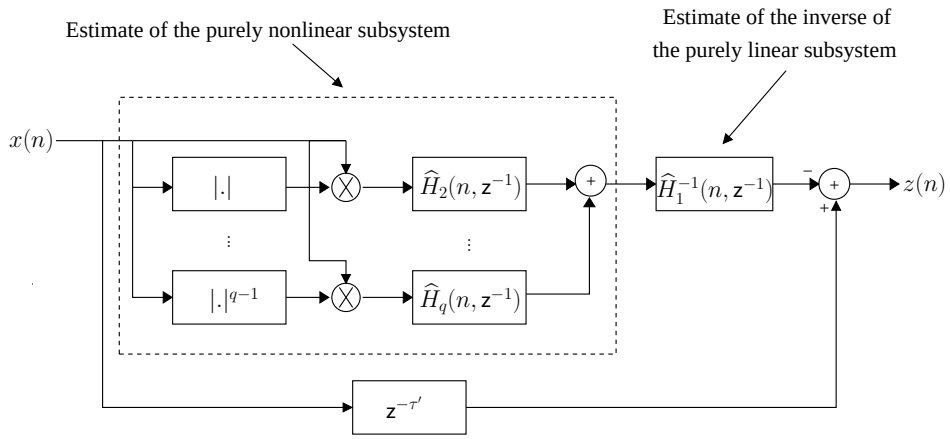

Figure 5.8 The N-L type predistorter model for memory polynomial systems.

As proposed in [77], the predistorter of the memory polynomial system can be modeled as another memory polynomial system. Also, considering that the memory polynomial system can be divided into a purely linear subsystem $\boldsymbol{h}_1^T\boldsymbol{y}_1(n)$ and a purely nonlinear subsystem $\sum_{k=2}^{q}\boldsymbol{h}_k^T\boldsymbol{y}_k(n)$, we can construct the predistorter using the N-L type model including the estimate of the purely nonlinear subsystem and the estimate of the inverse of the purely linear subsystem as in Fig. 5.8. Here, $\widehat{H}_k(n,\mathsf{z}^{-1})$ represents the FIR filter estimating $H_k(\mathsf{z}^{-1})$, and $\widehat{H}_1^{-1}(n,\mathsf{z}^{-1})$ represents the FIR filter estimating the inverse of $H_1(\mathsf{z}^{-1})$.

5.2.2. Adaptation algorithms using the DLA approach

If the memory polynomial system is known or has been identified, the parameters of the predistorter can be estimated using the DLA approach. The NFxLMS and NFxPEM algorithms can be derived.

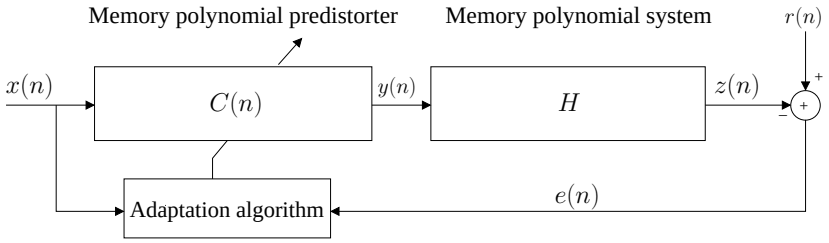

Figure 5.9 The DLA approach using a memory polynomial predistorter.

The DLA approach using the memory polynomial predistorter

The predistorter can be modeled as a pth-order memory polynomial system $C(n)$ and its output is given by

$$y(n) = \sum_{k=1}^{p} C_k(n, \mathsf{z}^{-1}) y_k(n) = \sum_{k=1}^{p} \boldsymbol{c}_k^T(n) \boldsymbol{x}_k(n). \tag{5.44}$$

The parameter vector $\boldsymbol{c}_k(n)$ is defined as

$$\boldsymbol{c}_k(n) = \begin{pmatrix} c_{k,0}(n) & \cdots & c_{k,\widehat{M}-1}(n) \end{pmatrix}^T. \tag{5.45}$$

where $\widehat{M}$ is the memory length. The corresponding input vector $\boldsymbol{x}_k(n)$ is

$$\boldsymbol{x}_k(n) = \begin{pmatrix} x(n)^k \\ \vdots \\ x(n-\widehat{M}+1)^k \end{pmatrix}. \tag{5.46}$$

The error signal $e(n)$ is

$$e(n) = r(n) - z(n) \tag{5.47}$$

where $r(n)$ is the reference signal defined as

$$r(n) = H_1(\mathsf{z}^{-1}) x(n). \tag{5.48}$$

The NFxLMS algorithm is obtained by applying the stochastic gradient algorithm [56, 63], as

$$\boldsymbol{c}(n+1) = \boldsymbol{c}(n) - \frac{\mu}{2} \boldsymbol{\Delta}_{\boldsymbol{c}}^T(n). \tag{5.49}$$

The gradient vector $\boldsymbol{\Delta}_{\boldsymbol{c}}(n)$ is defined as

$$\boldsymbol{\Delta}_{\boldsymbol{c}}(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)} e^2(n) = -2e(n) \boldsymbol{\nabla}_{\boldsymbol{c}(n)} z(n). \tag{5.50}$$

Straightforward derivation gives

$$\boldsymbol{\nabla}_{\boldsymbol{c}(n)} z(n) = \sum_{k=1}^{q} \sum_{m=0}^{M-1} q h_{q,m} y(n-m)^{q-1} \begin{pmatrix} \boldsymbol{x}_1(n-m) \\ \vdots \\ \boldsymbol{x}_p(n-m) \end{pmatrix}^T. \tag{5.51}$$

Hence, the gradient vector $\boldsymbol{\Delta}_{\boldsymbol{c}}(n)$ can be written as

$$\boldsymbol{\Delta}_{\boldsymbol{c}}(n) = -2e(n)\sum_{k=1}^{q}\sum_{m=0}^{M-1} qh_{q,m}y(n-m)^{q-1}\begin{pmatrix}\boldsymbol{x}_1(n-m)\\ \vdots \\ \boldsymbol{x}_p(n-m)\end{pmatrix}^T. \tag{5.52}$$

Similarly, the NFxPEM algorithm is derived by the minimization of the cost function

$$V(\boldsymbol{c}) = \lim_{\mathcal{N}\to\infty}\frac{1}{\mathcal{N}}\sum_{n=1}^{\mathcal{N}}\mathrm{E}\left[e^2(n)\right] \tag{5.53}$$

where $e(n)$ is the prediction error defined as

$$e(n) = r(n) - z(n). \tag{5.54}$$

The negative gradient of $e(n)$ w.r.t. $\boldsymbol{c}(n)$ is given by

$$\boldsymbol{\varphi}^T(n) = -\boldsymbol{\nabla}_{\boldsymbol{c}(n)}e(n) = \boldsymbol{\nabla}_{\boldsymbol{c}(n)}z(n) \approx \sum_{k=1}^{q}\sum_{m=0}^{M-1} qh_{q,m}|y(n-m)|^{q-1}\begin{pmatrix}\boldsymbol{x}_1(n-m)\\ \vdots \\ \boldsymbol{x}_p(n-m)\end{pmatrix}^T. \tag{5.55}$$

Then, the NFxPEM algorithm follows the same formulation as in Chapter 2 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}(0)$.

The DLA approach using the N-L type predistorter

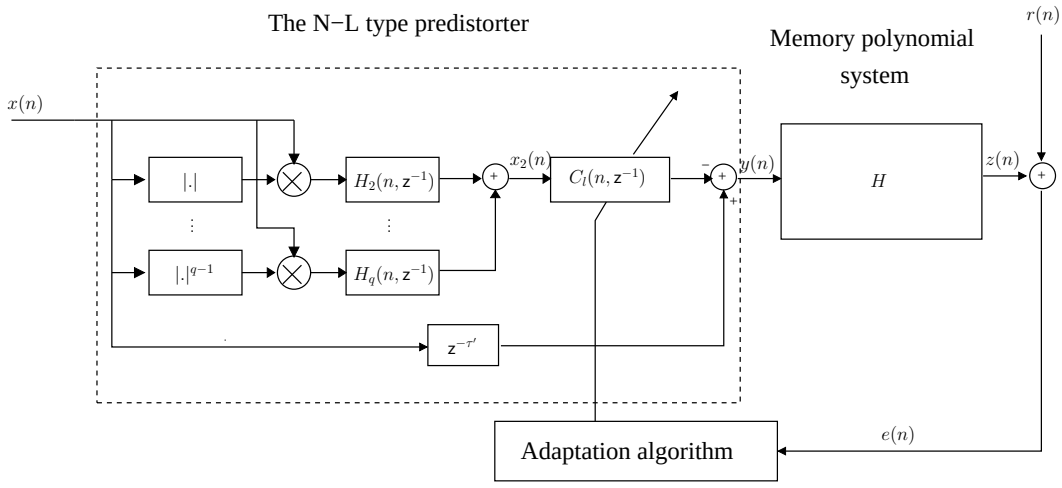

Figure 5.10 The DLA approach using the N-L type predistorter.

Since the memory polynomial system is known or has been identified, the purely nonlinear subsystem required in the N-L type predistorter is known. Therefore, it remains to estimate the inverse of the purely linear subsystem adaptively, see Fig. 5.10. This inverse is modeled

as a linear FIR filter denoted as $C_l(n, \mathsf{z}^{-1})$ with memory length M_l and parameter vector defined as

$$\boldsymbol{c}_l(n) = \begin{pmatrix} c_1(n) & \cdots & c_{M_l}(n) \end{pmatrix}^T. \tag{5.56}$$

The output signal of the predistorter $y(n)$ is given by

$$y(n) = x(n-\tau') - C_l(n, \mathsf{z}^{-1})x_2(n) = x(n-\tau') - \boldsymbol{c}_l^T(n)\boldsymbol{x}_2(n) \tag{5.57}$$

where

$$\boldsymbol{x}_2(n) = \begin{pmatrix} x_2(n) & \cdots & x_2(n-M_l+1) \end{pmatrix}^T. \tag{5.58}$$

The intermediate signal $x_2(n)$ is obtained by

$$x_2(n) = \sum_{k=2}^{q} \boldsymbol{h}_k^T(n)\boldsymbol{x}(n) \tag{5.59}$$

where

$$\boldsymbol{x}(n) = \begin{pmatrix} x(n)^k \\ \vdots \\ x(n-M+1)^k \end{pmatrix}. \tag{5.60}$$

The NFxLMS algorithm is obtained by applying the stochastic gradient algorithm [56, 63], as

$$\boldsymbol{c}_l(n+1) = \boldsymbol{c}_l(n) - \frac{\mu_l}{2}\boldsymbol{\Delta}_l^T(n). \tag{5.61}$$

and in this case, the gradient vector $\boldsymbol{\Delta}_l(n)$ is given as

$$\boldsymbol{\Delta}_l(n) = 2e(n)\sum_{k=1}^{q}\sum_{m=0}^{M-1} qh_{q,m}y(n-m)^{q-1}\boldsymbol{x}_2^T(n-m) \tag{5.62}$$

where $\boldsymbol{x}_2(n)$ is defined in (5.58) - (5.59).

The NFxPEM algorithm can also be derived similarly as the memory polynomial predistorter by minimizing the cost function

$$V(\boldsymbol{c}_l) = \lim_{\mathcal{N}\to\infty}\frac{1}{\mathcal{N}}\sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n)\right]. \tag{5.63}$$

where $e(n)$ is the prediction error defined as

$$e(n) = r(n) - z(n). \tag{5.64}$$

The required negative gradient of the error signal $e(n)$ w.r.t. $\boldsymbol{c}_l(n)$ is given by

$$\boldsymbol{\varphi}_l^T(n) = -\sum_{k=1}^{q}\sum_{m=0}^{M-1} qh_{q,m}y(n-m)^{q-1}\boldsymbol{x}_2^T(n-m) \tag{5.65}$$

where $\boldsymbol{x}_2(n)$ is defined in (5.58) with the elements calculated in (5.59). Then, the NFxPEM algorithm follows the same formulation as in Chapter 2 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}_l(0)$.

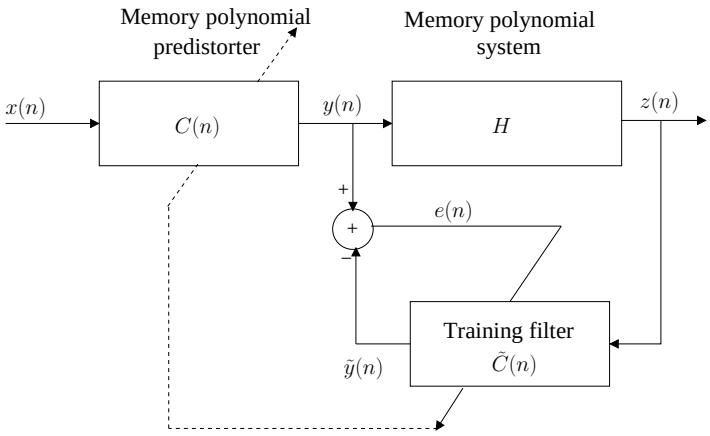

Figure 5.11 The ILA-I approach for the predistortion of memory polynomial systems.

5.2.3. Adaptation algorithms using the ILA approach

The ILA-I approach using memory polynomial predistorter

The ILA approach using memory polynomial predistorter is depicted in Fig 5.11. $\tilde{C}(n)$ is a training filter connected in parallel with the nonlinear physical system H. Then the coefficients of $C(n)$ are estimated *indirectly* as a copy of the coefficients of the training filter $\tilde{C}(n)$. The input-output relation of the training filter is given as

$$\tilde{y}(n) = \sum_{k=1}^{p} \tilde{\boldsymbol{c}}_k^T(n)\boldsymbol{z}_k(n) \tag{5.66}$$

where the parameter vector $\tilde{\boldsymbol{c}}_k(n)$ is defined as

$$\tilde{\boldsymbol{c}}_k(n) = \begin{pmatrix} \tilde{c}_{k,0}(n) & \cdots & \tilde{c}_{k,\widehat{M}-1}(n) \end{pmatrix}^T \tag{5.67}$$

and the corresponding input vector $\boldsymbol{z}_k(n)$ is

$$\boldsymbol{z}_k(n) = \begin{pmatrix} z(n)^k \\ \vdots \\ z(n-\widehat{M}+1)^k \end{pmatrix}. \tag{5.68}$$

Let us define the coefficient vector of the training filter as

$$\tilde{\boldsymbol{c}}(n) = \begin{pmatrix} \tilde{\boldsymbol{c}}_1^T(n) & \cdots & \tilde{\boldsymbol{c}}_p^T(n) \end{pmatrix}^T \tag{5.69}$$

and the error signal $e(n)$ as

$$e(n) = y(n) - \tilde{y}(n). \tag{5.70}$$

The coefficients of the predistorter will be estimated recursively using the RPEM algorithm. The RPEM algorithm is derived by minimization of the cost function

$$V(\tilde{\boldsymbol{c}}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e^2(n)\right]. \tag{5.71}$$

The negative gradient of $e(n)$ w.r.t. the parameter vector $\tilde{\boldsymbol{c}}(n)$ is given by

$$\boldsymbol{\varphi}_t^T(n) = -\boldsymbol{\nabla}_{\tilde{\boldsymbol{c}}(n)} e(n) = \boldsymbol{\nabla}_{\tilde{\boldsymbol{c}}(n)} \tilde{y}(n) = \begin{pmatrix} \boldsymbol{z}_1(n) \\ \vdots \\ \boldsymbol{z}_p(n) \end{pmatrix}^T . \tag{5.72}$$

Then, the NFxPEM algorithm follows the same formulation as in Chapter 3 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}_t(0)$.

The ILA-II approach using the N-L type predistorter

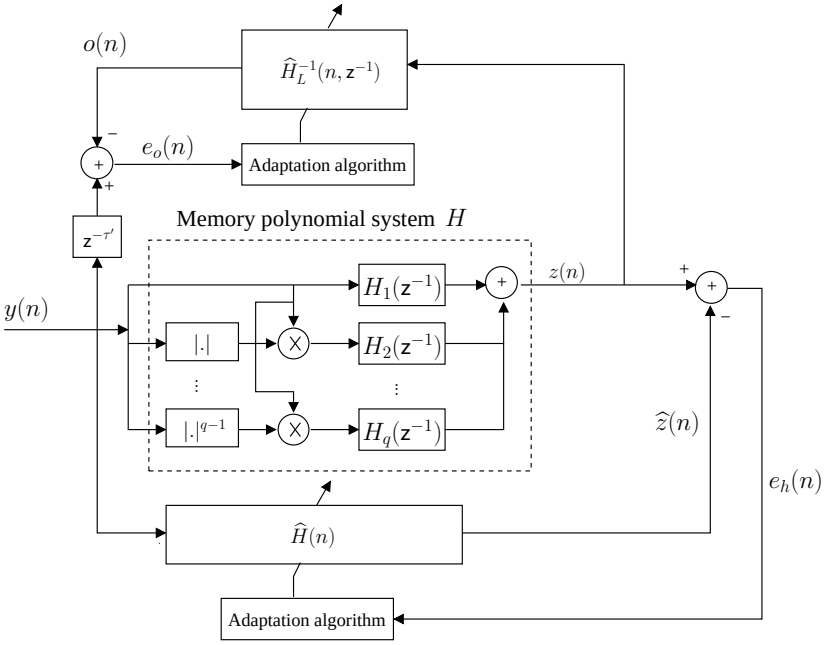

Figure 5.12 System identification in the ILA-II approach.

The system identification scheme for the memory polynomial system and the inverse of the purely linear subsystem is given in Fig. 5.12. Another memory polynomial system $\widehat{H}(n)$ with similar structure is used to identify H. The output signal $\widehat{z}(n)$ can be written as

$$\widehat{z}(n) = \sum_{k=1}^{q} \widehat{\boldsymbol{h}}_k^T(n) \widehat{\boldsymbol{y}}_k(n). \tag{5.73}$$

The parameter vector $\widehat{\boldsymbol{h}}_k(n)$ is defined as

$$\widehat{\boldsymbol{h}}_k(n) = \begin{pmatrix} \widehat{h}_{k,0}(n) & \cdots & \widehat{h}_{k,\widehat{M}_k-1}(n) \end{pmatrix}^T . \tag{5.74}$$

where $\widehat{M}_k$ is the memory length. The corresponding input vector $\widehat{\boldsymbol{y}}_k(n)$ is

$$\widehat{\boldsymbol{y}}_k(n) = \begin{pmatrix} y(n)^k \\ \vdots \\ y(n - \widehat{M}_k + 1)^k \end{pmatrix} . \tag{5.75}$$

Let us define the parameter vector $\widehat{\boldsymbol{h}}(n)$ of $\widehat{H}(n)$ as

$$\widehat{\boldsymbol{h}}(n) = \begin{pmatrix} \widehat{\boldsymbol{h}}_1^T(n) & \cdots & \widehat{\boldsymbol{h}}_q^T(n) \end{pmatrix}^T. \tag{5.76}$$

In order to estimate the parameter vector $\widehat{\boldsymbol{h}}$, the RPEM algorithm is derived by minimizing the cost function

$$V(\widehat{\boldsymbol{h}}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \mathrm{E}\left[e_h^2(n)\right] \tag{5.77}$$

where $e_h(n)$ is the prediction error defined as

$$e_h(n) = z(n) - \widehat{z}(n). \tag{5.78}$$

The formulation of the RPEM algorithm requires the negative gradient of $e_h(n)$ w.r.t. $\widehat{\boldsymbol{h}}(n)$, defined as

$$\boldsymbol{\varphi}_h^T(n) = -\boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}(n)} e(n) = \boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}(n)} \widehat{z}(n). \tag{5.79}$$

Straightforward derivation gives

$$\boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}(n)} \widehat{z}(n) = \begin{pmatrix} \widehat{\boldsymbol{y}}_1(n) \\ \vdots \\ \widehat{\boldsymbol{y}}_q(n) \end{pmatrix}^T. \tag{5.80}$$

Then, the RPEM algorithm follows the same formulation as in Chapter 3 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}_h(0)$.

The inverse of the purely linear subsystem is required to model the predistorter and it is estimated using an adaptive FIR filter $\widehat{H}_L^{-1}(n, \mathsf{z}^{-1})$ with K-tap memory. The output signal of this filter can be written as

$$o(n) = \widehat{H}_L^{-1}(n, \mathsf{z}^{-1}) z(n) = \left[\widehat{\boldsymbol{h}}_L^{-1}(n)\right]^T \boldsymbol{z}(n) \tag{5.81}$$

where the parameter vector $\widehat{\boldsymbol{h}}_L^{-1}(n)$ is defined as

$$\widehat{\boldsymbol{h}}_L^{-1}(n) = \begin{pmatrix} \widehat{h}_0^{-1}(n) & \cdots & \widehat{h}_{K-1}^{-1}(n) \end{pmatrix}^T \tag{5.82}$$

and the corresponding input vectors $\boldsymbol{z}(n)$ is

$$\boldsymbol{z}(n) = \begin{pmatrix} z(n) & \cdots & z(n-K+1) \end{pmatrix}^T. \tag{5.83}$$

The gradient vector $\boldsymbol{\varphi}_o(n)$ required in RPEM algorithm is given by

$$\boldsymbol{\varphi}_o^T(n) = -\boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}_L^{-1}(n)} e_o(n) = \boldsymbol{\nabla}_{\widehat{\boldsymbol{h}}_L^{-1}(n)} o(n) = \boldsymbol{z}^T(n). \tag{5.84}$$

and the RPEM algorithm can follow the same formulation as in Chapter 3 with the design parameters λ_0, $\lambda(0)$ and $\boldsymbol{P}_s(0)$.

Predistorter	**Adaptive Algorithms**	μ	μ_l	$\boldsymbol{P}(0)$	$\boldsymbol{P}_l(0)$	$\boldsymbol{P}_t(0)$	$\boldsymbol{P}_h(0)$	$\boldsymbol{P}_s(0)$
M. P. (odd)	NFxLMS (DLA)	10^{-1}	-	-	-	-	-	-
M. P. (odd)	NFxPEM (DLA)	-	-	$10^2\boldsymbol{I}$	-	-	-	-
M. P. (full)	NFxLMS (DLA)	10^{-1}	-	-	-	-	-	-
M. P. (full)	NFxPEM (DLA)	-	-	$10^2\boldsymbol{I}$	-	-	-	-
N-L type	NFxLMS (DLA)	-	0.9	-	-	-	-	-
N-L type	NFxPEM (DLA)	-	-	-	$10\boldsymbol{I}$	-	-	-
M. P. (odd)	RPEM (ILA-I)	-	-	-	-	$10\boldsymbol{I}$	-	-
M. P. (full)	RPEM (ILA-I)	-	-	-	-	$10\boldsymbol{I}$	-	-
N-L type	RPEM (ILA-II)	-	-	-	-	-	$\boldsymbol{I}$	$\boldsymbol{I}$

Table 5.3 Initialization of all adaptation algorithms for predistortion of the memory polynomial system.

5.2.4. Simulation results

In this simulation study, the following memory polynomial system is considered:

$$z(n) = \sum_{q=1}^{5} \boldsymbol{h}_q^T \boldsymbol{y}_q(n) \tag{5.85}$$

with the kernels

$$\begin{aligned} \boldsymbol{h}_1 &= \begin{pmatrix} 1.0513 & -0.0680 & 0.0289 \end{pmatrix}^T \\ \boldsymbol{h}_3 &= \begin{pmatrix} -0.0542 & 0.2234 & -0.0621 \end{pmatrix}^T \\ \boldsymbol{h}_5 &= \begin{pmatrix} -0.1655 & -0.2451 & 0.1229 \end{pmatrix}^T. \end{aligned} \tag{5.86}$$

In the DLA approach, the memory polynomial predistorter is assumed to be a 5th-order model with memory length $\widehat{M} = 3$ and the predistorter is modeled using either the odd-order terms or full-order terms as suggested in [77]. Also, the memory length of $C_l(n, \mathsf{z}^{-1})$ in the N-L type predistorter is assumed to be $M_l = 7$.

In the ILA-II approach, for modeling the N-L type predistorter, $\widehat{H}(n)$ is chosen as a 5th-order memory polynomial system with memory length $\widehat{M}_1 = \widehat{M}_3 = \widehat{M}_5 = 3$ and $\widehat{H}_L^{-1}(n, \mathsf{z}^{-1})$ is chosen as an adaptive linear FIR filter with memory length $K = 7$.

The number of independent experiments is 100. In each experiment, the input signal to the predistorter is chosen to be a random signal with uniform distribution over $(-1, 1)$ and data length of 4×10^4. The bandwidth of the input signal was limited by a low-pass filter in order to prevent aliasing [13] and the normalized cut-off frequency of this filter is chosen as $\frac{\pi}{10}$. In the ILA-II approach, the predistorter starts to copy the coefficients after 3.2×10^4 input samples. The adaptation algorithms are initialized as in Table 5.3, where M. P. denotes memory polynomial predistorter and the design parameters λ_0 and $\lambda(0)$ in the NFxPEM and RPEM algorithms are chosen as 0.99 and 0.95, respectively.

Predistorter	Adaptive Algorithms	MSD (dB)
M. P. (odd)	NFxLMS (DLA)	−54
M. P. (odd)	NFxPEM (DLA)	−82
M. P. (full)	NFxLMS (DLA)	−56
M. P. (full)	NFxPEM (DLA)	−82
N-L type	NFxLMS (DLA)	−51
N-L type	NFxPEM (DLA)	−75
M. P. (odd)	RPEM (ILA-I)	−81
M. P. (full)	RPEM (ILA-I)	−77
N-L type	RPEM (ILA-II)	−78

Table 5.4 Comparison of the MSD for predistortion of a memory polynomial system.

The MSD value of the memory polynomial system without predistorter is about −15 dB. The MSD of different approaches and algorithms is given in Table. 5.4. From the table, we can see that the predistorter using the DLA approach and NFxLMS algorithm has much worse performance than using other predistortion techniques, and the memory polynomial predistorter using the DLA approach and NFxPEM algorithm achieves the best performance.

Figure 5.13 shows mean PSDs of output signals of the memory polynomial system without and with the predistorter after 4×10^4 samples. From this figure, we can see that the memory polynomial predistorter using the DLA approach and NFxLMS algorithm can't reduce the spectral regrowth efficiently. The N-L type predistorter using the DLA approach and NFxLMS algorithm has better performance but still much worse than other techniques. Normally, the full-order memory polynomial predistorters have a little better performance than the odd-order memory polynomial predistorters, but the odd-order predistorters have lower computational complexity.

5.3. Summary

In this chapter, examples were given where adaptive predistortion was applied to several nonlinear physical system used in practical communication systems, such as the parallel Wiener-type systems and the memory polynomial systems. From the simulation, we can see that most of the proposed methods in previous sections can well compensate the nonlinear distortion and reduce the spectral regrowth caused by the nonlinear physical system.

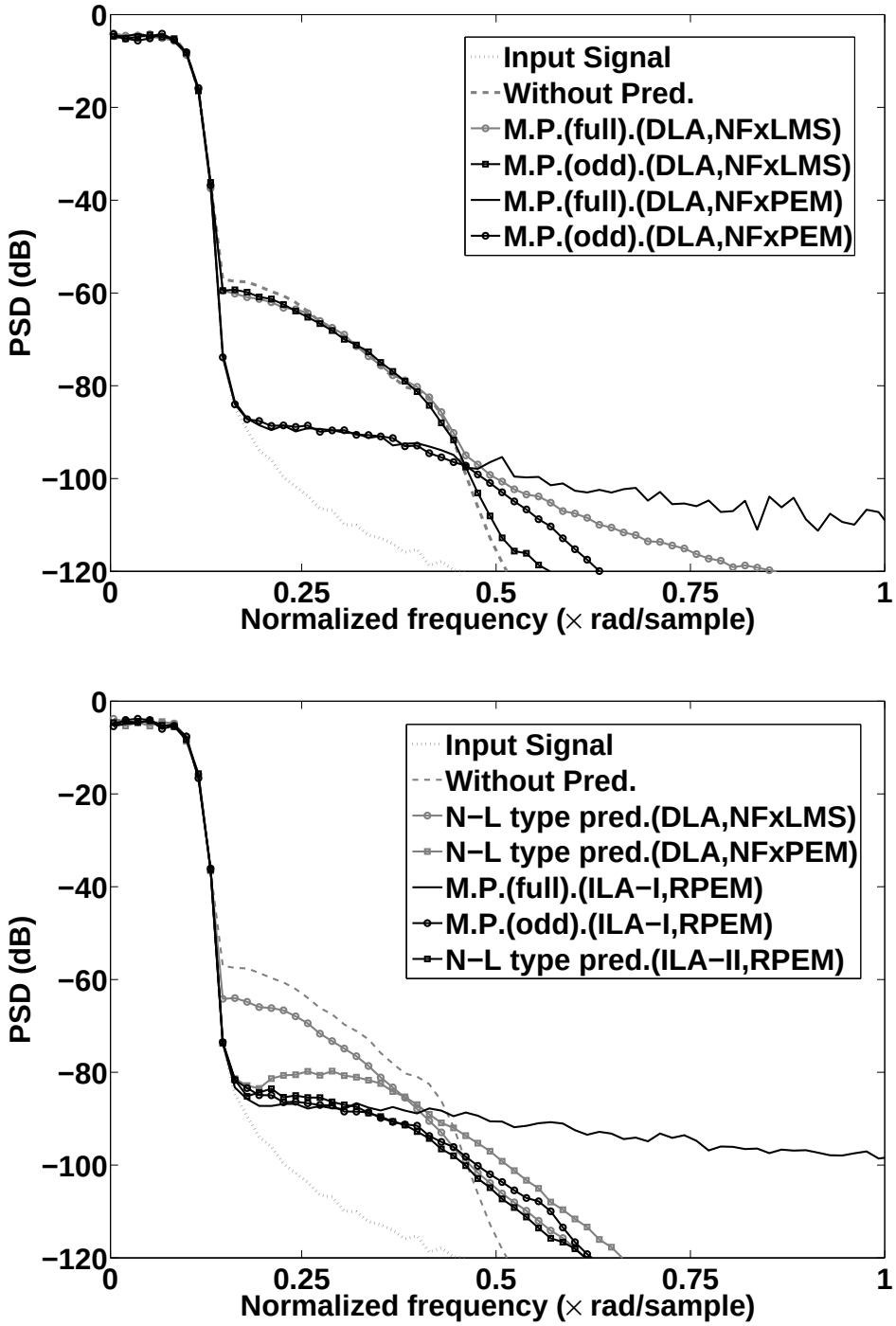

Figure 5.13 Mean PSDs comparison for predistortion of a memory polynomial system.

Chapter 6

Conclusion and Outlook

The primary aim of this thesis was to contribute to both theory and implementation of adaptive predistortion of nonlinear systems. Classic models for the nonlinear physical system include: Volterra, Wiener and Hammerstein models and the corresponding predistorters are modeled as Volterra, Hammerstein and Wiener systems, respectively. The parameters of the predistorter can be estimated adaptively using recursive algorithms based on the Direct Learning Architecture (DLA) or Indirect Learning Architecture (ILA).

In Chapter 2, the adaptation algorithms using the DLA approach have been introduced. The Nonlinear Filtered-x Least Mean Squares (NFxLMS) algorithm is a common adaptation algorithm in the time domain. However, it usually suffers from slow convergence and bad performance. The Nonlinear Filtered-x Recursive Least Squares (NFxRLS) algorithm can speed up the convergence but can only be used for predistortion of the nonlinear physical system, where the predistorter output is linear in its coefficients. The proposed Nonlinear Filtered-x Prediction Error Method (NFxPEM) algorithm can be implemented for predistortion of different nonlinear physical systems. As compared to the NFxLMS algorithm, the NFxPEM algorithm has very fast convergence speed and is much more efficient in suppressing the spectral regrowth. All of these time domain adaptation algorithms require accurate system identification of the nonlinear physical system for the adaptation process. For the predistortion of Wiener systems, the proposed NFxLMS with Initial Subsystem Estimates (NFxLMS-ISE) and NFxPEM with Initial Subsystem Estimates (NFxPEM-ISE) algorithms relax the accurate system identification requirement in NFxLMS and NFxPEM algorithms, by using a simple and fast ISE instead. The NFxLMS-ISE and NFxPEM-ISE algorithms can achieve similar performance as the NFxLMS and NFxPEM algorithms, respectively. To relax the requirement of accurate system identification for other nonlinear physical systems is still an open issue for future research. The proposed SMM method is a frequency domain technique suitable for predistortion of different nonlinear systems. Although this method has high computational complexity and requires matrix inversion in each adaptation iteration, it does not need accurate identification of the nonlinear physical system as required in time domain algorithms.

Chapter 3 concerned with the adaptation algorithms using the ILA approach, which are classified as the ILA-I approach and ILA-II approach. In the ILA-I approach, the Recursive Least Squares (RLS), Kalman Filter (KF) and Recursive Prediction Error Method (RPEM) algorithms can be used to estimate the coefficients of the training filter for the predistortion

of Volterra systems. These algorithms have similar performance. The RPEM algorithm can also be used for predistortion of Wiener and Hammerstein systems. The RPEM algorithm has very fast convergence speed, as well as good performance in reducing the nonlinear distortion in the time domain and spectral regrowth in the frequency domain.

A rule to construct the predistorter is given in the ILA-II approach. If the nonlinear physical system is a weakly nonlinear system and can be divided into two subsystems: a purely linear subsystem and a purely nonlinear subsystem, the predistorter can be constructed using the nonlinear subsystem, the inverse of the linear subsystem and delayed input. Extra filters are needed in order to identify the nonlinear physical system - hence to obtain the estimate of the nonlinear subsystem, and also to estimate the inverse of the linear subsystem. The widely used Least Mean Squares (LMS) algorithm suffers from slow convergence speed and the inaccurate estimates degrade the performance of the predistorter. The proposed RPEM algorithm can obtain more accurate estimates - hence greatly improve the performance of the predistorter.

The adaptive predistorter design is discussed in Chapter 4. The various gradient calculations required in the adaptation algorithms using the DLA and ILA-I approaches can all be obtained by using the same calculation structure, proposed as General Gradient Calculation Architecture (GGCA). Other issues, such as the predistorter model, the adaptation algorithm and the computational complexity, are also discussed in this chapter. The relative comparisons of all adaptation algorithms, including system identification or ISE requirement, extra filters requirement, computational complexity per iteration and convergence speed are summarized. This can help to make a trade-off considering every aspect and choose the suitable techniques in implementation. The input signal also has an important effect [74] on predistortion and this issue will be considered in future research.

Chapter 5 gives several examples of adaptive predistortion applications. Nonlinear models used in practical communication systems, such as the parallel Wiener-type models and memory polynomial models, are considered. The predistorter models are proposed, and the adaptation algorithms using different learning architectures are derived. The simulation results for comparison of all the techniques are given.

In this thesis, the predistorter needs to be at the same sampling rate as the nonlinear physical system, which means if the nonlinear physical system is running at the high sampling rate (after upsampling), the predistorter needs to run at the high sampling rate as well. This requirement increases the design cost of the predistorter. Running predistorters at low sampling rate (before upsampling) and the corresponding learning architecture and adaptation algorithms would be an interesting topic in the future.

Appendix A

Appendix

A.1. The pth-order inverse

The pth-order inverse theory has first been proposed in [16]. Assuming the given nonlinear system H is represented in the Volterra series as

$$y(n) = H[x(n)] = \sum_{k=1}^{\infty} H_k[x(n)] \tag{A.1}$$

where H_k denotes the kth order operator, the pth-order inverse is defined as a Volterra system C, when connected in tandem with H, results in a system in which the first-order Volterra kernel is a unit impulse and the second through the pth-order Volterra kernels are zero. Thus, if C is connected in cascade after H (called post-inverse), see Fig. A.1, the overall system Q consisting of H and C can be rewritten as

$$z(n) = Q[x(n)] = \sum_{k=1}^{\infty} Q_k[x(n)] = x(n) + \sum_{k=p+1}^{\infty} Q_k[x(n)] \tag{A.2}$$

in which Q_k is the kth-order operator of the system Q. Also, if C is connected in cascade before H (called pre-inverse), see Fig. A.1, the overall system R consisting of C and H can be rewritten as

$$z(n) = R[x(n)] = \sum_{k=1}^{\infty} R_k[x(n)] = x(n) + \sum_{k=p+1}^{\infty} R_k[x(n)] \tag{A.3}$$

in which R_k is the kth-order operator of the system R.

The pth-order inverse theory shows that the pth-order post-inverse of a given system H is identical to its pth-order pre-inverse, in the sense that the first p kernels of C in post-inverse and pre-inverse are identical. However, the kernels with orders greater than p are usually different.

Considering the pth-order post-inverse, the 1st-order kernel of C can be determined by requiring

$$Q_1[x(n)] = C_1[H_1[x(n)]] = x(n). \tag{A.4}$$

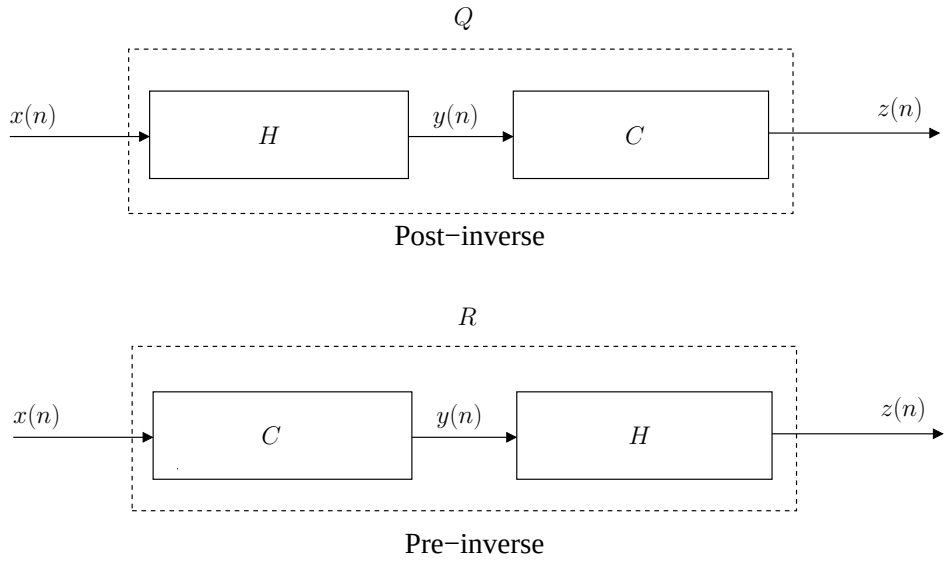

Figure A.1 Post-inverse and Pre-inverse.

Therefore, the 1st-order kernel of C should satisfy

$$C_1(s) = \frac{1}{H_1(s)} \tag{A.5}$$

where $C_1(s)$ and $H_1(s)$ are the Laplace transforms of the 1st-order kernel of C and H.

The 2nd-order kernel of C can be determined by requiring

$$Q_2[x(n)] = C_1[H_2[x(n)]] + C_2[H_1[x(n)]] = 0. \tag{A.6}$$

This will be satisfied only if the operator C_2 satisfies

$$C_2(s)H_1(s) = -C_1(s)H_2(s). \tag{A.7}$$

Post-multiplying both sides by the inverse of the $H_1(s)$ gives

$$C_2(s) = -C_1(s)H_2(s)H_1^{-1}(s) = -C_1(s)H_2(s)C_1(s). \tag{A.8}$$

Figure A.2 is the block diagram of the operator C_2.

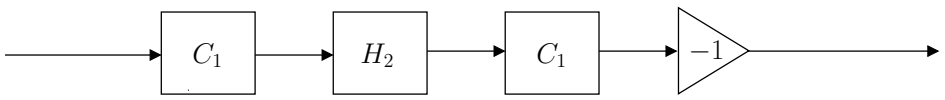

Figure A.2 The operator C_2.

Consequently, the operator $C_k, \quad k = 3, \cdots, p$ can be determined in the same manner. For example, Fig. A.3 gives the block diagram of the operator C_3.

Note that it is always possible to isolate the operator C_k since the only term in the expression for $Q_k[x(n)]$ that involves the operator C_k is $C_k[H_1[x(n)]]$. Also, when determining

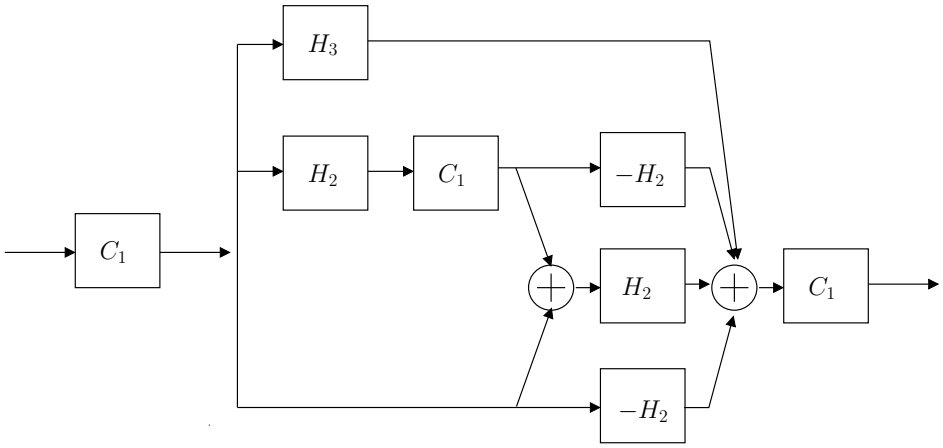

Figure A.3 The operator C_3.

the operator C_k from Q_k, all other terms involve the operators C_i for $i < k$ which have been determined. Also we can observe from the construction that each of the operators C_k is causal and stable if the system H and the operator C_1 is causal and stable. Thus, we can conclude that a stable and causal pth-order post-inverse of a stable and causal nonlinear system H exists if and only if the inverse of the linear operator H_1 is stable and causal, *i.e.*, it is a minimum phase system. Due to that the pth-order post-inverse of a given system H is identical to its pth-order pre-inverse, the design of the pre-inverse will come to the same conclusions.

Bibliography

[1] G. Pupolin, "Performance analysis of digital radio links with nonlinear transmit amplifiers," *IEEE J. on Select Area Communication*, vol. SAC-5, no. 4, pp. 534–546, April 1987, ➠ PDF.

[2] C. Cripps, *RF Power Amplifiers for Wirelss Communications*. Norwood, MA, USA: Artech House, 1999.

[3] A. Conti, D. Dardari, and V. Tralli, "An analytical framework for CDMA system with a nonlinear amplifier and AWGN," *IEEE Trans. on Communications*, vol. 50, no. 7, pp. 1110–1120, 2002, ➠ PDF.

[4] L. Rugini, P. Banelli, and S. Cacopardi, "SER performance of linear multiuser detectors for DS-CDMA downlink with transmitter nonlinear distortions," *IEEE Trans. on Vehicular Technology*, vol. 53, no. 7, pp. 992–1000, 2004, ➠ PDF.

[5] H. Kressel and eds., *Semiconductor Devices for Optical Communication, (Topics in Applied Physics vol. 39)*. New York: Springer-Verlag, 1980.

[6] X. N. Fernando and A. B. Sesay, "Higher order adaptive filter based predistortion for nonlinear distortion compensation of radio over fiber links," in *Proc. of the IEEE International Conference on Communications*, 2000, pp. 367–371, ➠ PDF.

[7] H. F. Olson, *Acoustical Engineering*. Toronto: D. Van Nostrand Company, Inc., 1964.

[8] K. B. Benson and eds., *Audio Engineering Handbook*. Toronto: McGraw-Hill Book Company, 1988.

[9] X. Y. Gao and W. M. Snelgrove, "Adaptive linearization schemes for weakly nonlinear systems using adaptive linear and nonlinear FIR filters," in *Proc. of the Midwest Symposium on Circuits and Systems*, 1990, pp. 9–12, ➠ PDF.

[10] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[11] V. E. DeBrunner and D. Y. Zhou, "Active nonlinear noise control with certain nonlinearities in the secondary path," in *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, 2003, pp. 2053–2057, ➠ PDF.

[12] D. Y. Zhou and V. E. DeBrunner, "A new active noise control algorithm that requires no secondary path identification based on the SPR property," *IEEE Trans. on Signal Processing*, vol. 55, no. 5, pp. 1719–1729, 2007, ➠ PDF.

[13] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems.* Florida, USA: R. E. Krieger, 1989.

[14] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing.* N.Y., USA: John Wiley & Sons, Inc., 2000.

[15] W. A. Frank, "On the equalization of nonlinear systems," in *Proc. of the Asilomar Conference on Signals, Systems and Computers*, 1996, pp. 1157–1160, ➠ PDF.

[16] M. Schetzen, "Theory of pth-order inverses of nonlinear systems," *IEEE Trans. on Circuits and Systems*, vol. CAS-23, no. 5, pp. 285–291, 1976, ➠ PDF.

[17] A. A. M. Saleh and J. Salz, "Adaptive linearization of power amplifiers in digital radio systems," *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1019–1033, Apr 1983.

[18] J. K. Cavers, "Amplifier linearization using a digital predistorter with fast adaptation and low memory requirement," *IEEE Trans. on Vehicular Technology*, vol. 39, no. 4, pp. 374–382, Nov 1990, ➠ PDF.

[19] M. Ghaderi, S. Kumar, and D. E. Dodds, "Fast adaptive polynomial I and Q predistorter with global optimization," *Inst. Elect. Eng. Proc. Commun.*, vol. 143, no. 2, pp. 78–86, Apr 1996, ➠ PDF.

[20] W. Boesch and G. Gatti, "Measurement and simulation of memory effects in predistortion linearization," *IEEE Trans. on Microwave Theory and Techniques*, vol. 37, no. 12, pp. 1885–1890, Dec 1989, ➠ PDF.

[21] J. H. K. Vuolevi, T. Rahkonen, and J. P. A. Manninen, "Measurement technique for characterizing memory effects in RF power amplifiers," *IEEE Trans. on Microwave Theory and Techniques*, vol. 49, no. 8, pp. 1383–1388, Aug 2001, ➠ PDF.

[22] J. Kim and K. Konstantinou, "Digital predistortion of wideband signals based on power amplifier model with memory," *Electron. Lett*, vol. 37, no. 23, pp. 1417–1418, Nov 2001, ➠ PDF.

[23] Y. H. Lim, Y. S. Cho, I. W. Cha, and D. H. Youn, "An adaptive nonlinear prefilter for compensation of distortion in nonlinear systems," *IEEE Trans. on Signal Processing*, vol. 46, no. 6, pp. 1726–1730, 1998, ➠ PDF.

[24] H. W. Kang, Y. S. Cho, and D. H. Youn, "Adaptive precompensation of Wiener systems," *IEEE Trans. on Signal Processing*, vol. 46, no. 10, pp. 2825–2829, 1998, ➠ PDF.

[25] ——, "On compensating nonlinear distortions of an OFDM system using an efficient adaptive predistorter," *IEEE Trans. on Communications*, vol. 47, no. 4, pp. 522–526, 1999, ➠ PDF.

[26] D. Y. Zhou and V. E. DeBrunner, "Novel adaptive nonlinear predistorters based on the direct learning algorithm," *IEEE Trans. on Signal Processing*, vol. 55, no. 1, pp. 120–133, 2007, ➠ PDF.

[27] C. Eun and E. J. Powers, "A new Volterra predistorter based on indirect learning architecture," *IEEE Trans. on Signal Processing*, vol. 45, no. 1, 1997, ➠ PDF.

[28] L. Ding, Z. Ma, D. R. Morgan, M. Zierdt, and J. Pastalan, "A least square/Newton method for digital predistortion of wideband signals," *IEEE Trans. on Communications*, vol. 54, no. 5, pp. 833–840, May 2006, ➠ PDF.

[29] D. R. Morgan, Z. Ma, and L. Ding, "Reducing measurement noise effects in digital predistortion of RF power amplifiers," in *Proc. of the IEEE International Conference on Communications*, 2003, pp. 2436–2439, ➠ PDF.

[30] Y. Qian and T. Yao, "Structure for adaptive predistortion suitable for efficient adaptive algorithm application," *Electron. Letter*, vol. 38, pp. 1282–1283, Oct 2002, ➠ PDF.

[31] P. Singerl, A. Agrawal, A. Garg, Neelabh, G. Kubin, and H. Eul, "Complex baseband predistorters for nonlinear wideband RF power amplifiers," in *Proc. of the IEEE Midewest Symposium on Circuits and Systems*, 2006, pp. 675–678, ➠ PDF.

[32] G. L. Sicuranza and G. Ramponi, "Adaptive nonlinear digital filters using distributed arithmetic," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-34, pp. 518–526, 1986, ➠ PDF.

[33] T. Ogunfunmi, *Adaptive Nonlinear System Identification.* Berlin: Springer, 2007.

[34] E. Abd-Elrady and L. Gan, "Direct predistortion of nonlinear systems using adaptive Volterra system and prediction error method," *IEEE Signal Processing Letters*, Submitted in late 2009.

[35] ——, "Direct predistortion of Hammerstein and Wiener systems using prediction error method," *Signal Processing*, Submitted in late 2009.

[36] E. Abd-Elrady, L. Gan, and G. Kubin, "Direct linearization of weakly nonlinear Volterra systems using adaptive linear and nonlinear FIR filters," in *Proc. of the IFAC Symposium on System Identification*, 2009, ➠ PDF.

[37] ——, "Adaptive predistortion of nonlinear Volterra systems using spectral magnitude matching," in *Proc. of the International Conference on Acoustics, Speech, and Signal Processing*, 2009, pp. 2985–2988, ➠ PDF.

[38] L. Gan and E. Abd-Elrady, "A NFxLMS algorithm with initial subsystem estimates for digital predistortion of Wiener systems," in *Proc. of the European Signal Processing Conference*, 2008, ➠ PDF.

[39] ——, "Adaptive predistortion of IIR Hammerstein system using the nonlinear filtered-x LMS algorithm," in *Proc. of the Symposium on Communication Systems, Networks and Digital Signal Processing*, 2008, pp. 702–705, ➠ PDF.

[40] E. Abd-Elrady, L. Gan, and G. Kubin, "Direct and indirect learning methods for adaptive predistortion of IIR Hammerstein systems," *e & i Elektrotechnik und Informationstechnik. Special issue on Analog Mixed Signal Circuit and Systems*, vol. 125, no. 4, pp. 126–131, April 2008, ➠ PDF.

[41] L. Gan and E. Abd-Elrady, "Linearization of weakly nonlinear systems using adaptive FIR filters and recursive prediction error method," in *Proc. of the International Workshop on Machine Learning for Signal Processing*, 2008, pp. 409–414, ➠ PDF.

[42] E. Abd-Elrady and L. Gan, "Adaptive predistortion of Hammerstein systems based on indirect learning architecture and prediction error method," in *Proc. of the International Conference on Signals and Electronic Systems*, 2008, pp. 389–392, ➠ PDF.

[43] E. Abd-Elrady, L. Gan, and G. Kubin, "Distortion compensation of nonlinear systems based on indirect learning architecture," in *Proc. of the IEEE International Symposium on Communications, Control and Signal Processing*, 2008, pp. 184–187, ➠ PDF.

[44] L. Gan and E. Abd-Elrady, "Digital predistortion of memory polynomial systems using direct and indirect learning architectures," in *Proc. of the IASTED Conference on Signal and Image Processing*, 2009, ➠ PDF.

[45] L. Gan, E. Abd-Elrady, and G. Kubin, "A simplified predistorter for distortion compensation of parallel Wiener-type systems based on direct learning architecture," in *Proc. of the DSP/SPE Workshop*, 2009, pp. 72–77, ➠ PDF.

[46] L. Gan and E. Abd-Elrady, "Digital predistortion of parallel Wiener-type systems using the RPEM and NFxLMS algorithms," in *Proc. of the International Conference on Signal Processing*, 2008, pp. 149–152, ➠ PDF.

[47] E. Abd-Elrady and L. Gan, "Identification of Hammerstein and Wiener models using spectral magnitude matching," in *Proc. of the IFAC World Congress on Automatic Control*, 2008, pp. 6440–6445, ➠ PDF.

[48] E. Abd-Elrady, "Adaptive predistortion of Wiener and Hammerstein systems using spectral magnitude matching," in *Proc. of the IASTED Conference on Signal and Image Processing*, 2009, ➠ PDF.

[49] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 1985.

[50] P. L. Feintuch, N. J. Bershad, and A. K. Lo, "A frequency domain model for filtered LMS algorithms - Stability analysis design and elimination of the training mode," *IEEE Trans. on Singal Processing*, vol. 41, no. 4, pp. 1518–1531, Apr 1993, ➠ PDF.

[51] E. Bjarnason, "Analysis of the filtered-x LMS algorithm," *IEEE Trans. on Speech and Audio Processing*, vol. 3, no. 6, pp. 504–514, Nov 1995, ➠ PDF.

[52] A. K. Wang and W. Ren, "Convergence analysis of the multi-variable filtered-x LMS algorithm with application to active noise control," *IEEE Trans. Singal Processing*, vol. 47, no. 4, pp. 1166–1169, Apr 1999, ➠ PDF.

[53] M. H. Costa, J. C. M. Bermudez, and N. J. Bershad, "Stochastic analysis of the filtered-X LMS algorithm in systems with nonlinear secondary paths," *IEEE Trans. on Signal Processing*, vol. 50, no. 6, 2002, ➠ PDF.

[54] Y. Hinamoto and H. Sakai, "Analysis of the filtered-x LMS algorithm and a related new algorithm for active control of multitonal noise," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 14, no. 1, pp. 123–130, Jan 2006, ➠ PDF.

[55] S. Haykin, *Adaptive Filter Theory*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[56] L. Ljung, *System Identification - Theory for the User*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[57] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA, USA: M.I.T. Press, 1983.

[58] S. D. Snyder and C. H. Hansen, "The influence of transducer transfer functions and acoustic time delays on the LMS algorithm in active noise control systems," *J. Sound Vibration*, vol. 140, pp. 409–424, 1990, ➠ PDF.

[59] ——, "The effect of transfer function estimation errors on the Filtered-x LMS algorithm," *IEEE Trans. on Signal Processing*, vol. 42, no. 4, pp. 950–953, 1994, ➠ PDF.

[60] Z. Vukic, L. Kuljaca, D. Donlagic, and S. Tesnjak, *Nonlinear Control Systems*. N.Y., USA: CRC, 2003.

[61] T. F. Quatieri, D. A. Reynolds, and G. C. O'Leary, "Estimation of handset nonlinearity with application to speaker recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 8, no. 5, pp. 567–584, 2000, ➠ PDF.

[62] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*. Reading, Massachusetts: Addison-Wesley, 1973.

[63] T. Söderström and P. Stoica, *System Identification*. Hemel Hempstead, United Kingdom: Prentice-Hall International, 1989.

[64] M. H. Hayes, *Statistical Digital Signal Prcoessing and Modeling*. New York, USA: Wiley, 1998.

[65] P. Crama and J. Schoukens, "Initial estimates of Wiener and Hammerstein systems using multisine excitation," *IEEE Trans. on Instrumentation and Measurement*, vol. 50, no. 6, pp. 1791–1795, 2001, ➠ PDF.

[66] T. P. Dobrowiecki and J. Schoukens, "Practical choices in the FRF measurement in presence of nonlinear distortions," *IEEE Trans. on Instrumentation and Measurement*, vol. 50, no. 1, pp. 2–7, Feb 2001, ➠ PDF.

[67] E. V. der Ouderaa and J. Schoukens, "Peak factor minimization using a time-frequency domain swapping algorithm," *IEEE Trans. on Instrumentation and Measurement*, vol. IM-37, no. 1, pp. 145–147, Mar 1988, ➠ PDF.

[68] P. Gilabert, G. Montoro, and E. Bertran, "On the Wiener and Hammerstein models for power amplifier predistortion," in *Proc. of the Asia-Pacific Microwave Conference*, vol. 2, Suzhou, China, 2005, ➠ PDF.

[69] E. Abd-Elrady, "A recursive prediction error algorithm for digital predistortion of FIR Wiener systems," in *Proc. of the Symposium on Communication Systems, Networks and Digital Signal Processing*, 2008, pp. 698–701, ➠ PDF.

[70] D. Schwingshackl, *Digital Enhancement and Multirate Processing Methods for Nonlinear Mixed Signal Systems*. PhD thesis, Graz, Austria: TU Graz, 2005.

[71] A. Carini, G. L. Sicuranza, and V. J. Mathews, "On the inversion of certain nonlinear systems," *IEEE Signal Processing Letters*, vol. 4, no. 12, pp. 334–336, Dec 1997, ➠ PDF.

[72] A. Carini, V. J. Mathews, and G. L. Sicuranza, "Equalization of recursive polynomial systems," *IEEE Signal Processing Letters*, vol. 6, no. 12, pp. 312–314, Dec 1999, ➠ PDF.

[73] H. Koeppl and G. Paoli, "Non-linear modeling of a broadband SLIC for ADSL-Lite-over-POTS using harmonic analysis," in *Proc. of the IEEE International Symposium on Circuits and Systems*, vol. 2, Phoenix-Scottsdale, Ariz, USA, 2002, pp. 133–136, ➠ PDF.

[74] H. Koeppl, A. S. Josan, G. Paoli, and G. Kubin, "The Cramer-Rao bound and DMT signal optimisation for the identification of a Wiener type model," *EURASIP J. on Applied Signal Processing*, vol. 12, pp. 1817–1830, 2004, ➠ PDF.

[75] H. Koeppl, *Nonlinear System Identification for Mixed Signal Processing*. PhD thesis, Graz, Austria: TU Graz, 2004.

[76] M. Isaksson, D. Wisell, and D. Ronnow, "A comparative analysis of behavioral models for RF power amplifiers," *IEEE Trans. on Microwave Theory and Techniques*, vol. 54, no. 1, pp. 348–359, 2006, ➠ PDF.

[77] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina, "Memory polynomial predistorter based on the indirect learning architecture," in *Proc. of GLOBECOM*, Taipei, Taiwan, 2002, pp. 967–971, ➠ PDF.

[78] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenny, J. Kim, and C. R. Giardina, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Trans. on Communications*, vol. 52, no. 1, pp. 159–165, Jan 2004, ➠ PDF.

[79] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan, "A generalized memory polynomial model for digital predistortion of RF power amplifiers," *IEEE Trans. on Signal Processing*, vol. 54, no. 10, pp. 3852–3860, Oct 2006, ➠ PDF.