Graz University of Technology

~ Master's Thesis ~

Reconstruction Methods for Time-Varying Systems

conducted at the

Signal Processing and Speech Communications Laboratory
Graz University of Technology, Austria

by

Matthias Hotz

Supervisor:
DI Dr. Christian Vogel

Assessor:
Assoc. Prof. DI Dr. Klaus Witrisal

Graz, September 5, 2012

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Date	(Signature)

Zusammenfassung

Reale Signalverarbeitungssysteme weisen zumeist eine Abweichung vom zugrundeliegenden idealen Modell auf und ändern sich aufgrund äußerer Einflüsse oftmals mit der Zeit. Folglich sind in etlichen Anwendungen Rekonstruktionsmethoden für zeitvariante Systeme notwendig, um die Abweichung vom idealen Modell zu korrigieren oder verringern. Für lineare zeitvariante (LZV) Systeme sind bereits etliche Rekonstruktionsmethoden verfügbar. Um diese jedoch in Echtzeitsystemen oder bei hohen Datenraten einsetzen zu können, ist aufgrund der Leistungsgrenzen der Hardware Parallelverarbeitung notwendig. Bisher sind noch keine einfachen und systematischen Methoden für die Parallelverarbeitung mit LZV Systemen verfügbar und werden im ersten Teil der vorliegenden Arbeit erörtert. Dabei wird eine praxistaugliche und effiziente Methode für die Parallelisierung von faltungsbasierten LZV Systemen vorgestellt, welche auch als Basis für komplexere LZV Systeme dient. Die Methode wird anhand des Farrow-Filters und iterativer Korrekturverfahren für Fehlanpassungen bei zeitlich versetzt arbeitenden Analog-Digital-Umsetzern veranschaulicht. Die Literatur zu nichtlinearen zeitvarianten Systemen ist vergleichsweise überschaubar und es wurden bisher nur wenige Rekonstruktionsmethoden vorgestellt. Im zweiten Teil dieser Arbeit wird ein neuer Ansatz für eine spezielle Klasse von Rekonstruktionsmethoden, sogenannten Entzerrern, vorgestellt. Dabei wird gezeigt, dass bestimmte nichtlineare zeitvariante Systeme auf ein LZV System zurückgeführt werden können. Diese neue Darstellung des Systems wird anschließend herangezogen, um zwei Methoden zur Entzerrung abzuleiten und zu analysieren. Darauffolgend werden Zusammenhänge zu bestehenden Methoden aufgezeigt, die Berechungskomplexität diskutiert und die Qualität der Entzerrung anhand von Simulationen verglichen. Dabei stellt sich heraus, dass die hergeleiteten Methoden unter bestimmten Rahmenbedingungen sehr gute Rekonstruktionsergebnisse erzielen und bei nichtlinearen Systemen, welche sich verhältnismäßig schnell mit der Zeit ändern, einen wesentlichen Vorteil in Bezug auf die Berechungskomplexität bieten.

Abstract

Practical signal processing systems deviate in general from the underlying ideal model and often vary with time because of external influences. Consequently, reconstruction methods for time-varying systems are desired to compensate or mitigate the deviation from the ideal system. For linear time-varying (LTV) systems, various reconstruction methods have already been established. However, for real-time or high data rate applications, a systematic method for parallel processing is required, which has not been available yet. This method is addressed in the first part of this thesis, where a convenient and efficient method for parallel processing with convolution-based LTV systems is introduced. Furthermore, this result provides the foundation for parallel processing with more complex structures, which is demonstrated for Farrow filters and iterative correction structures in the context of mismatch correction for time-interleaved analog-to-digital converters. In the case of nonlinear time-varying systems, the literature is rather sparse and only a few reconstruction methods exist. The second part of this thesis is devoted to the investigation of a novel approach to a particular class of reconstruction methods for nonlinear time-varying systems, i.e., equalizers. It is shown that a large class of nonlinear systems can be represented as an LTV system and, based on this perspective, two equalization methods are derived and analyzed. Furthermore, various relationships between these and existing methods are highlighted, the computational complexity is discussed, and the equalization performance is compared by means of a simulation. It is concluded that the derived methods perform very well under certain conditions and exhibit a serious computational advantage for nonlinear systems which vary reasonably fast with time.

Acknowledgements

I would like to profoundly thank Verena Frick for the encouragement to pursue an academic education, for her continuous support, and for enriching my life in countless ways. I am also deeply grateful to my parents, for their love, dedication and support, and my siblings, especially my brother Thomas, who guided me into engineering. Furthermore, I appreciatively acknowledge the Republic of Austria for funding my studies.
I am thankful to Assoc. Prof. DI Dr. Klaus Witrisal for assessing this thesis and I want to express my sincere gratitude to DI Dr. Christian Vogel, who provided me with a continuous stream of exciting and challenging ideas, encouraged the scientific orientation of this work, and supported me within and beyond this thesis.

Matthias Hotz
Graz, September 2012

Contents

List of Figures

List of Algorithms

List of Abbreviations

ADC	Analog-to-digital converter
DMC	Differentiator-multiplier cascade
FIR	Finite impulse response
IIR	Infinite impulse response
LTI	Linear time-invariant
LTV	Linear time-varying
MIMO	Multiple-input multiple-output
SISO	Single-input single-output
SNR	Signal-to-noise ratio
TI-ADC	Time-interleaved analog-to-digital converter

Chapter 1

Introduction

Besides the intended behavior, practical signal processing systems often exhibit some undesired impact on the processed signal as well. For example, communication channels do not only transfer but also filter the transmitted signal or analog circuits feature slightly nonlinear characteristics even if they are designed for linear operation. Due to external influences, for instance, changes in the communication channel or temperature variations, real-world systems generally vary with time and, therefore, need to be modeled as time-varying systems. Consequently, reconstruction methods for time-varying systems are required, which provide the means to compensate or correct for the undesired impact on the processed signal. Because of the different underlying theory, it is necessary to distinguish between reconstruction methods for linear time-varying (LTV) systems and nonlinear time-varying systems. For LTV systems, applicable reconstruction methods are readily available [1–6]. However, for real-time or high data rate applications, it is often necessary to parallelize operations to overcome the speed limitations of the hardware or increase the system's throughput. This necessity provided the impulse to investigate systematic methods for parallel processing with LTV systems, which is covered in Chapter 2. Reconstruction methods for nonlinear time-varying systems are seldom discussed in the literature, probably due to the significant computational complexity of the existing models for nonlinear systems with memory. However, the continuous advances in semiconductor technology lead to increasing processing power for digital signal processing systems. This in turn makes it worthwhile to study such reconstruction methods, which is effectively done in Chapter 3.

The resulting structures of reconstruction methods for LTV systems are mostly LTV systems as well [1]. As LTV systems with a demand for parallel processing also emerge in various other contexts, e.g., biomedical signal processing, audio signal processing, and telecommunications, it is beneficial to consider a general LTV filter to facilitate the universal applicability of the results. In Chapter 2, a systematic method for parallel processing is derived for a general LTV filter and, subsequently, this result is utilized to attain parallel processing with more involved structures. These are the widely applicable Farrow filter and iterative reconstruction methods, where the latter is discussed in general and for a particular reconstruction method for nonuniformly sampled band limited signals, the differentiator-multiplier cascade.

Chapter 3 considers special reconstruction methods, i.e., equalization methods, for a particular class of time-varying nonlinear systems, i.e., those which can be modeled using a time-varying discrete-time Volterra series. An overview of existing methods is provided and, subsequently, a

novel perspective on Volterra systems is established. This new system description is utilized to derive and analyze equalization methods. Finally, simulations are performed to compare their performance to the existing methods and highlight their particular properties.

1.1 Scientific Contributions

The work on this thesis gave rise to some scientific results, which shall be highlighted below to distinguish them from previous findings.

Parallel Processing with Linear Time-Varying Filters:

- A systematic approach to parallel processing with linear time-invariant (LTI) systems has already been established, e.g., in [7], and provides a set of equations, termed *design equations*, which describe the structure for parallel processing. For LTV systems, the fundamental concept for parallel processing was addressed in [8], but it lacks a design equation. In this thesis, a design equation for systematic parallel processing with LTV systems is introduced, which is the generalization of the design equations for LTI systems.
- Using the novel design equation, parallel processing with the widely applicable *Farrow filter* is presented, where no computational overhead is introduced.
- The necessary structural modifications for parallel processing and utilization of the design equation is discussed for *iterative correction structures*, which provides the means for parallel processing with several reconstruction methods.
- A structure for parallel processing with the differentiator-multiplier cascade is introduced, which illustrates how the various concepts are combined to attain parallel processing with more complex structures.

Equalization of Time-Varying Nonlinear Systems:

- It is shown that the post-equalization method presented in [9], which is based on a nonlinear fixed-point iteration, is in fact a particular Pth-order inverse.
- A novel perspective on the Volterra series is established by illustrating that it can be viewed as an LTV system, which provides a new foundation to derive and analyze equalization methods.
- Two equalization methods are derived, the Richardson and Jacobi equalizer, where the former is shown to coincide with the nonlinear Richardson iteration and the latter is entirely novel. Furthermore, it is demonstrated that the Richardson equalizer can be regarded as a generalization of the post-equalization method discussed in [9].
- The range of applicability, i.e., the condition for convergence, is established analytically for the Richardson equalizer.
- The equalization performance of four different methods is compared via simulations to illustrate their particular characteristics. Furthermore, the computational complexity is discussed and related to the equalization performance.

Chapter 2

Parallel Processing with Linear Time-Varying Filters

2.1 Introduction

For single-input single-output (SISO) LTI filters, established methods for parallel processing are readily available and, e.g., are documented in [7]. Therein, a SISO LTI filter is transformed into a multiple-input multiple-output (MIMO) LTI filter to enable parallel processing, i.e., block processing of consecutive samples. This transformation is based on a polyphase decomposition of the SISO LTI filter and the structure of the MIMO LTI filter is described by a set of equations, which may be regarded as design equations for the MIMO LTI filter. However, this SISO to MIMO transformation cannot be applied to LTV filters without further consideration due to the time-varying impulse response. In [8], this issue is essentially addressed, but it might be overlooked quite easily as it is embedded into a general elaboration on time-varying filters and filter banks and, furthermore, the discussion therein lacks a convenient design equation as it is available for the time-invariant case. This motivated the derivation of a SISO to MIMO transformation which adopts the fundamental concepts in [8], but results in a design equation which generalizes the one for LTI filters to LTV filters.

This derivation is presented in Section 2.2 and the application of the design equation is exemplified in Section 2.3, where a Farrow filter and correction structures for time-interleaved analog-to-digital converters (TI-ADCs) are considered. Section 2.4 discusses some implementation aspects and Section 2.5 concludes the chapter with a summary.

2.2 SISO to MIMO LTV Filter Transformation

In the following, the transformation of a SISO LTV filter to a MIMO LTV filter is derived. The SISO LTV filter is shown in Figure 2.1, where $x[n]$ is the input signal and $y[n]$ is the output signal. The output of this filter can be described by the convolution of the input signal with the

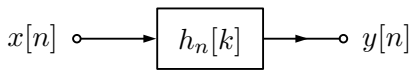


Figure 2.1: SISO LTV filter.

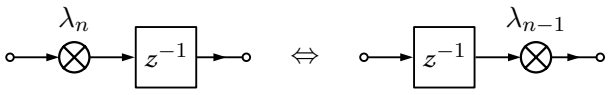


Figure 2.2: Interchange of a time-varying multiplier λ_n, where n denotes the time dependence, and a delay element denoted by z^{-1}, which delays the signal by one sample [8].

time-varying impulse response $h_n[k]$, i.e.,

$$y[n] = \sum_{k=-\infty}^{\infty} h_n[k]x[n-k] \; . \tag{2.1}$$

Since the conventional z-transform is not defined for a time-varying filter, it is redefined here as the z-transform of the filter "frozen" at time instant n [8], i.e.,

$$H_n(z) = \sum_{k=-\infty}^{\infty} z^{-k} h_n[k] \; . \tag{2.2}$$

In the context of time-varying filters, it is important to recognize that delay elements and time-varying multipliers may *not* be interchanged without further consideration as in the time-invariant case, but the time dependency needs to be taken into account as well [8], which is illustrated in Figure 2.2. In order to keep the underlying structure transparent in the z-domain, a refined notation is introduced for the remainder of this chapter. In particular, it is defined that the order of terms in equations in the z-domain corresponds to the structure of the underlying filter, i.e., a delay preceding a filter is written to its left, whereas a delay following a filter is written to its right. This implies that delay elements, LTV filters and time-varying multipliers do not commute under multiplication in the z-domain, but the rule in Figure 2.2 must be respected instead. Furthermore, this concept of notation is extended to the z-transform of filters, where finite impulse response (FIR) filters with the delay chain at the input (direct form) are denoted by writing the z to the left, as done in (2.2), and FIR filters with the delay chain at the output (transposed form) are denoted by writing the z to the right [8].

A linear M-periodically time-varying filter may be represented as a maximally decimated M-channel filter bank by processing M subsequent samples in parallel using the corresponding impulse responses and time-interleaving the results using decimators and expanders [10]. The same concept may as well be applied to a SISO LTV filter as depicted in Figure 2.3. Therein, the filters are followed by a decimator, hence a polyphase decomposition may be applied to reduce the number of arithmetic operations per unit time [10]. In order to keep the derivation as simple as possible, the interchange of time-varying multipliers and delay elements is avoided during polyphase decomposition by assuming $H_n(z)$ to be a direct form FIR filter, cf. Appendix A. However, the method below is nonetheless applicable to FIR filters in transposed form and even to infinite impulse response (IIR) filters, if they permit an appropriate polyphase decomposition [10]. Given this assumption, the M-fold polyphase decomposition of $H_n(z)$ in (2.2) is given by [8]

$$H_n(z) = \sum_{l=0}^{M-1} z^{-l} \tilde{H}_n^{(l)}(z^M) \; , \tag{2.3}$$

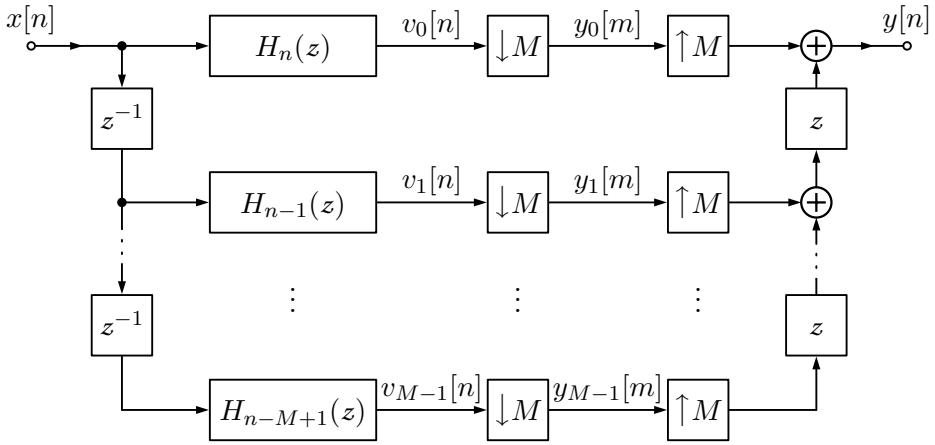


Figure 2.3: SISO LTV filter in multirate respresentation (cf. [10]).

where the polyphase components are

$$\tilde{H}_n^{(l)}(z^M) = \sum_{k=-\infty}^{\infty} z^{-Mk} h_n[Mk+l] \; . \tag{2.4}$$

In order to apply the polyphase decomposition to Figure 2.3, the output of the filter $H_{n-k}(z)$ in channel k is identified as

$$V_k(z) = X(z) z^{-k} H_{n-k}(z) \; ,$$

for $k = 0, \ldots, M-1$. Applying the polyphase decomposition in (2.3) gives

$$V_k(z) = \sum_{l=0}^{M-1} X(z) z^{-(k+l)} \tilde{H}_{n-k}^{(l)}(z^M) \; , \tag{2.5}$$

where z^{-k} and z^{-l} were combined as both precede the filters $\tilde{H}_{n-k}^{(l)}(z^M)$. In order to move the M-fold decimator in front of the filters $\tilde{H}_{n-k}^{(l)}(z^M)$ (cf. Nobel identity 1 in [10]), it is applied to $V_k(z)$. Decimation is described in the z-domain by [10]

$$Y_k(z) = \Big[V_k(z) \Big]_{\downarrow M} = \frac{1}{M} \sum_{i=0}^{M-1} V_k(z^{1/M} W_M^i) \; , \tag{2.6}$$

where $W_M = \mathrm{e}^{-\jmath 2\pi/M}$, i.e., the Mth root of unity. However, using (2.5) in (2.6) is not straightforward, as (2.6) is only capable of describing the implications in the z-domain and obscures the impact on the time-domain.[1] Besides retaining only every Mth sample and discarding the ones in between, which is well described by (2.6), the decimator further changes the time index from n before the decimator to m after the decimator (cf. Figure 2.3), where $n = Mm$, i.e., one time step in m corresponds to M time steps in n. Therefore, the M-fold decimation of $\tilde{H}_n^{(l)}(z^M)$

[1] Note that the z-domain is nonetheless better suited for the derivation as it provides a straightforward insight into the filter structure, which is not that explicitly visible in the time domain description.

leads to the polyphase components

$$H_{Mm}^{(l)}(z) = \sum_{k=-\infty}^{\infty} z^{-k} h_{Mm}^{(l)}[k] \tag{2.7}$$

with the corresponding impulse responses

$$h_{Mm}^{(l)}[k] = h_{Mm}[Mk + l] \,,$$

where the change in the time index ($n \to Mm$) and extraction of every Mth sample ($z^M \to z$) is respected. Considering these particularities, (2.5) may be used in (2.6), and in conjunction with (2.7) this results in

$$Y_k(z) = \frac{1}{M} \sum_{i=0}^{M-1} \sum_{l=0}^{M-1} X(z^{1/M} W_M^i) \left(z^{1/M} W_M^i\right)^{-(k+l)} H_{Mm-k}^{(l)}(z) \,.$$

Rearranging the sums yields

$$Y_k(z) = \sum_{l=0}^{M-1} \left[\frac{1}{M} \sum_{i=0}^{M-1} X(z^{1/M} W_M^i) \left(z^{1/M} W_M^i\right)^{-(k+l)} \right] H_{Mm-k}^{(l)}(z) \,,$$

where the term in square brackets, by comparison to (2.6), can be identified as the M-fold decimation of the time-shifted input signal $X(z)$, thus

$$Y_k(z) = \sum_{l=0}^{M-1} \left[X(z) z^{-(k+l)} \right]_{\downarrow M} H_{Mm-k}^{(l)}(z) \,. \tag{2.8}$$

This equation specifies the output signal $y_k[m] = y[Mm - k]$ of channel k as the sum of time-shifted and decimated versions of the input signal filtered by polyphase components. In a MIMO LTV filter, the input $x[n]$ is provided in blocks of M samples, thus the channel input signals are identified as

$$x_i[m] = x[Mm - i] \,, \tag{2.9}$$

with the corresponding z-transform

$$X_i(z) = \left[X(z) z^{-i} \right]_{\downarrow M} \,,$$

for the channels $i = 0, \ldots, M-1$. In order to map the channel inputs to the time-shifted and decimated input signals in (2.8), the delay $z^{-(k+l)}$ in (2.8) is considered, which is between $z^{-2(M-1)}$ and z^0 due to the range of k and l. By comparison to the definition of $X_i(z)$, it is observed that

$$\begin{aligned} k + l \leq M - 1 \quad &\Rightarrow \quad [X(z) z^{-(k+l)}]_{\downarrow M} = X_{k+l}(z) \\ k + l > M - 1 \quad &\Rightarrow \quad [X(z) z^{-(k+l)}]_{\downarrow M} = [X(z) z^{-(k+l-M)} z^{-M}]_{\downarrow M} = X_{k+l-M}(z) z^{-1} \,. \end{aligned}$$

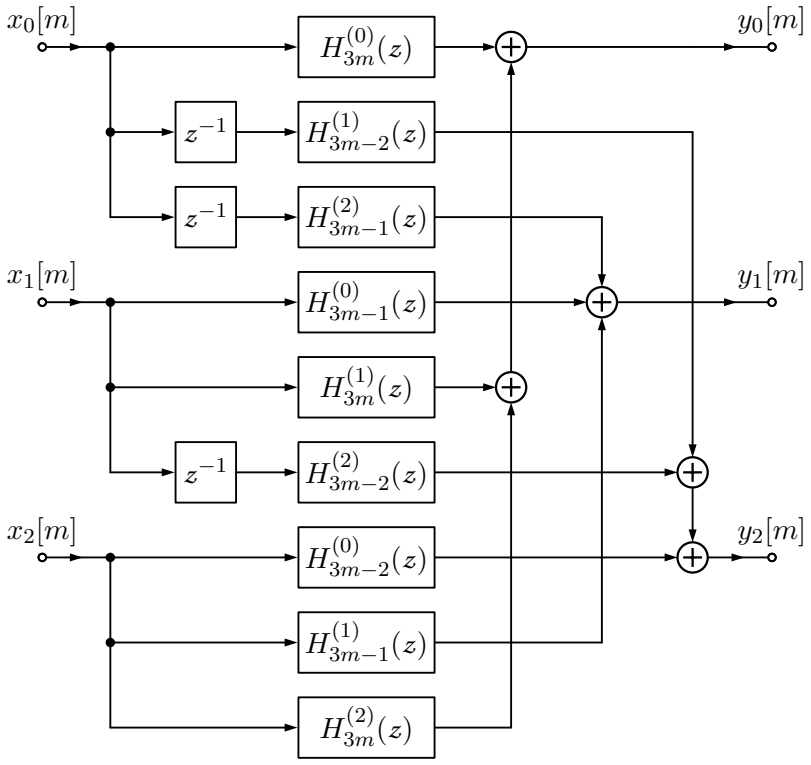


Figure 2.4: MIMO LTV filter for $M = 3$ described by (2.10).

Consequently, (2.8) can be expressed in terms of the channel input signals $X_i(z)$ as

$$Y_k(z) = \sum_{l=0}^{M-1} X_{((k+l))_M}(z) z^{-\lfloor (k+l)/M \rfloor} H_{Mm-k}^{(l)}(z) \,, \tag{2.10}$$

for $k = 0, \ldots, M-1$, where $\lfloor \cdot \rfloor$ denotes the floor function and $((k+l))_M = (k+l) \bmod M$ denotes the modulo operation, yielding a value beween 0 and $M-1$ (cf. [11]). Equation (2.10) specifies the output of channel k, i.e., $Y_k(z)$, in terms of the channel input signals $X_i(z)$ and, therefore, can be regarded as the *design equation* for the structure of the MIMO LTV filter. Indeed, (2.10) is the generalization of the design equations for LTI filters in [7] to LTV filters.[2] The MIMO LTV filter described by (2.10) is shown in Figure 2.4 for $M = 3$. It should be pointed out that the MIMO structure comprises exactly M times the multipliers and adders of the SISO LTV filter and, therefore, exhibits the *same* number of arithmetic operations per unit time as the SISO LTV filter due to the parallel processing of M samples. This implies that the transformation does not introduce any additional computational effort on a per-sample basis. Furthermore, due to the polyphase decomposition, the individual subfilters in the MIMO structure comprise only $1/M$th of the SISO LTV filter coefficients and as the transformation places at maximum $M-1$ adders between the subfilters and the channel outputs, the critical path [7] is reduced as well if the order of the SISO LTV filter is at least M.[3] Finally, in the

[2] Note that the design equation in [7] is put in a more extensive way without the use of a floor and modulo operation. However, it was found that the formulation in (2.10) is more convenient, as one can create a table comprising k, l, $((k+l))_M$, and $\lfloor (k+l)/M \rfloor$, from which the structure is directly obtained.

[3] As in the derivation, a direct form implementation of the FIR filters is assumed.

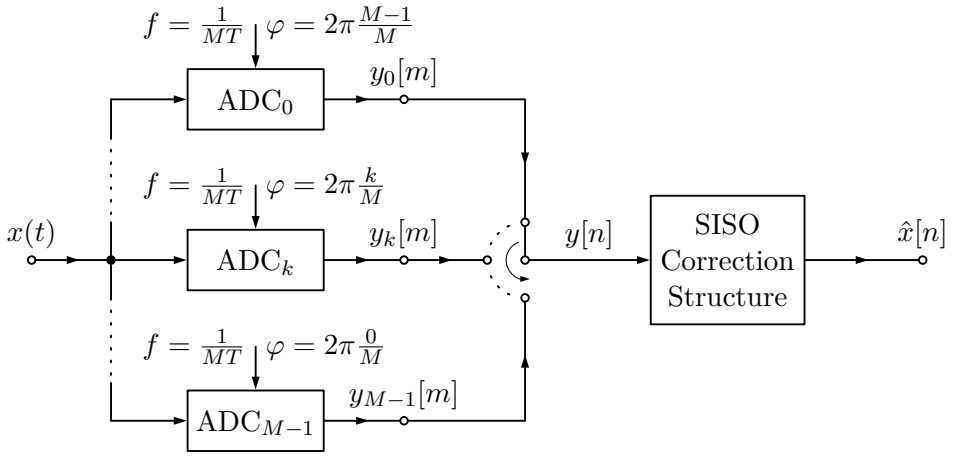


Figure 2.5: TI-ADC (cf. [15]) followed by a SISO correction structure.

special case of a linear and M-periodically time-varying SISO filter, i.e., $H_n(z) = H_{n-M}(z)$, the filters in (2.10) become time-invariant, i.e., $H^{(l)}_{Mm-k}(z) = H^{(l)}_k(z)$ [8, 10].
As a concluding remark, note that the delay chain at the output in Figure 2.3 is acausal and, therefore, not realizable. However, this is easily overcome by adding a delay z^{-M+1} at the output, which translates to the obvious fact that an output block is obtained as soon as a complete input block is available.

2.3 Application Examples

The correction of errors in TI-ADCs shall serve here as a basis for practical examples. Aimed at high data rate applications, TI-ADCs [12] comprise M parallel channel ADCs working at the rate $1/(MT)$, which are time-interleaved to yield samples of the input signal at the rate $1/T$, where T denotes the sampling period. Mismatches between the channel ADCs and clock skew significantly degrade the performance [13–15] and various correction methods were already proposed [1, 3, 4, 16–18]. These approaches require the data stream at the full rate $1/T$, i.e., the samples of the M channel ADCs are time-interleaved and, subsequently, fed into a SISO correction structure as shown in Figure 2.5, where the time-interleaving of samples is visualized via a commutator [10]. However, in order to mitigate the requirements on the speed of the hardware, it is desirable to have MIMO correction structures, which accept the direct output of the channel ADCs and work at the M-times lower rate $1/(MT)$ as illustrated in Figure 2.6.
Many digital correction structures are linear filters and the design of the TI-ADC suggests an underlying M-periodicity, but as those structures are mostly adapted, e.g., to the influence of temperature and voltage variations or aging effects, they can be regarded as general LTV filters. Therefore, parallel processing may be obtained by applying the SISO to MIMO transformation discussed in Section 2.2. In this context, note that parallel processing of M samples is the most obvious choice for an M channel TI-ADC, but the degree of parallelism is not linked to the number of channels.
The design equation (2.10) can be readily applied to correction structures based on adapted FIR filters, e.g., explicitly designed correction filters [1]. Focusing on more involved examples,

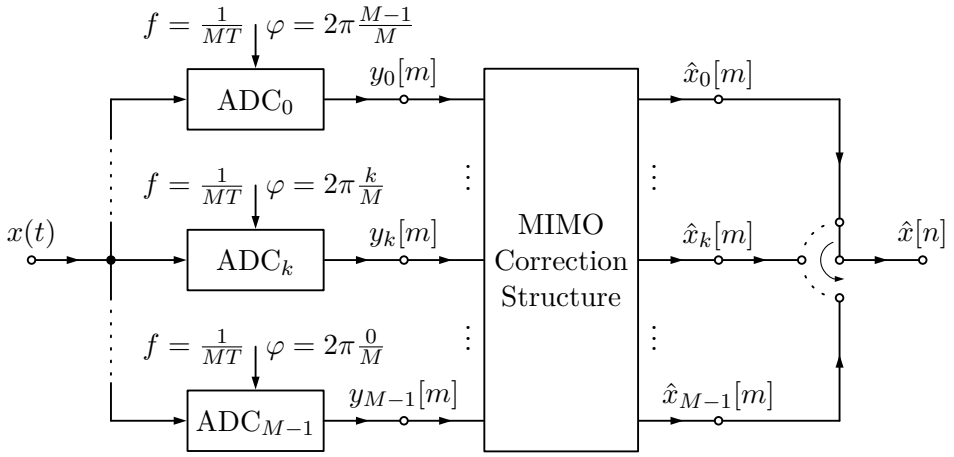


Figure 2.6: TI-ADC followed by a MIMO correction structure. Note that the sampling progression during one iteration begins at ADC_{M-1} and ends at ADC_0 for consistence with the previously introduced notation.

the SISO to MIMO transformation is discussed below for the widely applicable Farrow filter, followed by a general view of iterative correction structures and a specific realization thereof, the differentiator-multiplier cascade.

2.3.1 Farrow Filter

Farrow filters [19] are FIR filters with a parameter λ_n, where the variation of the impulse response coefficients $h_n[k]$ with respect to λ_n is approximated with polynomials of degree P, i.e.,

$$h_n[k] = \sum_{p=0}^{P} b_p[k]\lambda_n^p \ . \tag{2.11}$$

Therein, the subscript n denotes the dependence on the time index n and $b_p[k]$, for $p = 0, \ldots, P$, are the coefficients of the polynomial for the coefficient $h_n[k]$ of the impulse response. It can be shown that the z-transform of (2.11) evaluates to

$$H_n(z) = \sum_{p=0}^{P} B_p(z)\lambda_n^p \ , \tag{2.12}$$

where

$$B_p(z) = \sum_{k=-\infty}^{\infty} z^{-k} b_p[k] \ .$$

A Farrow filter with input $x[n]$ and output $y[n]$ can be regarded as an LTV filter as depicted in Figure 2.1. Hence, a corresponding M-channel MIMO Farrow filter is described by the design equation (2.10), if the polyphase components $H_{Mm}^{(l)}(z)$ defined in (2.7) of the Farrow filter are known. To arrive at this polyphase decomposition, the decomposition of the time-invariant

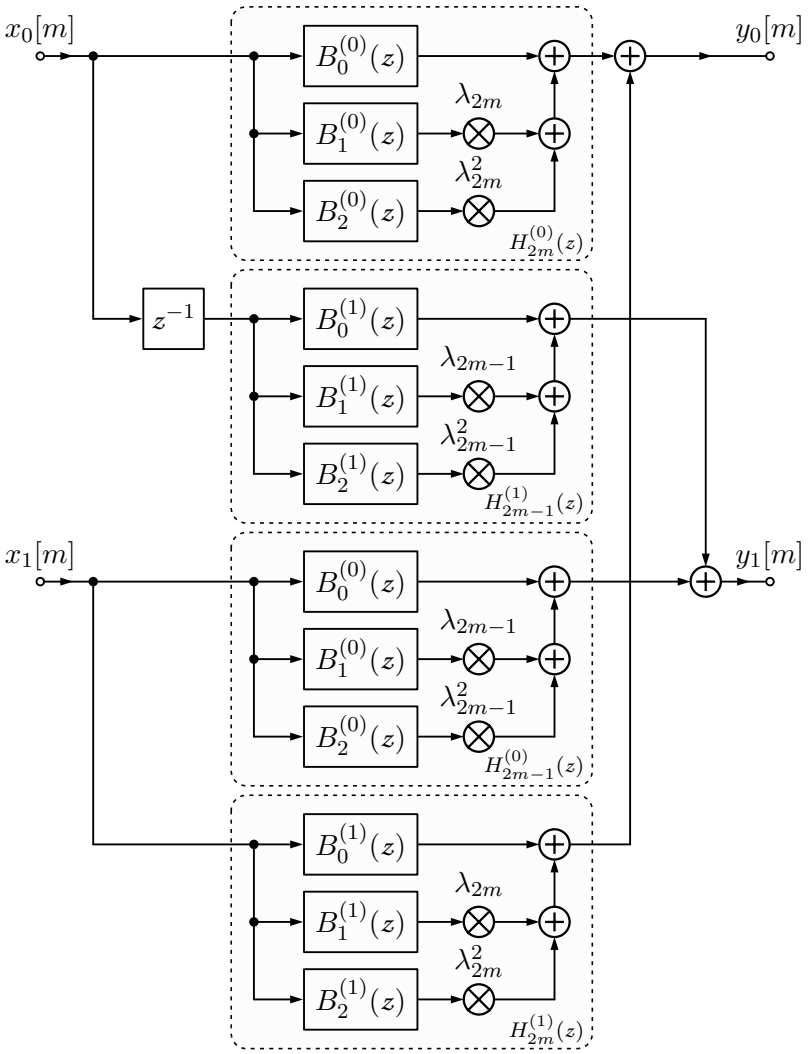


Figure 2.7: MIMO Farrow filter for $M = 2$ and $P = 2$.

polynomial filters $B_p(z)$ [10] is considered first, i.e.,

$$B_p(z) = \sum_{l=0}^{M-1} z^{-l} B_p^{(l)}(z^M) \,, \tag{2.13}$$

where the polyphase components are

$$B_p^{(l)} = \sum_{m=-\infty}^{\infty} z^{-m} b_p^{(l)}[m]$$

with the corresponding impulse responses

$$b_p^{(l)}[m] = b_p[Mm + l] \,. \tag{2.14}$$

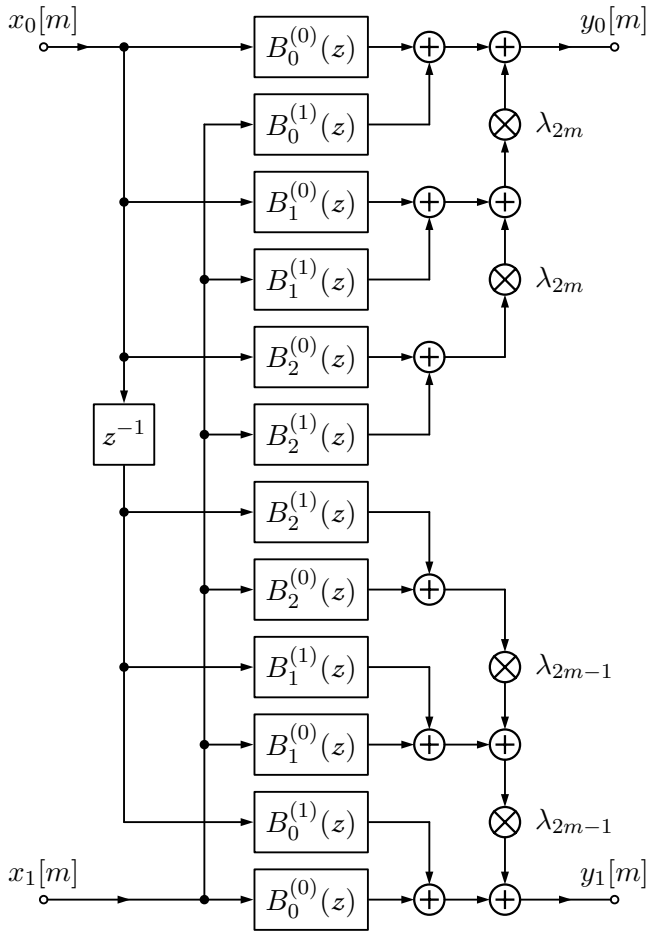


Figure 2.8: MIMO Farrow filter for $M = 2$ and $P = 2$, reorganized to reduce the number of time-varying multipliers.

Using (2.13) in (2.12) gives

$$H_n(z) = \sum_{l=0}^{M-1} z^{-l} \sum_{p=0}^{P} B_p^{(l)}(z^M) \lambda_n^p \,, \tag{2.15}$$

where no delays and time-varying multipliers were interchanged as λ_n^p follows the polynomial filters. A comparison of (2.15) to the polyphase decomposition in (2.3) identifies $\tilde{H}_n^{(l)}(z^M)$ for the Farrow filter as

$$\tilde{H}_n^{(l)}(z^M) = \sum_{p=0}^{P} B_p^{(l)}(z^M) \lambda_n^p \,.$$

The particularities of decimating a time-varying filter as discussed in Section 2.2 apply here as well, but as only the multipliers λ_n^p are time-varying, it is sufficient to take care of their time indices. Consequently, the polyphase components after decimation are given by

$$H_{Mm}^{(l)}(z) = \sum_{p=0}^{P} B_p^{(l)}(z) \lambda_{Mm}^p \,. \tag{2.16}$$

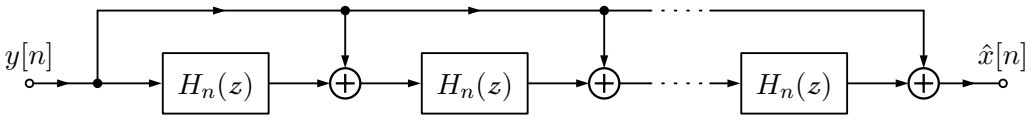


Figure 2.9: SISO cascade correction structure.

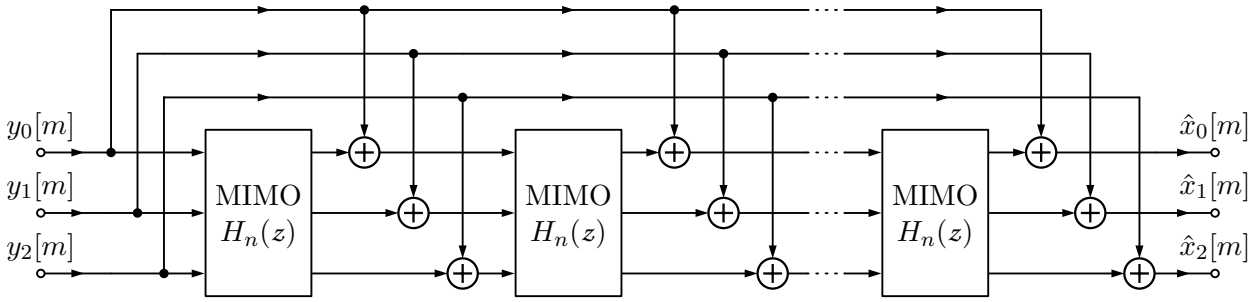


Figure 2.10: MIMO cascade correction structure, where the MIMO equivalent of $H_n(z)$ is illustrated in Figure 2.4.

Concluding, using $H_{Mm}^{(l)}(z)$ in (2.16) in the design equation (2.10) yields the description of an M-channel MIMO Farrow filter. The resulting structure is shown in Figure 2.7 for $M = 2$ and $P = 2$. This figure is an exact graphical analogon of the design equation (2.10) with $H_{Mm}^{(l)}(z)$ as defined in (2.16). However, this structure may be reorganized to reduce the number of time-varying multipliers, which is illustrated in Figure 2.8. It is important to observe that the reorganized MIMO Farrow filter exhibits the same number of additions as well as time-invariant and time-varying multiplications per unit time as the the SISO Farrow filter and, therefore, the transformation does not introduce any additional computational effort on a per-sample basis. Analogous to the direct form SISO LTV filter discussed in Section 2.2, it can be shown that the the critical path [7] is reduced if all polynomial filters of the SISO Farrow filter are at least of order M and implemented in direct form.

2.3.2 Iterative Correction Structures

Correction structures based on iterative methods comprising a cascade of correction stages are recent and efficient strategies for the correction of mismatch errors in TI-ADCs [4, 5, 18]. Here, only the fundamental structure of those approaches shall be considered, which is depicted in Figure 2.9, where $H_n(z)$ is a direct form FIR LTV filter. Note that the notation is adapted to [4], i.e., the input signal to the correction structure is denoted $y[n]$, which is the output of the TI-ADC and itself the sampled analog input signal $x(t)$ impaired by mismatches and clock skew, and the output signal is denoted $\hat{x}[n]$, being the reconstructed signal approximating the ideal output $x[n] = x(nT)$ (cf. Figure 2.5). Once the polyphase decomposition of $H_n(z)$ described by (2.7) is known, the design equation (2.10) provides the structure of the MIMO representation of $H_n(z)$. Therewith, the filters only need to be connected accordingly to obtain the MIMO system as illustrated in Figure 2.10 for $M = 3$.

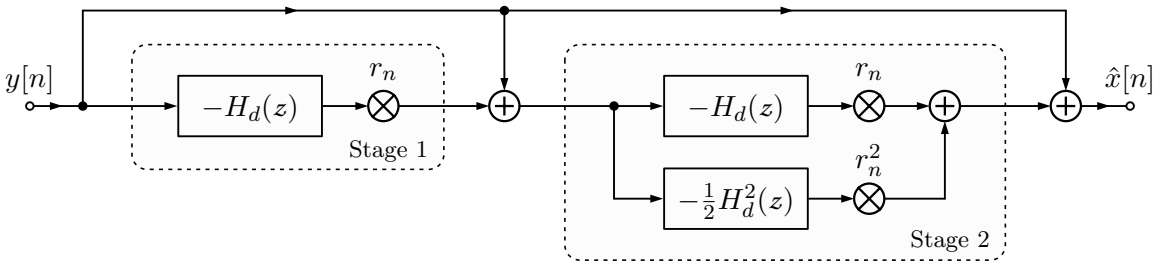


Figure 2.11: DMC with 2 stages.

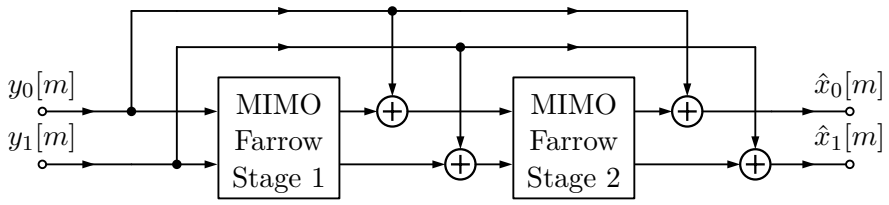


Figure 2.12: MIMO DMC with 2 stages for $M = 2$, where the MIMO Farrow filters are illustrated in Figure 2.8.

2.3.3 Differentiator-Multiplier Cascade

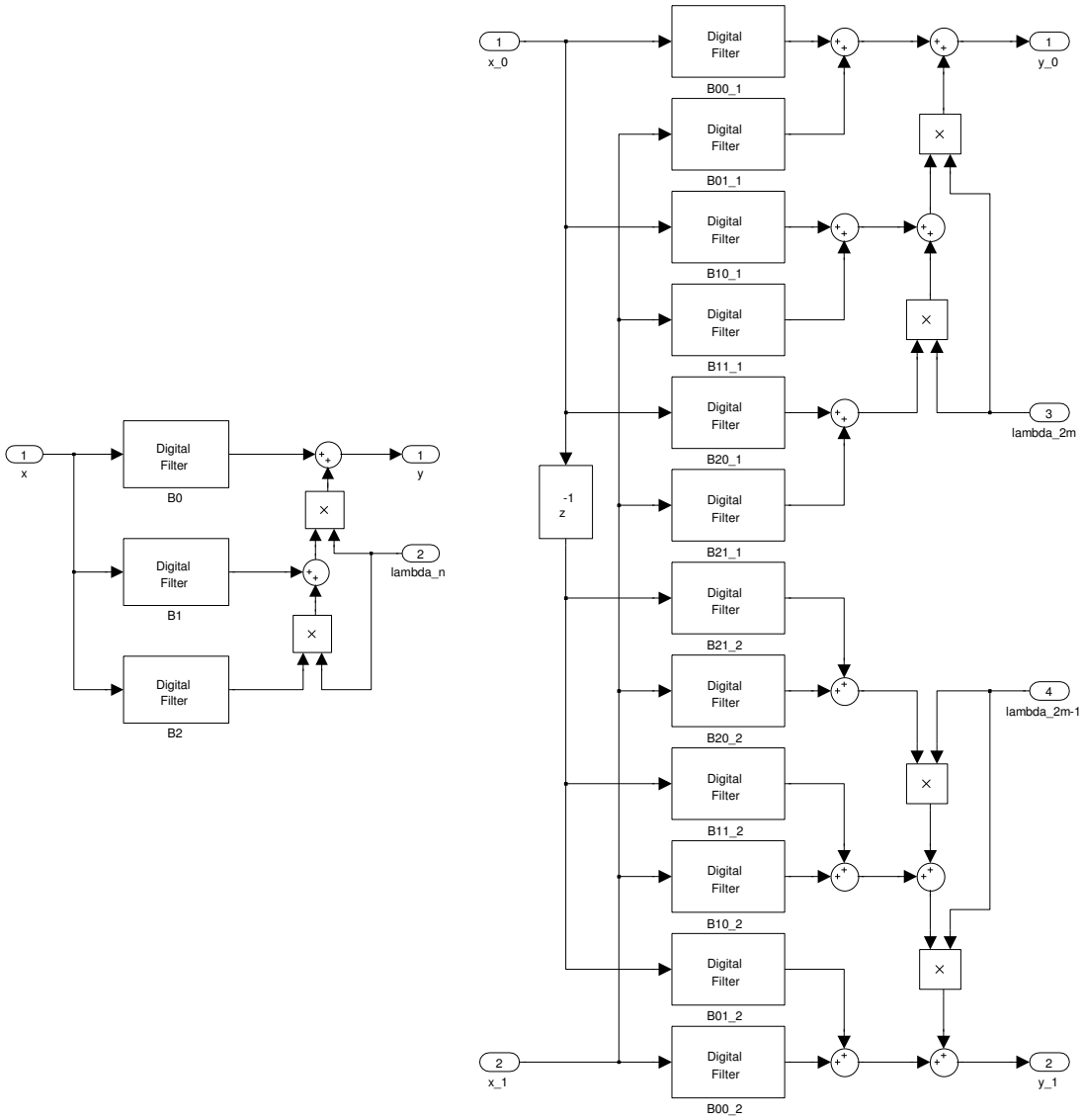
The differentiator-multiplier cascade (DMC) introduced in [16] is an iterative correction structure for timing mismatch correction based on a Taylor series expansion. Again, only its structure shall be considered and serve as a particular example of an iterative correction structure. Furthermore, the notation of Section 2.3.2 is adopted, which is used in [16] as well.
A DMC with two stages is illustrated in Figure 2.11, where $H_d(z)$ is an ideal first-order discrete-time differentiator, i.e., its output is the first derivative of its input, and r_n is the time-varying sampling time error in fractions of the sampling period T. The structure of the stages can be identified as Farrow filters with $\lambda_n = r_n$ and

- $P = 1$, $B_0(z) = 0$, $B_1(z) = -H_d(z)$ for stage 1, and
- $P = 2$, $B_0(z) = 0$, $B_1(z) = -H_d(z)$, $B_2(z) = -H_d^2(z)/2$ for stage 2.

Consequently, the polyphase decomposition of the stages is given by (2.16). The stages themselves are connected according to Figure 2.9, whose MIMO respresentation was discussed in Section 2.3.2. Concluding, a 2-channel MIMO DMC with 2 stages is shown in Figure 2.12, whose MIMO stages are depicted in Figure 2.8.

2.4 Implementation and Practical Aspects

The theoretical results for the Farrow filter and DMC discussed in Section 2.3 were verified by a simulation in MathWorks® Simulink®. In the following, these simulations shall be discussed briefly, as they provide an insight into some issues which arise in practical implementations.



(a) SISO Farrow filter for $P = 2$. (b) MIMO Farrow filter for $M = 2$ and $P = 2$.

Figure 2.13: SISO and MIMO Farrow filter implemented as a subsystem in MathWorks® Simulink®.

2.4.1 Farrow Filter

The implementation of the SISO and MIMO Farrow filter is fairly straightforward in Simulink®, which is visible in Figure 2.13. The filter order of the polynomial filters $b_i[m]$, with $i = 0, 1, 2$, is chosen as $N = 32$ and the input signal $x[n]$, the parameters λ_n as well as the filter coefficients $b_i[k]$ are drawn from a uniform distribution on the open interval $(-1, 1)$. The sequences $x[n]$ and λ_n need to be decomposed into two subsequences for the MIMO Farrow filter with $M = 2$, e.g., the input signal vector `x` for the Simulink® model, which is generated in Matlab®, is split into `x(1:2:end)` for $x_1[n]$ and `x(2:2:end)` for $x_0[n]$, where it should be observed that $x_0[n]$ comprises the more recent samples in the input blocks compared to $x_1[n]$, cf. (2.9). In contrary, when the impulse responses of the polynomial filters are decomposed into their polyphase components, the temporal ordering with respect to the polyphase index is just the opposite, e.g., the impulse response vector `b0` for $b_0[m]$ is decomposed into `b0(1:2:end)` for $b_0^{(0)}[m]$ and `b0(2:2:end)` for

$b_0^{(1)}[m]$, cf. (2.14). The equivalence of the output of the SISO and MIMO Farrow Simulink® model is determined by analyzing the error, i.e., the difference of the outputs, in terms of a signal-to-noise ratio (SNR). If the output of the SISO Farrow filter is denoted by $y_{\mathrm{SISO}}[n]$ and the output signal of the MIMO Farrow filter, which results from time-interleaving the output signals $y_0[m]$ and $y_1[m]$, is denoted by $y_{\mathrm{MIMO}}[n]$, then

$$\mathrm{SNR} = 10 \cdot \log\left(\frac{\sum_{n=0}^{N-1}|y_{\mathrm{SISO}}[n]|^2}{\sum_{n=0}^{N-1}|e[n]|^2}\right) ,$$

where the error signal $e[n]$ is defined as

$$e[n] = y_{\mathrm{SISO}}[n] - y_{\mathrm{MIMO}}[n]$$

and the number of samples is set to $N = 2^{12}$. The resulting SNR is about 312 dB, which represents an extremely minor deviation of the outputs and can be attributed to numerical issues due to the simulation with finite precision. Consequently, the output signals are equal and demonstrate the correctness of the SISO to MIMO transformation.

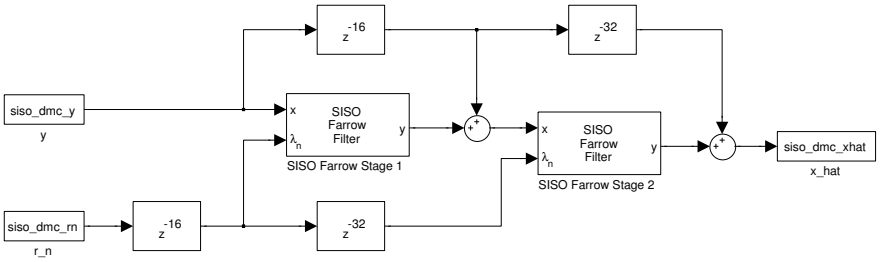
2.4.2 Differentiator-Multiplier Cascade

The SISO and MIMO DMC with 2 stages were implemented in Simulink® using the SISO and MIMO Farrow filter model of Section 2.4.1, respectively, and they are depicted in Figure 2.14. Instead of comparing the output of these models, they are compared to the output of the implementation of Tertinek and Vogel[4], which they provide alongside their paper on the DMC [16] and is called "reference implementation" in the discussion below. Following the exemplary simulation of Tertinek and Vogel [16],[5] a white noise signal, band limited to 0.7 times the Nyquist frequency, is sampled to form the input signal. The standard deviation of the zero-mean normally distributed timing jitter is chosen to $0.01T$, where T is the ideal sampling period, and the first-order differentiator is designed using the Parks-McClellan algorithm [11] (`firpm` in Matlab®) with a filter order $N = 32$.[6] To replicate the behavior of the reference implementation, the impulse response of the second-order differentiator utilized in the second stage is obtained by convolving the impulse response of the first-order differentiator with itself. Note that this increases its filter order to $2N = 64$ and, consequently, the output of the first-order differentiator used in the second stage needs to be delayed by $N/2 = 16$ samples, as otherwise the output of the two differentiators would be misaligned. One may have questioned the additional delays in Figure 2.14, which are not present in the structures in Section 2.3.3. In order to motivate their insertion, the differentiators need to be considered. In Section 2.3.3, they are assumed to be ideal and, therefore, do not delay the signal. However, ideal differentiators are acausal and, consequently, not realizable. A practical implementation of a differentiator does not only differentiate but also delay the signal. Thus, the delay introduced by the subfilters in the stages needs to be considered when their output is multiplied with the time-varying parameter r_n and

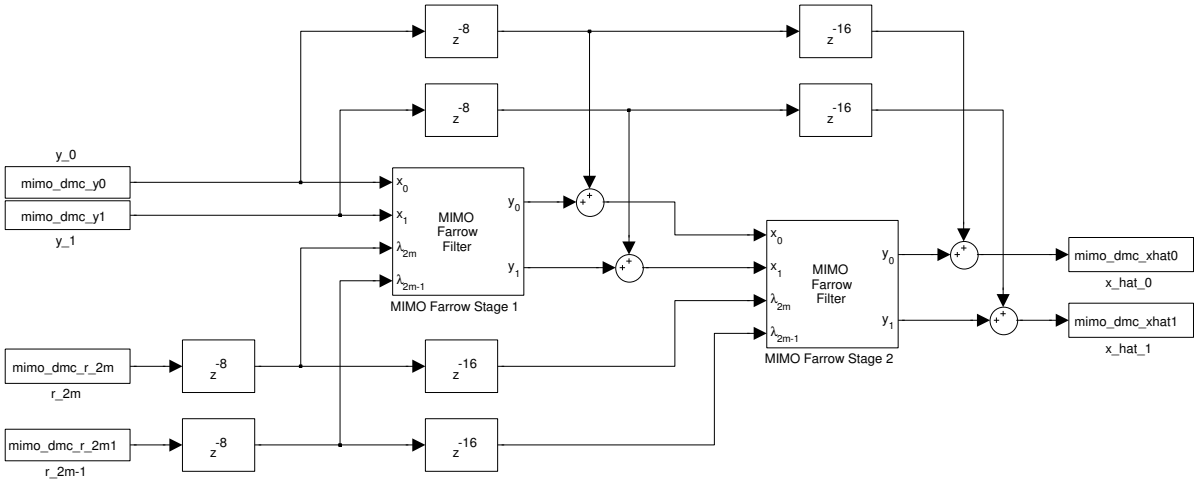
[4] `http://userver.ftw.at/~vogel/Code/DMC.m`

[5] `http://userver.ftw.at/~vogel/Code/Demo.m`

[6] The reason for using a band limited input signal originates from the issue of designing a first-order FIR differentiator with an integer delay. More information on this issue is provided in Section VIII of [16].



(a) SISO DMC utilizing the SISO Farrow filter in Figure 2.13a.



(b) MIMO DMC for $M = 2$ utilizing the MIMO Farrow filter in Figure 2.13b.

Figure 2.14: SISO and MIMO DMC with 2 stages implemented in MathWorks® Simulink®.

when this result is added to the input signal, cf. [16]. In case of the MIMO DMC, the delay must be implemented with block processing in mind. Assuming the signal in a channel should be delayed by D samples, then it suffices to delay all channel signals by D/M samples if D is a multiple of M, i.e., $D \bmod M \equiv 0$. However, if $D \bmod M \neq 0$, the channels need to be "cross-connected" to realize the delay, i.e., to delay the signal in channel i by D samples involves delaying it by $\lfloor (i + D)/M \rfloor$ samples and connecting it to the channel $(i + D) \bmod M$ of the subsequent structure. Fortunately, the former is the case for the MIMO DMC discussed here, hence it amounts to simply inserting the corresponding delays as visible in Figure 2.14b.
The equivalence of the output of the SISO and MIMO DMC Simulink® model with respect to the reference implementation is analyzed analogously to Section 2.4.1, i.e., their SNR with respect to the reference output is considered. For both, the SISO and MIMO DMC, the resulting SNR is about 321 dB. Again, this extremely minor deviation of the outputs can be attributed to numerical issues and, consequently, the output signals are equal and demonstrate the correctness of the SISO and MIMO DMC model.

2.5 Summary

In this chapter, a systematic approach for parallel processing with SISO LTV filters was introduced by means of a transformation to a MIMO LTV filter using multirate theory and polyphase decomposition. The structure of the MIMO LTV filter is described by a design equation for an

arbitrary degree M of parallelism using the polyphase decomposition of the SISO LTV filter. It was shown that the transformation does not introduce any additional computational effort, as the number of arithmetic operations per sample is not affected, and under certain conditions, it even reduces the critical path. Compared to the design equation for parallel processing with LTI filters described in [7], the proposed design equation is its generalization to LTV filters. The practical application and benefit of the transformation was discussed for a Farrow filter, a general iterative correction structure, and a differentiator-multiplier cascade, and it was demonstrated that even for these complex structures, the invariance of the computational effort per sample with respect to the transformation can be preserved. Finally, the verification of the method via simulation was discussed and important aspects regarding practical implementations were highlighted.

Chapter 3

Equalization of Time-Varying Nonlinear Systems

3.1 Introduction

The equalization of nonlinear systems, i.e., cancelling the impact a nonlinear system has on its input signal, is required in many applications, e.g., amplifier predistortion [20–23], nonlinear echo cancellation [24–27] and channel equalization [28–30]. In this work, it is assumed that the nonlinearity can be modeled by a Volterra series [31–34], which is a commonly used approximator for weak nonlinearities, i.e., continuous nonlinearities with fading memory [35,36]. As real-world nonlinear systems often vary with time, e.g., due to environmental influences like temperature variations or changes in the channel, the Volterra series is assumed to be time-varying. The fast-paced and continuous advances in semiconductor technology initiated a shift of most signal processing tasks into the digital domain [37]. This fact is respected in the following by considering discrete-time equalizers, where the term *equalizer* refers to any system that cancels the impact of a nonlinear system. An equalizer may be placed in front of a nonlinear system, which is commonly known as *predistortion*, or behind it, which is termed *post-equalization* and illustrated in Figure 3.1. Due to the origin of the proposed equalizers, this work devotes itself to post-equalization methods and, for convenience, the term *equalizer* is used to refer to post-equalization in the remainder of this text.

In the following, this chapter continues in Section 3.2 by introducing the time-varying discrete-time Volterra series and defining the employed notation. Subsequently, Section 3.3 provides an overview of existing equalization methods and discusses two particular ones in more detail, which are later on used for a comparison. The derivation and theoretical treatment of the proposed equalization structures is covered in Section 3.4, 3.5, and 3.6, in which a novel view on the Volterra series is established and, subsequently, utilized to derive and analyze equalization structures. These theoretical findings are put into operation in Section 3.7 by means of a simulation in MathWorks® Matlab®, where the proposed equalizers are compared to existing methods. Finally, Section 3.8 concludes the chapter with a summary.

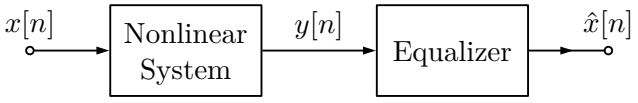


Figure 3.1: Post-equalization of a nonlinear system, in which $\hat{x}[n]$ is a reconstruction of $x[n]$.

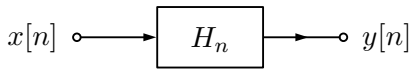


Figure 3.2: Volterra system H_n.

3.2 Time-Varying Discrete-Time Volterra Series

The time-varying discrete-time Volterra series depicted in Figure 3.2, for convenience referred to as *Volterra system* in the remainder, which produces the output sample $y[n]$ given the input signal $x[n]$ can be defined at time instant n as [6]

$$y[n] = H_n\{x[n]\} \,, \tag{3.1}$$

in which the time-varying Volterra system *operator* H_n is given by

$$H_n\{x[n]\} = \sum_{p=1}^{\infty} H_{p,n}\{x[n]\} \tag{3.2}$$

and the time-varying pth-order Volterra operators $H_{p,n}$ are

$$H_{p,n}\{x[n]\} = \sum_{k_1,\ldots,k_p \in \mathbb{Z}} h_{p,n}[k_1,\ldots,k_p] \prod_{i=1}^{p} x[n-k_i] \,, \tag{3.3}$$

with $h_{p,n}$ being the time-varying Volterra kernels.[1] The subscript n denotes the time dependence and it shall be pointed out that the time dependence of the Volterra kernels $h_{p,n}$ is the only difference to the time-invariant Volterra series [31, 32, 34].
The notation in (3.1) to (3.3) suites some situations very well, but sometimes it is unnecessarily bulky. However, especially to accomplish a more homogeneous notation with the concepts introduced later, an alternative notation is established, which describes the Volterra system via a vector function. Therefore, an input vector $\mathbf{x} \in \mathbb{R}^{\mathbb{Z}}$ comprising the samples of the input signal $x[n]$, i.e.,

$$\mathbf{x} = (\ldots, x[n-1], x[n], x[n+1], \ldots)^{\mathsf{T}} \,, \tag{3.4}$$

and an output vector $\mathbf{y} \in \mathbb{R}^{\mathbb{Z}}$ comprising the samples of the output signal $y[n]$, i.e.,

$$\mathbf{y} = (\ldots, y[n-1], y[n], y[n+1], \ldots)^{\mathsf{T}} \,, \tag{3.5}$$

is defined, where $\mathbb{R}^{\mathbb{Z}}$ denotes the vector space of bi-infinite real-valued sequences [9]. The Volterra system *function* $\mathcal{H} : \mathbb{R}^{\mathbb{Z}} \to \mathbb{R}^{\mathbb{Z}}$ relates the input vector to the output vector, i.e.,

$$\mathbf{y} = \mathcal{H}(\mathbf{x}) \,, \tag{3.6}$$

[1] Note that in the most general form, the Volterra series includes a term of order 0, i.e., a time-varying offset $h_{0,n}$. To simplify the discussion, it is common to require that the offset is compensated separately and, therefore, it is assumed that $h_{0,n} = 0$ [38]. Furthermore, implementation-relevant kernel reorganizations, e.g., kernel symmetrization or triangularization, are not considered. The interested reader is referred to [31, 33].

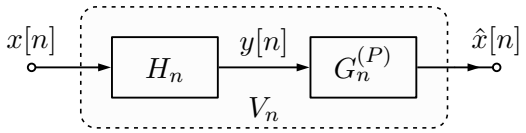


Figure 3.3: Cascade of the nonlinear system H_n and the Pth-order inverse $G_n^{(P)}$. V_n denotes the Volterra system representing the cascade of H_n and $G_n^{(P)}$.

and is defined as

$$\mathcal{H}(\mathbf{x}) = \sum_{p=1}^{\infty} \mathcal{H}_p(\mathbf{x}) \ , \tag{3.7}$$

in which $\mathcal{H}_p : \mathbb{R}^{\mathbb{Z}} \to \mathbb{R}^{\mathbb{Z}}$ are the pth-order Volterra functions. The vector function $\mathcal{H}_p$ can be defined in terms of the time-varying pth-order Volterra operators in (3.3), i.e.,

$$\mathcal{H}_p(\mathbf{x}) = (\ldots, H_{p,n-1}\{\mathbf{x}\}, H_{p,n}\{\mathbf{x}\}, H_{p,n+1}\{\mathbf{x}\}, \ldots)^{\mathsf{T}} \ , \tag{3.8}$$

where $\mathbf{x}$ is used as an alternative notation for the signal $x[n]$ passed to $H_{p,n}$. At first sight, the vector respresentation may take some getting used to, but it is essential to the concepts discussed later on. It is important to observe that if one considers the row for time instant n in the vector notation, it is equivalent to the definition in (3.1) to (3.3), which is utilized in the remainder if the time-dependence has to be made explicit, e.g., in block diagrams.

3.3 Existing Equalization Methods

The equalization of Volterra systems has been considered for quite some time and a common approach is the Pth-order inverse introduced in [39], which allows the construction of an inverse system based on the knowledge of the Volterra system. Another method that utilizes the knowledge of the Volterra system to generate an inverse system is the reformulation as a fixed-point problem, which is subsequently solved using a nonlinear fixed-point iteration [9]. In case a desired response signal is available, adaptive Volterra filters are applicable [38], or, if the input signal stems from a finite symbol set, nonlinear decision feedback [40] or a method based on root-finding [41] may be used.

As the Pth-order inverse and nonlinear fixed-point iteration provide an inverse system given the Volterra system, they are similar in spirit to the equalizers introduced below and, therefore, are suited for a comparison. In the following, a concise overview of these two methods is provided to outline their fundamental properties. More detailed information can be found in the references given in the text.

3.3.1 Pth-Order Inverse

The time-invariant Pth-order inverse was introduced by Schetzen in [39] and is extended to the time-varying case in [6]. In this approach, the Volterra system $\mathcal{H}$ is equalized by another Volterra system $\mathcal{G}^{(P)}$, called the Pth-order inverse, as illustrated in Figure 3.3. Therefore, the

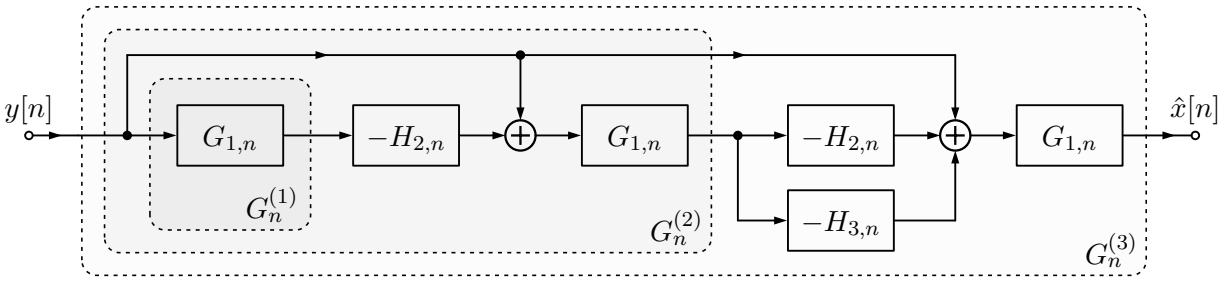


Figure 3.4: 3rd-order inverse $G_n^{(3)}$ based on (3.12).

Algorithm 1 Pth-order inverse

1. Calculate the inverse filter $G_{1,n}$ for the first-order Volterra operator $H_{1,n}$
2. Generate the Pth-order inverse $G_n^{(P)}$ via

$$G_n^{(P)}\{y[n]\} = G_{1,n}\Big\{y[n] - \sum_{p=2}^{P} H_{p,n}\{G_n^{(P-1)}\{y[n]\}\}\Big\} \tag{3.10}$$

3. Apply the Pth-order inverse $G_n^{(P)}$ to the signal $y[n]$ to obtain the reconstruction $\hat{x}[n]$

input $\mathbf{y}$ to the equalizer $\mathcal{G}^{(P)}$ is given by (3.6) and the reconstruction $\hat{\mathbf{x}}$ of $\mathbf{x}$ is

$$\hat{\mathbf{x}} = \mathcal{G}^{(P)}(\mathbf{y}) \, .$$

Alternatively, $\hat{\mathbf{x}}$ may be considered as the response of a Volterra system $\mathcal{V}$ to the input $\mathbf{x}$, where $\mathcal{V}$ comprises the cascade of $\mathcal{H}$ and $\mathcal{G}^{(P)}$, i.e.,

$$\hat{\mathbf{x}} = \mathcal{V}(\mathbf{x}) = \mathcal{G}^{(P)}(\mathcal{H}(\mathbf{x})) \, .$$

The Pth-order inverse is defined in terms of the cascade system $\mathcal{V}$ by requiring that its first-order Volterra function $\mathcal{V}_1$ forwards the input signal and the outputs of the pth-order Volterra functions $\mathcal{V}_p$, for $p = 2, \ldots, P$, are zero, i.e.,

$$\mathcal{V}_p(\mathbf{x}) = \begin{cases} \mathbf{x} \, , & \text{if } p = 1 \\ \mathbf{0} \, , & \text{if } 2 \leq p \leq P \, . \end{cases} \tag{3.9}$$

Therefore, it is required that the first-order Volterra function $\mathcal{H}_1$ is invertible, i.e., it possesses an inverse $\mathcal{G}_1 = \mathcal{H}_1^{-1}$, such that

$$\mathcal{V}_1(\mathbf{x}) = \mathcal{G}_1(\mathcal{H}_1(\mathbf{x})) = \mathbf{x} \, . \tag{3.11}$$

Using the constraints in (3.9), one can recursively find expressions for $\mathcal{G}^{(P)}$ as a function of $\mathcal{G}_1$ and $\mathcal{H}_p$, for $p = 2, \ldots, P$, as discussed in [6, 39]. It is important to observe that a Pth-order inverse is not unique, as the pth-order Volterra functions $\mathcal{V}_p$, for $p > P$, are not constrained by (3.9). This is utilized in [42] to derive a recursive relation to synthesize a Pth-order inverse,

which results in a structure that is less complex compared to the Pth-order inverse of Schetzen. In particular, this recursive relation is given by [6, 42]

$$\mathcal{G}^{(P)}(\mathbf{y}) = \mathcal{G}_1\Big(\mathbf{y} - \sum_{p=2}^{P} \mathcal{H}_p(\mathcal{G}^{(P-1)}(\mathbf{y}))\Big) \tag{3.12}$$

and $\mathcal{G}^{(1)} = \mathcal{G}_1$. The resulting equalizer is explained in Algorithm 1 and a block diagram of a 3rd-order inverse is illustrated in Figure 3.4. A similar recursive relation is presented in [43], which coincides with (3.12), except that the pth-order Volterra functions, for $p > P$, are utilized as well, i.e.,

$$\mathcal{G}^{(P)}(\mathbf{y}) = \mathcal{G}_1\Big(\mathbf{y} - \sum_{p=2}^{\infty} \mathcal{H}_p(\mathcal{G}^{(P-1)}(\mathbf{y}))\Big) \,. \tag{3.13}$$

Therefore, from a structural viewpoint this is more complex than the Pth-order inverse in (3.12), but it will provide interesting insights later on in the text.

The implications of the requirement in (3.11) are rather extensive and are probably more obvious in the operator notation, i.e.,

$$V_{1,n}\{x[n]\} = G_{1,n}\{H_{1,n}\{x[n]\}\} = x[n] \,. \tag{3.14}$$

Therefore, the first-order Volterra operator $H_{1,n}$ has to possess an inverse $H_{1,n}^{-1}$ at every time instant n. In the time-varying case, the inverse $H_{1,n}^{-1}$ cannot be obtained by simply inverting the filter $H_{1,n}$, because it depends on the first-order Volterra operators at other time instants as well. This issue is out of scope for this discussion and treated, e.g., in [1, 2]. However, it shall be pointed out that, in general, the exact inverse does not exist and, therefore, has to be approximated [6]. On the one hand, this implies additional computational complexity due to the filter design, and on the other hand, (3.9) will only be satisfied approximately due to deviation from the exact inverse $H_{1,n}^{-1}$. Furthermore, if two Volterra systems with nonzero kernels up to order Q and S, respectively, are cascaded, the Volterra system representing the cascade of those two systems is of order QS [42]. This implies that for any Pth-order inverse, where P is finite, the cascade exhibits residual Volterra functions of order $p > P$, which introduce new higher-order distortions that may become as severe such that the equalizer even increases the overall distortion [44, 45]. Finally, assuming that an inverse $\mathcal{H}^{-1}$ for the Volterra system $\mathcal{H}$ exists, the Pth-order inverse is only an approximator for $\mathcal{H}^{-1}$ if the functional series represented by the Volterra series $\mathcal{G}^{(P)}$, for $P \to \infty$, converges, which may only hold for a limited range of input amplitudes [39, 44, 46].

3.3.2 Nonlinear Fixed-Point Iteration

The equalization of Volterra systems using a fixed-point approach was pioneered in [47, 48], is discussed thoroughly in [9] and the time-varying case is touched in [6]. In order to pose the equalization problem as a fixed-point problem, a fixed-point equation has to be established.

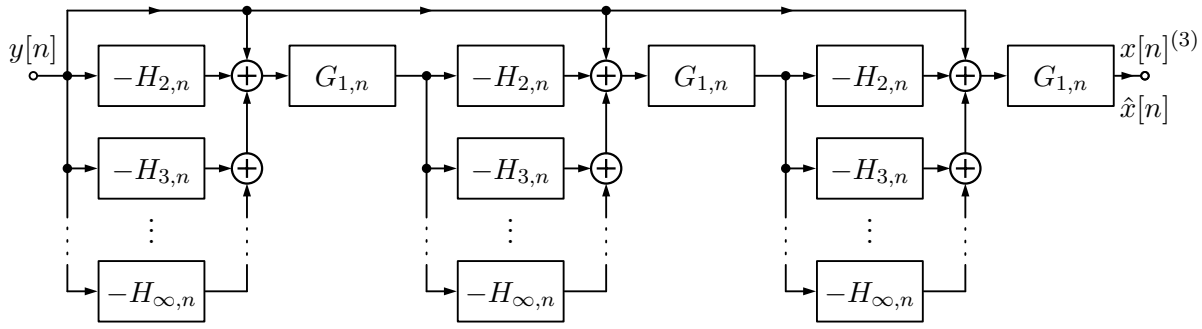


Figure 3.5: Nowak-Van-Veen equalizer with three iterations using the initial approximation $x[n]^{(0)} = y[n]$.

Algorithm 2 Nowak-Van-Veen equalizer

1. Calculate the inverse filter $G_{1,n}$ for the first-order Volterra operator $H_{1,n}$
2. Choose the number R of iterations according to the required equalization performance
3. Iterate the following step for $r = 0, 1, \ldots, R-1$:
   a) Calculate a new approximation $x[n]^{(r+1)}$ of the solution $x[n]$ using

$$x[n]^{(r+1)} = G_{1,n}\Big\{y[n] - \sum_{p=2}^{\infty} H_{p,n}\{x[n]^{(r)}\}\Big\} \tag{3.15}$$

4. Use $x[n]^{(R)}$ as the reconstruction $\hat{x}[n]$ of the input sample $x[n]$

Adding the term $\mathbf{x} - \mathcal{H}_n(\mathbf{x})$ to both sides of (3.6) yields

$$\mathbf{x} = \mathbf{x} + \mathbf{y} - \mathcal{H}(\mathbf{x}) \,, \tag{3.16}$$

and if the left hand side is defined as a vector function $\mathcal{T}(\mathbf{x})$, i.e.,

$$\mathcal{T}(\mathbf{x}) = \mathbf{x} + \mathbf{y} - \mathcal{H}(\mathbf{x}) \,,$$

then (3.16) can be expressed as

$$\mathbf{x} = \mathcal{T}(\mathbf{x}) \,. \tag{3.17}$$

Equations of this kind are called *fixed-point equation* and any $\mathbf{x}$ that satisfies this equation is called a *fixed point*. Consequently, a fixed-point of (3.16) is also a solution to (3.6). A fixed point may be found by successive approximation using a *fixed-point iteration* [49, 50], i.e.,

$$\mathbf{x}^{(r+1)} = \mathcal{T}(\mathbf{x}^{(r)}) \,, \tag{3.18}$$

in which r denotes the iteration index and, if the iteration converges as intended,

$$\lim_{r\to\infty} \mathbf{x}^{(r)} = \mathbf{x} \,.$$

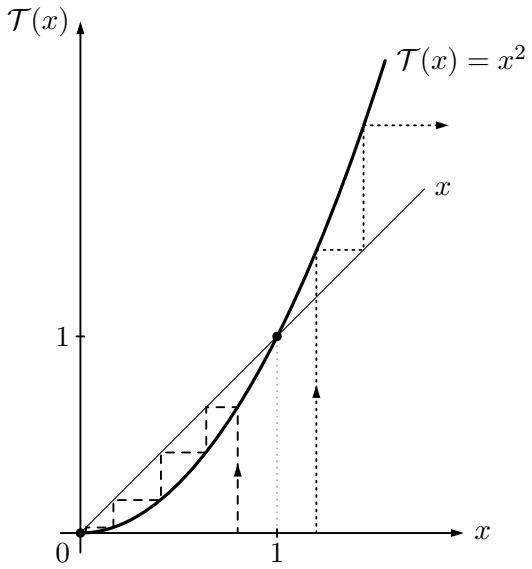


Figure 3.6: Exemplary one-dimensional nonlinear fixed-point iteration $x^{(r+1)} = \mathcal{T}(x^{(r)})$, in which $\mathcal{T}(x) = x^2$. According to (3.21), $\mathcal{T}(\cdot)$ is a contractive mapping on the interval $[0, 1)$ and, indeed, the iteration converges to the fixed point $x = 0$, cf. the dashed trajectory. In contrast, on the interval $[1, \infty)$, the mapping $\mathcal{T}(\cdot)$ is *not* contractive and the iteration diverges for any initial value $x^{(0)}$ that differs from the fixed point $x = 1$, cf. the dotted trajectory.

In the particular case of (3.16), the fixed-point iteration is given by

$$\mathbf{x}^{(r+1)} = \mathbf{x}^{(r)} + \mathbf{y} - \mathcal{H}(\mathbf{x}^{(r)}) \tag{3.19}$$

and is called *nonlinear Richardson iteration* [49]. The approach pursued in (3.16) is just one particular way to obtain a fixed-point equation of the form (3.17). For example, if the first-order Volterra function $\mathcal{H}_1$ possesses an inverse $\mathcal{G}_1 = \mathcal{H}_1^{-1}$, as it is required in (3.11), one can use (3.7) in (3.6) to obtain

$$\mathcal{H}_1(\mathbf{x}) = \mathbf{y} - \sum_{p=2}^{\infty} \mathcal{H}_p(\mathbf{x})$$

and apply $\mathcal{G}_1$, which yields

$$\mathbf{x} = \mathcal{G}_1\Big(\mathbf{y} - \sum_{p=2}^{\infty} \mathcal{H}_p(\mathbf{x})\Big) .$$

Therefore, the corresponding fixed-point iteration is given by

$$\mathbf{x}^{(r+1)} = \mathcal{G}_1\Big(\mathbf{y} - \sum_{p=2}^{\infty} \mathcal{H}_p(\mathbf{x}^{(r)})\Big) , \tag{3.20}$$

which was derived by Nowak and Van Veen in [9]. The resulting equalizer, in the remainder referred to as Nowak-Van-Veen equalizer, is explained in Algorithm 2 and a block diagram of an equalizer based on 3 iterations is illustrated in Figure 3.5. It is remarkable that the fixed-point iteration in (3.20) has the same structure as the Pth-order inverse in (3.13). Indeed, assuming

that the initial approximation is chosen as $\mathbf{x}^{(0)} = \mathbf{y}$, the Pth iteration $\mathbf{x}^{(P)}$ of (3.20) provides the same approximation to the solution $\mathbf{x}$ as the application of the Pth-order inverse defined in (3.13).

Fixed-point iterations as described by (3.18) only converge to a fixed-point $\mathbf{x}$ if $\mathcal{T}$ satisfies certain conditions. In particular, these conditions are described by the *Banach fixed-point theorem* [50], which requires that $\mathcal{T}$ is a *contractive mapping*, i.e.,

$$\|\mathcal{T}(\mathbf{x}_1) - \mathcal{T}(\mathbf{x}_2)\| \leq \alpha \|\mathbf{x}_1 - \mathbf{x}_2\| \,, \tag{3.21}$$

where $\|\cdot\|$ denotes a vector norm [9], $\alpha \in [0, 1)$, and $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{\mathbb{Z}}$. For an intuitive understanding of this requirement, a very simple example is illustrated in Figure 3.6. However, in practice it is rather difficult to show that (3.21) is fulfilled, which is addressed in [9].

3.4 Transformation of a Volterra System to an LTV System

In the following, a new perspective on Volterra systems shall be introduced, i.e., it is shown that such a nonlinear time-varying system may be regarded as an LTV system. The Volterra system defined in (3.1) to (3.3) may be written as

$$y[n] = \sum_{p=1}^{\infty} \sum_{k_1,\ldots,k_p \in \mathbb{Z}} h_{p,n}[k_1, \ldots, k_p] \prod_{i=1}^{p} x[n - k_i] \,,$$

which permits the rearrangement

$$y[n] = \sum_{k_1 \in \mathbb{Z}} \left(h_{1,n}[k_1] + \sum_{p=2}^{\infty} \sum_{k_2,\ldots,k_p \in \mathbb{Z}} h_{p,n}[k_1, \ldots, k_p] \prod_{i=2}^{p} x[n - k_i] \right) x[n - k_1] \,.$$

The term in parenthesis may be regarded as a time-varying impulse response $g_{x,n}[k_1]$, i.e.,

$$g_{x,n}[k_1] = h_{1,n}[k_1] + \sum_{p=2}^{\infty} \sum_{k_2,\ldots,k_p \in \mathbb{Z}} h_{p,n}[k_1, \ldots, k_p] \prod_{i=2}^{p} x[n - k_i] \,, \tag{3.22}$$

in which the subscript n denotes the time dependence and the subscript x denotes the dependence on the input signal $x[n]$. Using (3.22), the output signal $y[n]$ can be characterized by a convolution of the input signal $x[n]$ with $g_{x,n}[k_1]$, i.e.,

$$y[n] = \sum_{k_1 \in \mathbb{Z}} g_{x,n}[k_1] x[n - k_1] \,.$$

Therefore, the Volterra system defined in (3.1) to (3.3) can be viewed as an LTV system with the impulse response $g_{x,n}[k_1]$ in (3.22), which is not only time-dependent, but depends on the input signal $x[n]$ as well. This transition of perspective permits the description of the Volterra system as a matrix equation

$$\mathbf{A}_{\mathbf{x}} \mathbf{x} = \mathbf{y} \,, \tag{3.23}$$

in which $\mathbf{x}$ is the input vector and $\mathbf{y}$ the output vector as defined in (3.4) and (3.5), respectively, and the infinite coefficient matrix $\mathbf{A}_\mathbf{x}$ is given by

$$\mathbf{A}_\mathbf{x} = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \ddots \\ \cdots & g_{x,n-1}[1] & g_{x,n-1}[0] & g_{x,n-1}[-1] & \cdots \\ \cdots & g_{x,n}[1] & g_{x,n}[0] & g_{x,n}[-1] & \cdots \\ \cdots & g_{x,n+1}[1] & g_{x,n+1}[0] & g_{x,n+1}[-1] & \cdots \\ \ddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where the subscript $\mathbf{x}$ denotes the dependence on the input vector. If the element in row i and column j of $\mathbf{A}_\mathbf{x}$ is denoted by $a_{ij,x}$, the coefficient matrix is alternatively defined by

$$a_{ij,x} = g_{x,i}[i-j] \,, \tag{3.24}$$

where $i, j \in \mathbb{Z}$. Concluding, (3.23) is a different way to describe the Volterra system in (3.6) and, even more important, it provides a new foundation to derive and analyze equalizers. Finding the system's input vector $\mathbf{x}$ corresponds to the equalization task and, assuming that the coefficient matrix $\mathbf{A}_\mathbf{x}$ is known, $\mathbf{x}$ is found by solving the system of linear equations in (3.23), which is discussed in a vast amount of literature. However, the dependence of the coefficient matrix $\mathbf{A}_\mathbf{x}$ on the input vector $\mathbf{x}$, which is unknown to the equalizer, complicates the matters. In the following, $\mathbf{A}_\mathbf{x}$ is assumed to be known in Section 3.5, which permits the use of iterative methods to solve (3.23), cf. [4]. Based on this insight, the algorithms are modified in Section 3.6 to obtain realizable equalizers.

3.5 Iterative Methods for Solving Systems of Linear Equations

In this section, it is assumed that the coefficient matrix $\mathbf{A}_\mathbf{x}$ and the output vector $\mathbf{y}$ are known and that the input vector $\mathbf{x}$ should be found by solving the system of linear equations in (3.23). For real-time capability of the targeted equalizers, the underlying algorithms are required to operate on a per-sample basis, whereas block processing is not considered here. Many methods to solve systems of linear equations exist, but this requirement significantly reduces the number of applicable ones, because it requires the underlying algorithm to operate row by row. Some iterative methods feature this property and are well established in the literature [49, 51–53]. In the following, the (linear) Richardson iteration and the Jacobi iteration shall be considered. Note that although this is a sensible selection, other iterative methods with this property do exist, e.g., the Richardson and Jacobi iteration with the inclusion of a relaxation parameter [51, 53]. However, a more complex iterative method will lead to a more complex structure of the resulting equalizer and, therefore, the selection of the method is also limited by practical considerations.

3.5.1 Richardson Iteration

The fundamental principle of iterative methods to solve systems of linear equations is similar to the method discussed in Section 3.3.2, i.e., the problem is stated as a fixed-point equation and, subsequently, solved by successive approximation. For the Richardson iteration, $\mathbf{x} - \mathbf{A}_\mathbf{x}\mathbf{x}$

is added to both sides of (3.23), which results in

$$\mathbf{x} = (\mathbf{I} - \mathbf{A}_\mathbf{x})\mathbf{x} + \mathbf{y} \,, \tag{3.25}$$

where $\mathbf{I}$ denotes the *unit matrix*. Therefore, the corresponding fixed-point iteration is given by

$$\mathbf{x}^{(r+1)} = (\mathbf{I} - \mathbf{A}_\mathbf{x})\mathbf{x}^{(r)} + \mathbf{y} \tag{3.26}$$

and called the *Richardson iteration* [49]. A single iteration step may be performed row by row, i.e.,

$$x[i]^{(r+1)} = \sum_{j \in \mathbb{Z}} (\delta_{ij} - a_{ij,x}) x[j]^{(r)} + y[i] \,,$$

in which δ_{ij} is the *Kronecker delta*, i.e.,

$$\delta_{ij} = \begin{cases} 1 \,, & \text{if } i = j \\ 0 \,, & \text{if } i \neq j \,. \end{cases}$$

With (3.24), this iteration can be expressed in terms of the time-varying impulse response $g_{x,n}[k_1]$, i.e.,

$$x[n]^{(r+1)} = x[n]^{(r)} + y[n] - \sum_{k_1 \in \mathbb{Z}} g_{x,n}[k_1] x[n - k_1]^{(r)} \,. \tag{3.27}$$

Under the assumption that this fixed-point iteration converges as intended, which is investigated later on, it will provide the solution

$$\lim_{r \to \infty} x[n]^{(r)} = x[n] \,. \tag{3.28}$$

Note that if $g_{x,n}[k_1]$ is causal, i.e., $g_{x,n}[k_1] = 0$ for all $k_1 < 0$, the reconstruction of $x[n]$ in (3.27) only depends on the reconstruction of previous samples, i.e., $x[k]$ with $k < n$, and, therefore, is indeed realizable and real-time capable. Furthermore, this implies that one may use the reconstruction $\hat{x}[k]$ of the previous samples instead of the approximation $x[k]^{(r)}$ to improve the convergence [6]. However, this leads to a more complex structure of the equalizers and is omitted here to facilitate clarity.

3.5.2 Jacobi Iteration

The Jacobi iteration is a particular *splitting method* [53], i.e., a method based on a matrix splitting

$$\mathbf{A}_\mathbf{x} = \mathbf{U}_\mathbf{x} - \mathbf{V}_\mathbf{x} \,.$$

For the Jacobi iteration, the matrix $\mathbf{U}_\mathbf{x}$ comprises the diagonal elements of $\mathbf{A}_\mathbf{x}$, i.e.,

$$u_{ij,x} = \delta_{ij} a_{ij,x} \,, \tag{3.29}$$

and $\mathbf{V}_\mathbf{x}$ comprises the negated off-diagonal elements of $\mathbf{A}_\mathbf{x}$, i.e.,

$$v_{ij,x} = (\delta_{ij} - 1)a_{ij,x} \ , \tag{3.30}$$

with $i, j \in \mathbb{Z}$. Utilization of this splitting in (3.23) results in

$$\mathbf{U}_\mathbf{x}\mathbf{x} = \mathbf{V}_\mathbf{x}\mathbf{x} + \mathbf{y}$$

and, assuming that $\mathbf{U}_\mathbf{x}$ is invertible, i.e., $\det(\mathbf{U}_\mathbf{x}) \neq 0$, this is equivalent to

$$\mathbf{x} = \mathbf{U}_\mathbf{x}^{-1}\mathbf{V}_\mathbf{x}\mathbf{x} + \mathbf{U}_\mathbf{x}^{-1}\mathbf{y} \ , \tag{3.31}$$

in which $\mathbf{U}_\mathbf{x}^{-1}$ denotes the inverse matrix of $\mathbf{U}_\mathbf{x}$. Consequently, the corresponding fixed-point iteration is given by

$$\mathbf{x}^{(r+1)} = \mathbf{U}_\mathbf{x}^{-1}\mathbf{V}_\mathbf{x}\mathbf{x}^{(r)} + \mathbf{U}_\mathbf{x}^{-1}\mathbf{y} \tag{3.32}$$

and called the *Jacobi iteration* [51]. A single step of the Jacobi iteration may be performed row by row [51], i.e.,

$$x[i]^{(r+1)} = \frac{1}{a_{ii,x}}\Big(y[i] - \sum_{j \in \mathbb{Z}\setminus i} a_{ij,x}x[j]^{(r)}\Big) \ ,$$

and, using (3.24), it can be expressed in terms of the time-varying impulse response $g_{x,n}[k_1]$ as

$$x[n]^{(r+1)} = \frac{1}{g_{x,n}[0]}\Big(y[i] - \sum_{k_1 \in \mathbb{Z}\setminus 0} g_{x,n}[k_1]x[n-k_1]^{(r)}\Big) \ . \tag{3.33}$$

If this fixed-point iteration converges as intended, it will provide the solution in (3.28).

3.5.3 Condition for Convergence

The application of iterative methods like the Richardson and Jacobi iteration is only reasonable if it can be guaranteed that the iteration converges and that it indeed converges to a solution of (3.23). Preliminary to the analysis of the convergence it shall be pointed out that the Richardson and Jacobi iteration share the same fundamental structure of the fixed-point equation, i.e.,

$$\mathbf{x} = \mathbf{M}_\mathbf{x}\mathbf{x} + \mathbf{c}_\mathbf{x} \ , \tag{3.34}$$

and the corresponding fixed-point iteration, i.e.,

$$\mathbf{x}^{(r+1)} = \mathbf{M}_\mathbf{x}\mathbf{x}^{(r)} + \mathbf{c}_\mathbf{x} \ . \tag{3.35}$$

This is verified by comparing these equations to (3.25) and (3.26) for the Richardson iteration, and (3.31) and (3.32) for the Jacobi iteration. The condition for convergence is derived for the iteration in (3.35) by following the proof provided in [51]. Subsequently, this result is adapted to the special case of the Richardson and Jacobi iteration. With $\mathbf{x}$ being a solution to (3.34),

the error of the approximation $\mathbf{x}^{(r)}$ in iteration r can be defined as

$$\mathbf{e}^{(r)} = \mathbf{x} - \mathbf{x}^{(r)} \ .$$

Subtracting (3.35) from (3.34) results in

$$\mathbf{e}^{(r+1)} = \mathbf{M}_{\mathbf{x}}\mathbf{e}^{(r)} \tag{3.36}$$

and unfolding the recursion yields

$$\mathbf{e}^{(r+1)} = \mathbf{M}_{\mathbf{x}}\mathbf{e}^{(r)} = \mathbf{M}_{\mathbf{x}}^{2}\mathbf{e}^{(r-1)} = \ldots = \mathbf{M}_{\mathbf{x}}^{r+1}\mathbf{e}^{(0)} \ .$$

Convergence to the solution $\mathbf{x}$ of (3.23) is achieved if the error $\mathbf{e}^{(r)}$ eventually decays to zero, i.e.,

$$\lim_{r\to\infty} \mathbf{e}^{(r)} = \mathbf{0} \ .$$

Consequently, a necessary condition for convergence for an *arbitrary* initial error $\mathbf{e}^{(0)}$ is given by

$$\lim_{r\to\infty} \mathbf{M}_{\mathbf{x}}^{r} = \mathbf{0} \ . \tag{3.37}$$

A matrix $\mathbf{M}_{\mathbf{x}}$ that fulfills (3.37) is called a *convergent matrix* and satisfies

$$\|\mathbf{M}_{\mathbf{x}}\| < 1 \ , \tag{3.38}$$

in which $\|\cdot\|$ is any matrix norm [51]. In order to find a condition for convergence in terms of the matrix elements, the matrix norm induced by the maximum norm is chosen to evaluate (3.38), because of its beneficial property of row-by-row evaluation. Therefore, the *maximum norm* of some vector $\mathbf{x}$ is defined first [51], i.e.,

$$\|\mathbf{x}\|_{\infty} = \max_{n\in\mathbb{Z}} |x[n]| \ . \tag{3.39}$$

The matrix norm $\|\cdot\|_{\infty}$ *induced* by the maximum norm is given by [51],

$$\|\mathbf{M}_{\mathbf{x}}\|_{\infty} = \max_{i\in\mathbb{Z}} \sum_{j\in\mathbb{Z}} |m_{ij,x}| \ , \tag{3.40}$$

in which $m_{ij,x}$ are the elements of $\mathbf{M}_{\mathbf{x}}$. Using (3.40) in (3.38) states that a fixed-point iteration defined by (3.35) converges to a solution $\mathbf{x}$ of the system of linear equations in (3.23) if the iteration matrix $\mathbf{M}_{\mathbf{x}}$ satisfies

$$\max_{i\in\mathbb{Z}} \sum_{j\in\mathbb{Z}} |m_{ij,x}| < 1 \ . \tag{3.41}$$

In the following, this result is discussed for the Richardson and Jacobi iteration.

3.5.3.1 Richardson Iteration

In case of the Richardson iteration, the iteration matrix $\mathbf{M_x}$ amounts to

$$\mathbf{M_x} = \mathbf{I} - \mathbf{A_x} \ , \tag{3.42}$$

and the coefficient vector $\mathbf{c_x}$ is given by

$$\mathbf{c_x} = \mathbf{y} \ , \tag{3.43}$$

which is verified by comparing (3.26) to (3.35). With (3.42), the condition for convergence in (3.41) results in

$$\max_{i \in \mathbb{Z}} \sum_{j \in \mathbb{Z}} |\delta_{ij} - a_{ij,x}| < 1 \ .$$

Instead of evaluating the maximum, this restriction may as well be imposed on *all* rows i and, using (3.24), this leads to the condition[2]

$$\sum_{k_1 \in \mathbb{Z}} |\delta[k_1] - g_{x,n}[k_1]| < 1 \ , \tag{3.44}$$

which must be fulfilled at all time instants $n \in \mathbb{Z}$ and where $\delta[k_1]$ is the *unit impulse sequence* [11], i.e.,

$$\delta[k_1] = \begin{cases} 1 \ , & \text{if } k_1 = 0 \\ 0 \ , & \text{if } k_1 \neq 0 \ . \end{cases}$$

The definition of $g_{x,n}[k_1]$ in (3.22) and the *triangle inequality* can be used to find an upper bound for the left hand side in (3.44) that depends only on the Volterra kernels and the maximum amplitude $\|\mathbf{x}\|_\infty$ of the input signal $x[n]$. Therefore, consider the summand for $k_1 = 0$, where the application of the triangle inequality and the upper bound of the input yields

$$|1 - g_{x,n}[0]| \leq |1 - h_{1,n}[0]| + \sum_{p=2}^{\infty} \sum_{k_2,\ldots,k_p \in \mathbb{Z}} |h_{p,n}[0, k_2, \ldots, k_p]| \cdot \|\mathbf{x}\|_\infty^{p-1} \ .$$

Note that equality is only obtained if $h_{1,n}[0] = 1$. Using this upper bound in (3.44), applying the triangle inequality and the upper bound on the input on all summands for $k_1 \neq 0$, and adding $|h_{1,n}[0]|$ to both sides of the inequality results in

$$\sum_{p=1}^{\infty} \sum_{k_1,\ldots,k_p \in \mathbb{Z}} |h_{p,n}[k_1, \ldots, k_p]| \cdot \|\mathbf{x}\|_\infty^{p-1} < 1 + |h_{1,n}[0]| - |1 - h_{1,n}[0]| \ .$$

It can be observed that for $h_{1,n}[0] \leq 0$, the right hand side is zero and the inequality cannot be fulfilled. If $h_{1,n}[0]$ is between zero and one, the right hand side is equal to $2h_{1,n}[0]$, and if $h_{1,n}[0]$ is greater than one, the right hand side is always 2. In summary, the condition for convergence for the Ricardson iteration in terms of the Volterra kernels $h_{p,n}$ and the maximum

[2] Note that this corresponds to the "'weakly time-varying"'-property defined in [4].

input amplitude $\|\mathbf{x}\|_\infty$ is given by

$$\sum_{p=1}^{\infty} \sum_{k_1,\ldots,k_p \in \mathbb{Z}} |h_{p,n}[k_1,\ldots,k_p]| \cdot \|\mathbf{x}\|_\infty^{p-1} < 2\min(h_{1,n}[0],1) \;, \tag{3.45}$$

in which $\min(\cdot,\cdot)$ denotes the minimum value of its two arguments. Note that the inequality cannot be fulfilled for $h_{1,n}[0] \leq 0$ and $h_{1,n}[0] \geq 2$, as the left hand side is lower bounded by zero and $|h_{1,n}[0]|$ is a summand on the left hand side as well.

3.5.3.2 Jacobi Iteration

In case of the Jacobi iteration, the iteration matrix $\mathbf{M}_\mathbf{x}$ amounts to

$$\mathbf{M}_\mathbf{x} = \mathbf{U}_\mathbf{x}^{-1}\mathbf{V}_\mathbf{x} \;, \tag{3.46}$$

and the coefficient vector $\mathbf{c}_\mathbf{x}$ is given by

$$\mathbf{c}_\mathbf{x} = \mathbf{U}_\mathbf{x}^{-1}\mathbf{y} \;, \tag{3.47}$$

which is verified by comparing (3.32) to (3.35). With (3.46), (3.29) and (3.30), the condition for convergence in (3.41) evolves as [51]

$$\max_{i\in\mathbb{Z}} \sum_{j\in\mathbb{Z}\setminus i} \left|\frac{a_{ij,x}}{a_{ii,x}}\right| < 1 \;.$$

With (3.24), this condition may be restated in terms of the time-varying impulse response $g_{x,n}[k_1]$, i.e.,

$$\sum_{k_1\in\mathbb{Z}\setminus 0} |g_{x,n}[k_1]| < |g_{x,n}[0]| \;, \tag{3.48}$$

which must be fulfilled at all time instants $n \in \mathbb{Z}$. It should be pointed out that (3.48) requires that $g_{x,n}[0] \neq 0$ and that it is less restrictive compared to (3.44), because it represents a relative constraint, whereas (3.44) imposes a restriction on the absolute values. In order to find a condition that depends only on the Volterra kernels and the maximum amplitude $\|\mathbf{x}\|_\infty$ of the input, the coefficient at time lag zero of the first-order Volterra kernel is assumed to be greater than or equal to zero, i.e., $h_{1,n}[0] \geq 0$, which implies

$$|g_{x,n}[0]| \geq h_{1,n}[0] - \sum_{p=2}^{\infty} \sum_{k_2,\ldots,k_p\in\mathbb{Z}} |h_{p,n}[0,k_2,\ldots,k_p]| \cdot \|\mathbf{x}\|_\infty^{p-1} \;.$$

Using this result in (3.48), applying the triangle inequality and the upper bound on the input to the left hand side, and adding $h_{1,n}[0]$ to both sides of the inequality yields

$$\sum_{p=1}^{\infty} \sum_{k_1,\ldots,k_p\in\mathbb{Z}} |h_{p,n}[k_1,\ldots,k_p]| \cdot \|\mathbf{x}\|_\infty^{p-1} < 2h_{1,n}[0] \;. \tag{3.49}$$

Algorithm 3 Concept for a realizable equalizer

1. Use the initial approximation $\tilde{\mathbf{x}}^{(0)} = \mathbf{y}$
2. Iterate the following steps for $r = 0, 1, 2, \ldots$ until convergence:
   a) Calculate an approximation $\mathbf{M}_{\tilde{\mathbf{x}}^{(r)}}$ for the iteration matrix $\mathbf{M}_{\mathbf{x}}$ using $\tilde{\mathbf{x}}^{(r)}$
   b) Calculate an approximation $\mathbf{c}_{\tilde{\mathbf{x}}^{(r)}}$ for the coefficient vector $\mathbf{c}_{\mathbf{x}}$ using $\tilde{\mathbf{x}}^{(r)}$
   c) Calculate a new approximation $\tilde{\mathbf{x}}^{(r+1)}$ of the solution $\mathbf{x}$ by means of

$$\tilde{\mathbf{x}}^{(r+1)} = \mathbf{M}_{\tilde{\mathbf{x}}^{(r)}} \tilde{\mathbf{x}}^{(r)} + \mathbf{c}_{\tilde{\mathbf{x}}^{(r)}} \tag{3.50}$$

A comparison of (3.49) to (3.45) reveals that the condition for convergence for the Jacobi iteration is indeed less restrictive compared to the one for the Richardson iteration, as it only requires $h_{1,n}[0] > 0$ instead of $0 < h_{1,n}[0] < 2$.

3.6 Equalizers based on Iterative Methods

In Section 3.5, two algorithms were introduced that allow the reconstruction of the input $\mathbf{x}$ from $\mathbf{y}$ and $\mathbf{A}_{\mathbf{x}}$ as long as certain conditions are fulfilled. However, a realizable equalizer cannot generate the coefficient matrix $\mathbf{A}_{\mathbf{x}}$ defined by (3.24), because the time-varying impulse responses $g_{x,n}[k_1]$ in (3.22) require the knowledge of the input $\mathbf{x}$. Indeed, if the input is known, then equalization is trivial anyway. An ad-hoc solution to attain realizability would be to use $\mathbf{y}$ as an approximation for $\mathbf{x}$, construct a coefficient matrix $\mathbf{A}_{\mathbf{y}}$, and use it as a substitute for $\mathbf{A}_{\mathbf{x}}$ in the fixed-point iteration. As a consequence, a *different* system of linear equations would be solved, whose solution is in general different from $\mathbf{x}$. Consequently, such an attempt is not meaningful. However, if $\mathbf{A}_{\mathbf{y}}$ is used for *one* iteration and the obtained approximation is closer to $\mathbf{x}$ compared to $\mathbf{y}$, then it may be used to derive a better approximation for $\mathbf{A}_{\mathbf{x}}$. This thought leads to the equalization concept shown in Algorithm 3, which represents a realizable equalizer. Indeed, it is possible to find conditions for which convergence to the solution $\mathbf{x}$ is guaranteed, but the associated proofs are quite involved. Hence, before these conditions are investigated, the algorithm above shall be adapted for the Richardson and Jacobi iteration.

3.6.1 Richardson Equalizer

In accordance with (3.42) and (3.43), the iteration matrix $\mathbf{M}_{\tilde{\mathbf{x}}^{(r)}}$ and the coefficient vector $\mathbf{c}_{\tilde{\mathbf{x}}^{(r)}}$ based on the approximation $\tilde{\mathbf{x}}^{(r)}$ are defined as

$$\mathbf{M}_{\tilde{\mathbf{x}}^{(r)}} = \mathbf{I} - \mathbf{A}_{\tilde{\mathbf{x}}^{(r)}} \tag{3.51}$$

and

$$\mathbf{c}_{\tilde{\mathbf{x}}^{(r)}} = \mathbf{y} \, , \tag{3.52}$$

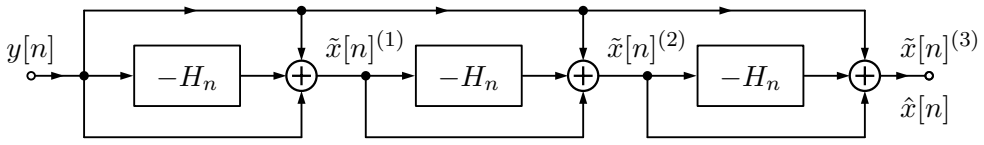


Figure 3.7: Richardson equalizer with three iterations using the initial approximation $\tilde{x}[n]^{(0)} = y[n]$.

Algorithm 4 Richardson equalizer

1. Use the initial approximation $\tilde{x}[n]^{(0)} = y[n]$
2. Choose the number R of iterations according to the required equalization performance
3. Iterate the following step for $r = 0, 1, \ldots, R-1$:
   a) Calculate a new approximation $\tilde{x}[n]^{(r+1)}$ of the solution $x[n]$ using (3.55)
4. Use $\tilde{x}[n]^{(R)}$ as the reconstruction $\hat{x}[n]$ of the input sample $x[n]$

respectively. The coefficient matrix $\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}}$ is specified in terms of its elements as

$$a_{ij,\tilde{x}^{(r)}} = g_{\tilde{x}^{(r)},i}[i-j] \;, \tag{3.53}$$

where $i, j \in \mathbb{Z}$ and the time-varying impulse response $g_{\tilde{x}^{(r)},n}[k_1]$ based on the approximation $\tilde{\mathbf{x}}^{(r)}$ is given by

$$g_{\tilde{x}^{(r)},n}[k_1] = h_{1,n}[k_1] + \sum_{p=2}^{\infty} \sum_{k_2,\ldots,k_p \in \mathbb{Z}} h_{p,n}[k_1, \ldots, k_p] \prod_{i=2}^{p} \tilde{x}[n-k_i]^{(r)} \;. \tag{3.54}$$

In order to obtain a representation of (3.50) in terms of the impulse response $g_{\tilde{x}^{(r)},n}[k_1]$, it is observed that the only difference to (3.26) is the substitution of $\mathbf{A}_{\mathbf{x}}$ by $\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}}$, hence the fixed-point iteration is obtained in analogy to (3.27) as

$$\tilde{x}[n]^{(r+1)} = \tilde{x}[n]^{(r)} + y[n] - \sum_{k_1 \in \mathbb{Z}} g_{\tilde{x}^{(r)},n}[k_1] \tilde{x}[n-k_1]^{(r)} \;.$$

Finally, using (3.54), this equation can be rewritten as

$$\tilde{x}[n]^{(r+1)} = \tilde{x}[n]^{(r)} + y[n] - H_n\{\tilde{x}[n]^{(r)}\} \;, \tag{3.55}$$

which is called *Richardson equalizer*. The corresponding equalization procedure can be described as in Algorithm 4 and a Richardson equalizer based on three iterations is depicted in Figure 3.7. It shall be pointed out that the Richardson equalizer coincides with the nonlinear Richardson iteration in (3.19). Furthermore, an interesting relationship to the Nowak-Van-Veen equalizer can be observed. Therefore, consider that an exact inverse $G_{1,n}$ for the first-order Volterra operator $H_{1,n}$ as defined in (3.14) exists and a Volterra system $\breve{H}_n$ is defined as

$$\breve{H}_n\{x[n]\} = G_{1,n}\{H_n\{x[n]\}\} \;. \tag{3.56}$$

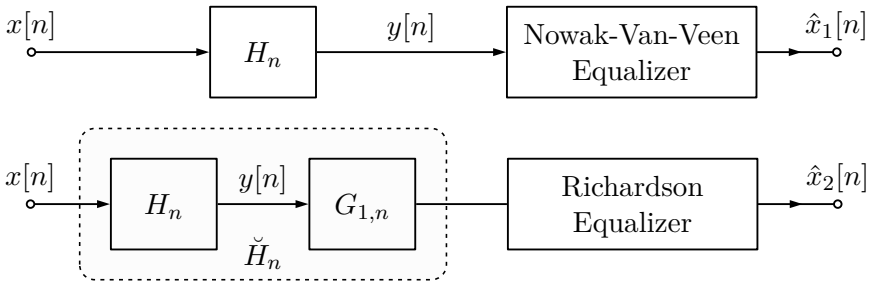


Figure 3.8: The application of the Richardson equalizer to the Volterra system $\breve{H}_n\{x[n]\}$ defined in (3.56) produces the same equalization result as the application of the Nowak-Van-Veen equalizer to the Volterra system H_n, i.e., $\hat{x}_1[n] \equiv \hat{x}_2[n]$.

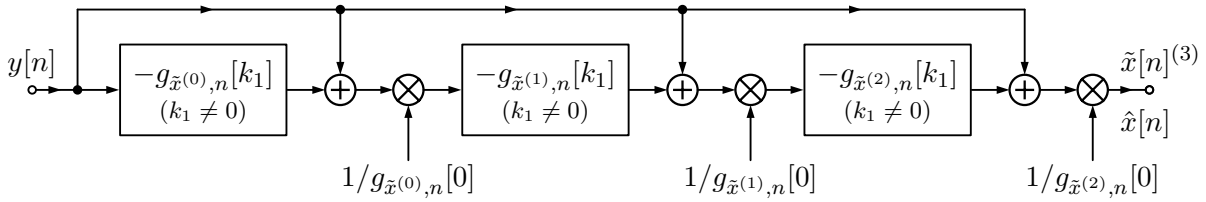


Figure 3.9: Jacobi equalizer with three iterations using the initial approximation $\tilde{x}[n]^{(0)} = y[n]$.

Algorithm 5 Jacobi equalizer

1. Use the initial approximation $\tilde{x}[n]^{(0)} = y[n]$
2. Choose the number R of iterations according to the required equalization performance
3. Iterate the following steps for $r = 0, 1, \ldots, R-1$:
   a) Calculate the impulse response $g_{\tilde{x}^{(r)},n}[k_1]$ in (3.54) using $\tilde{x}[n]^{(r)}$
   b) Calculate a new approximation $\tilde{x}[n]^{(r+1)}$ of the solution $x[n]$ using (3.59)
4. Use $\tilde{x}[n]^{(R)}$ as the reconstruction $\hat{x}[n]$ of the input sample $x[n]$

If this Volterra system is equalized using the Richardson equalizer, i.e.,

$$\begin{aligned}\tilde{x}[n]^{(r+1)} &= \tilde{x}[n]^{(r)} + G_{1,n}\{y[n]\} - G_{1,n}\{H_n\{\tilde{x}[n]^{(r)}\}\} \\ &= G_{1,n}\Big\{y[n] - \sum_{p=2}^{\infty} H_{p,n}\{\tilde{x}[n]^{(r)}\}\Big\}\,,\end{aligned}$$

and its output compared to (3.15), it turns out that the equalization result is *identical* to the Nowak-Van-Veen equalizer applied to the output of the Volterra system H_n, cf. Figure 3.8. Consequently, the Richardson equalizer may be regarded as a *generalization* of the Nowak-Van-Veen equalizer, because the latter amounts to the application of the Richardson equalizer to a modified Volterra system $\breve{H}_n$.

3.6.2 Jacobi Equalizer

In accordance with (3.46) and (3.47), the iteration matrix $\mathbf{M}_{\tilde{\mathbf{x}}^{(r)}}$ and the coefficient vector $\mathbf{c}_{\tilde{\mathbf{x}}^{(r)}}$ based on the approximation $\tilde{\mathbf{x}}^{(r)}$ are defined as

$$\mathbf{M}_{\tilde{\mathbf{x}}^{(r)}} = \mathbf{U}_{\tilde{\mathbf{x}}^{(r)}}^{-1} \mathbf{V}_{\tilde{\mathbf{x}}^{(r)}} \tag{3.57}$$

and

$$\mathbf{c}_{\tilde{\mathbf{x}}^{(r)}} = \mathbf{U}_{\tilde{\mathbf{x}}^{(r)}}^{-1} \mathbf{y} \, , \tag{3.58}$$

where $\mathbf{U}_{\tilde{\mathbf{x}}^{(r)}}$ and $\mathbf{V}_{\tilde{\mathbf{x}}^{(r)}}$ are defined in terms of $\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}}$ in the same fashion as $\mathbf{U}_{\mathbf{x}}$ and $\mathbf{V}_{\mathbf{x}}$ are defined for $\mathbf{A}_{\mathbf{x}}$ in (3.29) and (3.30), respectively. The fixed-point iteration in terms of the impulse response $g_{\tilde{x},n}[k_1]$ is found in analogy to (3.33) as

$$\tilde{x}[n]^{(r+1)} = \frac{1}{g_{\tilde{x}^{(r)},n}[0]} \Big(y[i] - \sum_{k_1 \in \mathbb{Z} \setminus 0} g_{\tilde{x}^{(r)},n}[k_1] \tilde{x}[n-k_1]^{(r)} \Big) \, , \tag{3.59}$$

which is called *Jacobi equalizer*. The corresponding equalization procedure can be described as in Algorithm 5 and a Jacobi equalizer based on three iterations is depicted in Figure 3.9.

3.6.3 Condition for Convergence

In the following, the condition for convergence is investigated. First, the general equalizer in (3.50) shall be considered and, subsequently, the result is discussed in the light of the Richardson equalizer, whereas the Jacobi equalizer will not be treated here. The analysis of convergence is inspired by the approach pursued in Section 3.5.3, i.e., based on the solution $\mathbf{x}$ of (3.34), the error of the approximation $\tilde{\mathbf{x}}^{(r)}$ in iteration r, i.e.,

$$\tilde{\mathbf{e}}^{(r)} = \mathbf{x} - \tilde{\mathbf{x}}^{(r)} \, , \tag{3.60}$$

is investigated. Subtracting (3.50) from (3.34) results in

$$\tilde{\mathbf{e}}^{(r+1)} = \mathbf{M}_{\mathbf{x}} \tilde{\mathbf{e}}^{(r)} + (\mathbf{M}_{\mathbf{x}} - \mathbf{M}_{\tilde{\mathbf{x}}^{(r)}}) \tilde{\mathbf{x}}^{(r)} + \mathbf{c}_{\mathbf{x}} - \mathbf{c}_{\tilde{\mathbf{x}}^{(r)}} \, .$$

If the error of the approximation $\mathbf{M}_{\tilde{\mathbf{x}}^{(r)}}$ with respect to $\mathbf{M}_{\mathbf{x}}$ is defined as

$$\overline{\mathbf{M}}_{\tilde{\mathbf{x}}^{(r)}} = \mathbf{M}_{\mathbf{x}} - \mathbf{M}_{\tilde{\mathbf{x}}^{(r)}} \tag{3.61}$$

and the error of the approximation $\mathbf{c}_{\tilde{\mathbf{x}}^{(r)}}$ with respect to $\mathbf{c}_{\mathbf{x}}$ is specified by

$$\overline{\mathbf{c}}_{\tilde{\mathbf{x}}^{(r)}} = \mathbf{c}_{\mathbf{x}} - \mathbf{c}_{\tilde{\mathbf{x}}^{(r)}} \, , \tag{3.62}$$

the error $\tilde{\mathbf{e}}^{(r+1)}$ renders

$$\tilde{\mathbf{e}}^{(r+1)} = \mathbf{M}_{\mathbf{x}} \tilde{\mathbf{e}}^{(r)} + \overline{\mathbf{M}}_{\tilde{\mathbf{x}}^{(r)}} \tilde{\mathbf{x}}^{(r)} + \overline{\mathbf{c}}_{\tilde{\mathbf{x}}^{(r)}} \, . \tag{3.63}$$

It is worthwhile to compare this result to the error in (3.36) of the original iteration, where it can be observed that the recursive relation has the same fundamental structure, but two additional terms emerged due to the deviation of $\mathbf{M}_{\tilde{\mathbf{x}}^{(r)}}$ and $\mathbf{c}_{\tilde{\mathbf{x}}^{(r)}}$ from $\mathbf{M}_{\mathbf{x}}$ and $\mathbf{c}_{\mathbf{x}}$, respectively. For $\tilde{\mathbf{x}}^{(r)}$ to converge to $\mathbf{x}$, the error $\tilde{\mathbf{e}}^{(r)}$ eventually has to decay to zero, i.e.,

$$\lim_{r\to\infty} \tilde{\mathbf{e}}^{(r)} = \mathbf{0} \,,$$

or, equivalently, its vector norm has to be strictly monotonically decreasing, i.e.,

$$\|\tilde{\mathbf{e}}^{(r+1)}\| < \|\tilde{\mathbf{e}}^{(r)}\| \,, \tag{3.64}$$

for all $r \geq 0$. Using (3.63) and the triangle inequality for vector norms [51] in (3.64) leads to the condition

$$\|\mathbf{M}_{\mathbf{x}}\tilde{\mathbf{e}}^{(r)}\| + \|\overline{\mathbf{M}}_{\tilde{\mathbf{x}}^{(r)}}\tilde{\mathbf{x}}^{(r)}\| + \|\overline{\mathbf{c}}_{\tilde{\mathbf{x}}^{(r)}}\| < \|\tilde{\mathbf{e}}^{(r)}\| \,.$$

For *induced* matrix norms, the inequality

$$\|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\| \tag{3.65}$$

holds [51], which leads to the condition

$$\|\mathbf{M}_{\mathbf{x}}\| \cdot \|\tilde{\mathbf{e}}^{(r)}\| + \|\overline{\mathbf{M}}_{\tilde{\mathbf{x}}^{(r)}}\| \cdot \|\tilde{\mathbf{x}}^{(r)}\| + \|\overline{\mathbf{c}}_{\tilde{\mathbf{x}}^{(r)}}\| < \|\tilde{\mathbf{e}}^{(r)}\| \,. \tag{3.66}$$

It is interesting to compare this result to the condition for convergence in (3.38) for the original iteration, where it can be observed that the constraint on the contractiveness of $\mathbf{M}_{\mathbf{x}}$ is tightened so that it can compensate for any influence due to $\overline{\mathbf{M}}_{\tilde{\mathbf{x}}^{(r)}}$ and $\overline{\mathbf{c}}_{\tilde{\mathbf{x}}^{(r)}}$. The condition in (3.66) is on a rather high level of abstraction and does not represent an immediately applicable criterion. In order to arrive at a practically useful formulation, (3.66) is investigated below in more detail for the Richardson equalizer.

3.6.3.1 Richardson Equalizer

In the following, the condition for convergence in (3.66) shall be utilized to find a constraint in terms of the kernels $h_{p,n}$ of the Volterra system and the maximum input amplitude $\|\mathbf{x}\|_\infty$, i.e., a condition that can be immediately verified. Therefore, $\overline{\mathbf{M}}_{\tilde{\mathbf{x}}^{(r)}}$ and $\overline{\mathbf{c}}_{\tilde{\mathbf{x}}^{(r)}}$ are investigated for the Richardson equalizer, i.e., using (3.42) and (3.51) in (3.61) provides

$$\overline{\mathbf{M}}_{\tilde{\mathbf{x}}^{(r)}} = \mathbf{A}_{\tilde{\mathbf{x}}^{(r)}} - \mathbf{A}_{\mathbf{x}}$$

and using (3.43) and (3.52) in (3.62) results in

$$\overline{\mathbf{c}}_{\tilde{\mathbf{x}}^{(r)}} = \mathbf{0} \,.$$

In conjunction with (3.42), the condition for convergence in (3.66) renders

$$\|\mathbf{I} - \mathbf{A}_{\mathbf{x}}\| \cdot \|\tilde{\mathbf{e}}^{(r)}\| + \|\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}} - \mathbf{A}_{\mathbf{x}}\| \cdot \|\tilde{\mathbf{x}}^{(r)}\| < \|\tilde{\mathbf{e}}^{(r)}\| \,. \tag{3.67}$$

It is not possible to utilize bounds on $\|\tilde{\mathbf{e}}^{(r)}\|$ to simplify (3.67), as its lower bound is zero and, therewith, the condition renders meaningless. Thus, the goal pursued in the following is to extract the factor $\|\tilde{\mathbf{e}}^{(r)}\|$ from $\|\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}} - \mathbf{A}_{\mathbf{x}}\|$ so that it cancels in (3.67). The analysis is based on the *maximum norm* because of its beneficial properties, which are discussed in Section 3.5.3. The matrix norm induced by the maximum norm defined in (3.40) yields

$$\|\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}} - \mathbf{A}_{\mathbf{x}}\|_\infty = \max_{i\in\mathbb{Z}} \sum_{j\in\mathbb{Z}} |a_{ij,\tilde{x}^{(r)}} - a_{ij,x}|$$

and, with (3.24) and (3.53), this may be expressed in terms of the impulse responses as

$$\|\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}} - \mathbf{A}_{\mathbf{x}}\|_\infty = \max_{n\in\mathbb{Z}} \sum_{k_1\in\mathbb{Z}} |g_{\tilde{x}^{(r)},n}[k_1] - g_{x,n}[k_1]| \,. \tag{3.68}$$

In order to find an upper bound for $\|\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}} - \mathbf{A}_{\mathbf{x}}\|_\infty$, the expression $|g_{\tilde{x}^{(r)},n}[k_1] - g_{x,n}[k_1]|$ is investigated by utilizing (3.22), (3.54), and the triangle inequality, which results in

$$|g_{\tilde{x}^{(r)},n}[k_1] - g_{x,n}[k_1]| \leq \sum_{p=2}^{\infty} \sum_{k_2,\ldots,k_p\in\mathbb{Z}} |h_{p,n}[k_1,\ldots,k_p]| \cdot \gamma_{\tilde{x}^{(r)},x,n}[p,k_2,\ldots,k_p] \,, \tag{3.69}$$

where

$$\gamma_{\tilde{x}^{(r)},x,n}[p,k_2,\ldots,k_p] = \left| \prod_{i=2}^{p} \tilde{x}[n-k_i]^{(r)} - \prod_{i=2}^{p} x[n-k_i] \right| .$$

Using (3.60), $\gamma_{\tilde{x}^{(r)},x,n}$ may be rewritten as

$$\gamma_{\tilde{x}^{(r)},x,n}[p,k_2,\ldots,k_p] = \left| \prod_{i=2}^{p} \left(x[n-k_i] - \tilde{e}[n-k_i]^{(r)} \right) - \prod_{i=2}^{p} x[n-k_i] \right| . \tag{3.70}$$

It can be observed that the first product in (3.70), if expanded, contains a factor that cancels with the second product. In order to find an upper bound for the remaining terms, it is highlighted that (3.39) enables the upper bound

$$\left| \prod_{i=2}^{p} \left(x[n-k_i] - \tilde{e}[n-k_i]^{(r)} \right) \right| \leq \left(\|\mathbf{x}\|_\infty + \|\tilde{\mathbf{e}}^{(r)}\|_\infty \right)^{p-1} ,$$

and the *binomial theorem* gives

$$\left(\|\mathbf{x}\|_\infty + \|\tilde{\mathbf{e}}^{(r)}\|_\infty \right)^{p-1} = \|\mathbf{x}\|_\infty^{p-1} + \sum_{\nu=1}^{p-2} \binom{p-1}{\nu} \|\mathbf{x}\|_\infty^{p-1-\nu} \cdot \|\tilde{\mathbf{e}}^{(r)}\|_\infty^{\nu} + \|\tilde{\mathbf{e}}^{(r)}\|_\infty^{p-1} .$$

The term $\|\mathbf{x}\|_\infty^{p-1}$ corresponds to the upper bound of the term that cancels with the second product in (3.70). Therefore, if the upper bound is utilized only for the remaining terms, $\gamma_{\tilde{x}^{(r)},x,n}$ is upper bounded by

$$\gamma_{\tilde{x}^{(r)},x,n}[p,k_2,\ldots,k_p] \leq \|\tilde{\mathbf{e}}^{(r)}\|_\infty \left[\|\tilde{\mathbf{e}}^{(r)}\|_\infty^{p-2} + \sum_{\nu=1}^{p-2} \binom{p-1}{\nu} \|\mathbf{x}\|_\infty^{p-1-\nu} \cdot \|\tilde{\mathbf{e}}^{(r)}\|_\infty^{\nu-1} \right] . \tag{3.71}$$

If the iteration converges, it fulfills (3.64) and, with (3.60), the error can be upper bounded by

$$\|\tilde{\mathbf{e}}^{(r)}\|_\infty \leq \|\tilde{\mathbf{e}}^{(0)}\|_\infty = \|\mathbf{x} - \tilde{\mathbf{x}}^{(0)}\|_\infty \,, \tag{3.72}$$

for $r \geq 0$. It may seem dubious to employ an inequality in a derivation, whose resulting condition should ensure that the utilized inequality actually holds. However, the resulting condition for convergence indeed ensures that (3.64) is fulfilled, which is shown below by induction. Assuming that the iteration is initialized with $\tilde{\mathbf{x}}^{(0)} = \mathbf{y}$, using (3.23) and (3.42) in (3.72) leads to the upper bound

$$\|\tilde{\mathbf{e}}^{(r)}\|_\infty \leq \|\mathbf{x} - \mathbf{y}\|_\infty = \|\mathbf{x} - \mathbf{A}_\mathbf{x}\mathbf{x}\|_\infty = \|(\mathbf{I} - \mathbf{A}_\mathbf{x})\mathbf{x}\|_\infty = \|\mathbf{M}_\mathbf{x}\mathbf{x}\|_\infty \,.$$

The right hand side may again be upper bounded using the property (3.65) of induced matrix norms, which gives

$$\|\tilde{\mathbf{e}}^{(r)}\|_\infty \leq \|\mathbf{M}_\mathbf{x}\|_\infty \cdot \|\mathbf{x}\|_\infty \,.$$

If the iteration converges, it fulfills (3.38), as it represents a weaker constraint compared to (3.66), and, therewith, the upper bound

$$\|\tilde{\mathbf{e}}^{(r)}\|_\infty < \|\mathbf{x}\|_\infty \tag{3.73}$$

is obtained, which holds for all $r \geq 0$ if $\tilde{\mathbf{x}}^{(0)} = \mathbf{y}$. Using this result in (3.71) yields

$$\gamma_{\tilde{x}^{(r)},x,n}[p, k_2, \ldots, k_p] < \|\tilde{\mathbf{e}}^{(r)}\|_\infty \cdot \|\mathbf{x}\|_\infty^{p-2} \left[1 + \sum_{\nu=1}^{p-2} \binom{p-1}{\nu}\right] .$$

Recalling that the term in square brackets emerged from the expansion of an expression $(a+b)^{p-1}$ and the subtraction of one term, it is evident that the expression in square brackets corresponds to the number of terms after expansion reduced by one. The recursive expansion

$$(a+b)^{p-1} = a(a+b)^{p-2} + b(a+b)^{p-2}$$

demonstrates that the full expansion results in 2^{p-1} terms. Consequently, the term in square brackets is $2^{p-1} - 1$ and

$$\gamma_{\tilde{x}^{(r)},x,n}[p, k_2, \ldots, k_p] < \|\tilde{\mathbf{e}}^{(r)}\|_\infty \cdot \|\mathbf{x}\|_\infty^{p-2}(2^{p-1} - 1) \,. \tag{3.74}$$

Finally, using (3.69) and (3.74) in (3.68) results in the upper bound

$$\|\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}} - \mathbf{A}_\mathbf{x}\|_\infty < \|\tilde{\mathbf{e}}^{(r)}\|_\infty \cdot \max_{n\in\mathbb{Z}} \sum_{p=2}^{\infty} \sum_{k_1,\ldots,k_p\in\mathbb{Z}} |h_{p,n}[k_1, \ldots, k_p]| \cdot \|\mathbf{x}\|_\infty^{p-2}(2^{p-1} - 1) \,, \tag{3.75}$$

where, again, it is assumed that $\tilde{\mathbf{x}}^{(0)} = \mathbf{y}$. Before utilizing this result in (3.67), an upper bound for $\|\tilde{\mathbf{x}}^{(r)}\|$ is derived using (3.60), the triangle inequality and (3.73), i.e.,

$$\|\tilde{\mathbf{x}}^{(r)}\|_\infty = \|\mathbf{x} - \tilde{\mathbf{e}}^{(r)}\|_\infty \leq \|\mathbf{x}\|_\infty + \|\tilde{\mathbf{e}}^{(r)}\|_\infty < 2\|\mathbf{x}\|_\infty \,. \tag{3.76}$$

The result in (3.45) provides an upper bound for $\|\mathbf{I}-\mathbf{A}_{\mathbf{x}}\|_\infty$, i.e.,

$$\|\mathbf{I}-\mathbf{A}_{\mathbf{x}}\|_\infty \leq 1-2\min(h_{1,n}[0],1)+\sum_{p=1}^{\infty}\sum_{k_1,\ldots,k_p\in\mathbb{Z}}|h_{p,n}[k_1,\ldots,k_p]|\cdot\|\mathbf{x}\|_\infty^{p-1}\,.$$

Using this upper bound with (3.75) and (3.76) in (3.67), a *sufficient* condition for convergence for the Richardson equalizer is given by

$$\sum_{p=1}^{\infty}\sum_{k_1,\ldots,k_p\in\mathbb{Z}}|h_{p,n}[k_1,\ldots,k_p]|\cdot\|\mathbf{x}\|_\infty^{p-1}(2^p-1)<2\min(h_{1,n}[0],1)\,, \tag{3.77}$$

which has to hold for all $n\in\mathbb{Z}$ and requires the initialization $\tilde{\mathbf{x}}^{(0)}=\mathbf{y}$. A comparison of this result to the condition for convergence in (3.45) for the original iteration reveals that the kernels are weighted by the additional factor 2^p-1, hence the constraints on the Volterra kernels of order 2 and higher are tightened in order to guarantee convergence. Due to the use of (3.73) and (3.76) in the derivation, the upper bound is not tight, meaning that it is not a necessary condition to guarantee convergence in the worst case. This can be attributed to the fact that the upper bound for $\|\tilde{\mathbf{e}}^{(r)}\|_\infty$ in (3.73), which is accurate for Volterra systems that are close to the bound in (3.45), is somewhat imprecise for Volterra systems that fulfill (3.77). Furthermore, due to the many parameters involved, the worst case is highly improbable, which implies that the condition for convergence in (3.77) is quite conservative with respect to practical scenarios. Finally, it is noteworthy that (3.77) is particularly simple to evaluate compared to the conditions reported for the Nowak-Van-Veen equalizer [9] and the Pth-order inverse [46].

In order to establish the completeness of the derivation, it is shown that the condition in (3.77) ensures that (3.64) holds. Therefore, (3.77) is multiplied by $\|\tilde{\mathbf{e}}^{(r)}\|_\infty$, divided by $2\min(h_{1,n}[0],1)$, and considered for the worst case, i.e.,

$$\eta\|\tilde{\mathbf{e}}^{(r)}\|_\infty<\|\tilde{\mathbf{e}}^{(r)}\|_\infty\,, \tag{3.78}$$

where the factor η is given by

$$\eta=\max_{n\in\mathbb{Z}}\frac{1}{2\min(h_{1,n}[0],1)}\sum_{p=1}^{\infty}\sum_{k_1,\ldots,k_p\in\mathbb{Z}}|h_{p,n}[k_1,\ldots,k_p]|\cdot\|\mathbf{x}\|_\infty^{p-1}(2^p-1)\,.$$

For $r=0$, (3.72) is an equality and, therefore, does not depend on the validity of (3.64). Consequently, (3.78) ensures that

$$\|\tilde{\mathbf{e}}^{(1)}\|_\infty<\eta\|\tilde{\mathbf{e}}^{(0)}\|_\infty<\|\tilde{\mathbf{e}}^{(0)}\|_\infty\,, \tag{3.79}$$

where the left inequality is indeed strict, as the derivation utilizes two upper bounds that are not tight. This establishes the basis for an induction, which ensures that (3.72) holds for $r=1$ as well and implicates that $\|\tilde{\mathbf{e}}^{(2)}\|_\infty$ is less than $\eta\|\tilde{\mathbf{e}}^{(1)}\|_\infty$. Consequently, this also applies to subsequent iterations, which proves the induction step

$$\|\tilde{\mathbf{e}}^{(r+1)}\|_\infty<\eta\|\tilde{\mathbf{e}}^{(r)}\|_\infty\,. \tag{3.80}$$

Algorithm 6 Generation of a random Volterra system

Require: The maximum input amplitude is limited to one, i.e., $\|\mathbf{x}\|_\infty \leq 1$

procedure GENERATEVOLTERRASYSTEM(N, Q, K)
  for $n = 0$ to $N-1$ **do**
    Sample $h_{1,n}[0]$ from a specified uniform distribution
    $c \leftarrow 2\min(h_{1,n}[0], 1) - h_{1,n}[0]$
    for $k_1 = 1$ to K **do**
      Sample $h_{1,n}[k_1]$ from $\mathcal{U}(0, s)$, where $s = c/(QK)$
    end for
    $c \leftarrow c - \sum_{k_1=1}^{K} h_{1,n}[k_1]$
    for $p = 2$ to Q **do**
      for $k_1, \ldots, k_p = 0$ to K **do**
        Sample $h_{p,n}[k_1, \ldots, k_p]$ from $\mathcal{U}(0, s)$, where $s = c/((Q+1-p)(2^p-1)(K+1)^p)$
      end for
      $c \leftarrow c - \sum_{k_1,\ldots,k_p=0}^{K} h_{p,n}[k_1, \ldots, k_p](2^p - 1)$
    end for
  end for
end procedure

As (3.77) ensures $\eta < 1$, this proves that (3.64) holds if the condition in (3.77) is fulfilled. The result in (3.80) provides another insight if the recursive relation is unfolded, i.e., after $R > 0$ iterations the error is upper bounded by

$$\|\tilde{\mathbf{e}}^{(R)}\|_\infty < \eta^R \|\tilde{\mathbf{e}}^{(0)}\|_\infty \, . \tag{3.81}$$

Consequently, in a logarithmic measure the Richardson equalizer guarantees a linear increase of equalization performance with respect to the number of iterations, which is also observed in the simulation results in the following section.

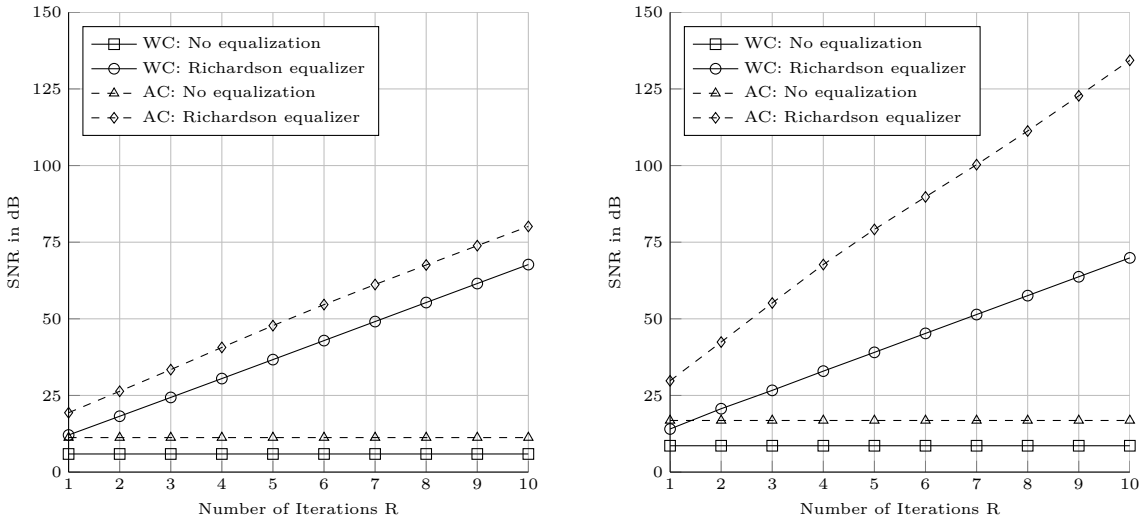
3.7 Simulation Results

The performance of the derived equalization methods, i.e., the Richardson and Jacobi equalizer, was analyzed and compared to the Pth-order inverse and the Nowak-Van-Veen equalizer by means of a simulation in MathWorks® Matlab®. The performance of the equalizers is quantified using a signal-to-noise ratio (SNR), in particular

$$\mathrm{SNR} = 10 \cdot \log\left(\frac{\sum_{n=0}^{N-1} |x[n]|^2}{\sum_{n=0}^{N-1} |x[n] - \hat{x}[n]|^2} \right) \, ,$$

in which $x[n]$ is the input signal to the Volterra system, $\hat{x}[n]$ is the output signal of the respective equalizer, and N is the number of samples. In order to guarantee convergence of the Richardson equalizer, the investigation is limited to Volterra systems that fulfill the condition in (3.77). Furthermore, only *causal* Volterra systems are considered, i.e.,

$$h_{p,n}[k_1, \ldots, k_p] = 0 \, , \; \forall k_1, \ldots, k_p < 0 \; \wedge \; \forall p \in \mathbb{N} \, .$$



(a) $Q = 1$ (b) $Q = 2$

Figure 3.10: Equalization of a Volterra system with $K = 4$ and $h_{1,n}[0] \sim \mathcal{U}(0.95, 1.05)$, where the equalization performance is shown with respect to the number of iterations R for two different maximum orders Q. In both simulations, the same Volterra system is applied to two different input signals, which are generated as $x[n] = 1$ (worst case, "WC") and $x[n] \sim \mathcal{U}(-1, 1)$ (average case, "AC"), to highlight the influence of the input signal.

The *memory depth* K, i.e.,

$$K = \inf \left\{ k \in \mathbb{N} : h_{p,n}[k_1, \ldots, k_p] = 0 \,,\ \forall k_1, \ldots, k_p > k \ \wedge\ \forall p \in \mathbb{N} \ \wedge\ \forall n \in \mathbb{Z} \right\} ,$$

of the Volterra system is set to $K = 4$ for all simulations, whereas its *maximum order* Q, i.e.,

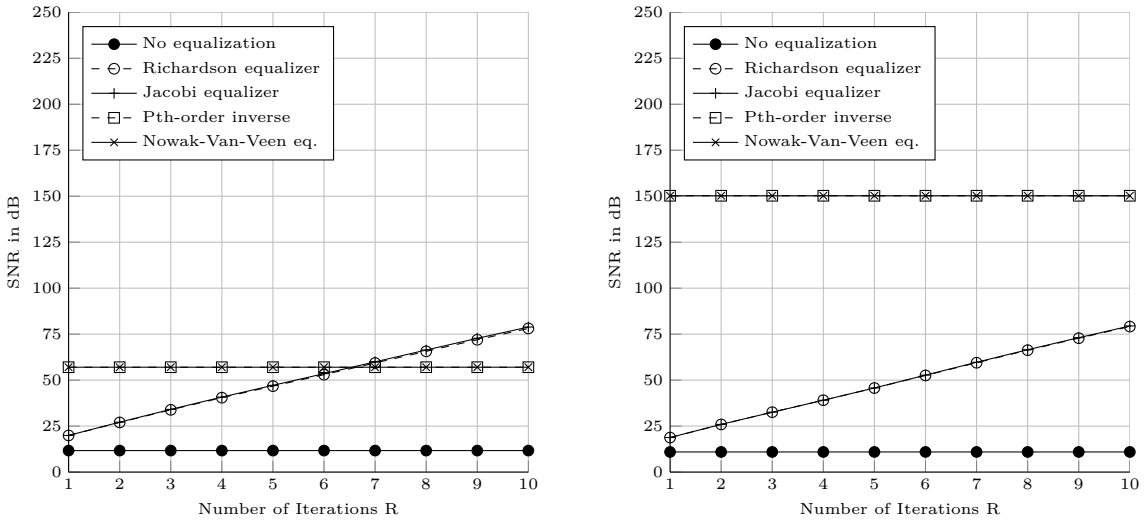
$$Q = \inf \left\{ q \in \mathbb{N} : h_{p,n}[k_1, \ldots, k_p] = 0 \,,\ \forall p > q \ \wedge\ \forall n, k_1, \ldots, k_p \in \mathbb{Z} \right\} ,$$

is varied. The equalization performance decreases as the Volterra system comes closer to the bound given by the condition for convergence in (3.77), which is implied by (3.81). In order to investigate bad cases in terms of equalization performance, the utilized Volterra systems are generated in a way that they are close to the bound in (3.77). Therefore, the coefficient $h_{1,n}[0]$ is sampled from some uniform distribution and the remaining difference to the upper bound is distributed among the other coefficients as described by Algorithm 6,[3] in which $\mathcal{U}(a, b)$ denotes a uniform distribution on the open interval (a, b). For all simulations, the input signal $x[n]$ comprises $N = 512$ samples, which are either independently sampled from $\mathcal{U}(-1, 1)$ or clamped to one. Consequently, the input amplitude is limited to one and, therefore, fulfills the requirement for Algorithm 6.

3.7.1 Influence of the Input Signal

Prior to the comparison of the equalizers, the influence of the input signal is demonstrated, which provides the background knowledge for the interpretation and discussion of the subsequent simulations. Figure 3.10 depicts the equalization performance of the Richardson equalizer for

[3] Depending on Q, this algorithm achieves a factor η in the range $0.8 < \eta < 0.9$, cf. Section 3.6.3.1.



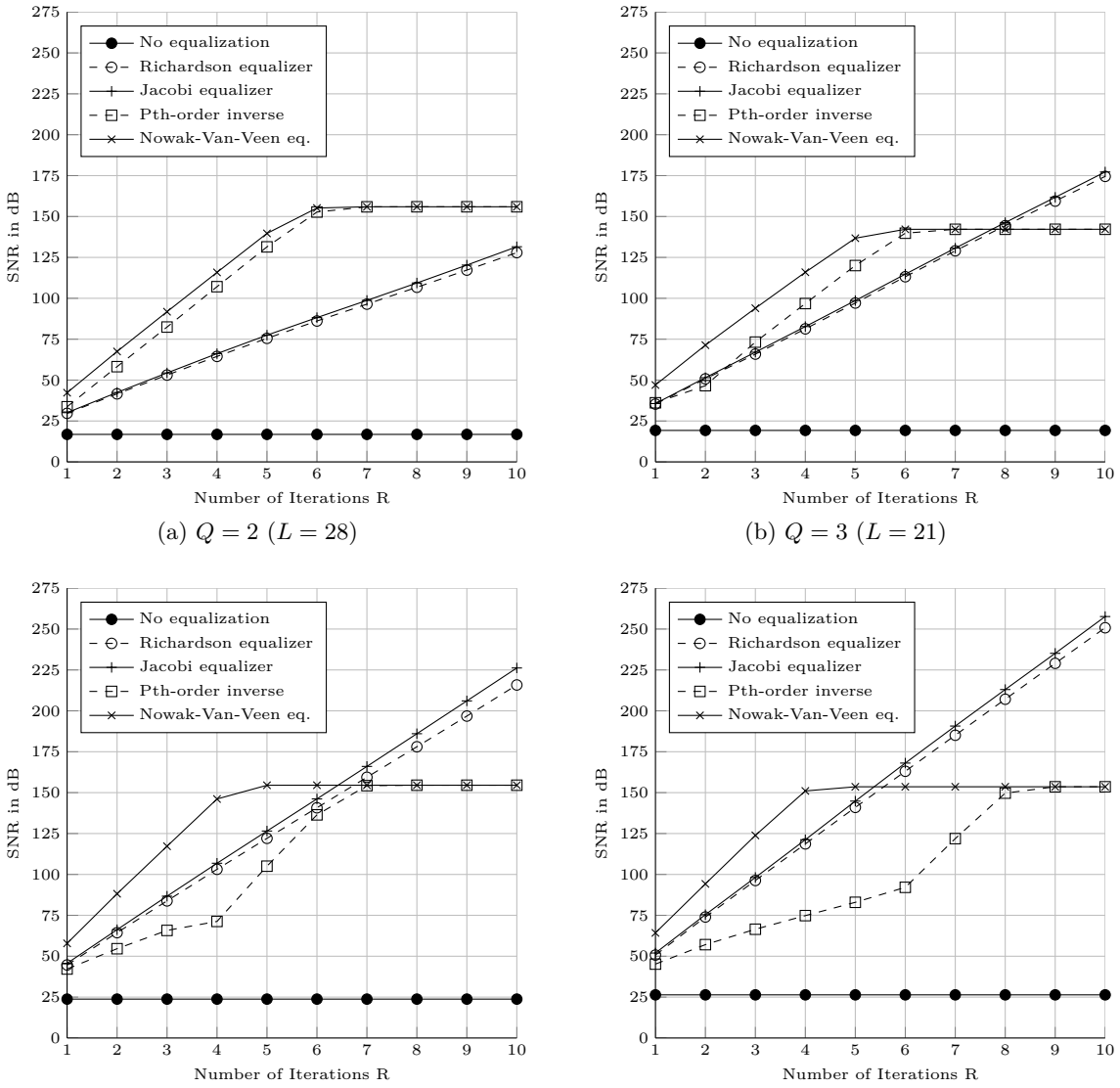
(a) $L = 13$ (b) $L = 38$

Figure 3.11: Equalization of a Volterra system with $K = 4$, $Q = 1$, and $h_{1,n}[0] \sim \mathcal{U}(0.95, 1.05)$, where the equalization performance is shown with respect to the number of iterations R for two different filter orders L of the inverse filter $G_{1,n}$. The samples of the input signal are $x[n] \sim \mathcal{U}(-1, 1)$.

different input signals, where the other equalization methods are omitted, as they exhibit a similar behavior. The input signal comprising the samples $x[n] = 1$ corresponds to the worst case with respect to equalization performance, because all coefficients of the Volterra system are positive and, therefore, the left hand side of (3.77) is attained exactly. In contrast, the input signal comprising independent and identically distributed samples $x[n] \sim \mathcal{U}(-1, 1)$ represents a more realistic stimulation, where the left hand side of (3.77) reflects the actual situation only rather imprecisely. This fact can be observed in the equalization performance, where the reconstruction of the latter input signal, called average case, is significantly better in terms of the observed SNR. Furthermore, the probability of the worst case drops considerably if the maximum order of the Volterra system is increased and, therefore, the equalization performance in the average case increases as well. This can be recognized by comparing Figure 3.10a and 3.10b, where it is evident that for $K = 4$ and $Q = 2$ the worst case is already highly improbable and the average equalization performance is much better compared to the worst case. Finally, it shall be pointed out that the equalization performance of the Richardson equalizer increases linearly with the number of iterations, which was already motivated theoretically in Section 3.6.3.1. In order to investigate the equalization performance for realistic situations, the input signal comprising the samples $x[n] \sim \mathcal{U}(-1, 1)$ is utilized in the subsequent simulations.

3.7.2 Comparison of the Equalizers

In this section, the four equalization methods are compared for different maximum orders Q of the Volterra system and different the numbers of iterations R of the equalizers, where the order of the Pth-order inverse is chosen to $P = R$ so that the resulting structure exhibits the same number of reconstruction stages, i.e., stages in which the approximation error is reduced. As discussed in Section 3.3.1 and 3.3.2, the Pth-order inverse and the Nowak-Van-Veen equalizer require an inverse filter $G_{1,n}$ for the first-order Volterra operator $H_{1,n}$. However, the exact

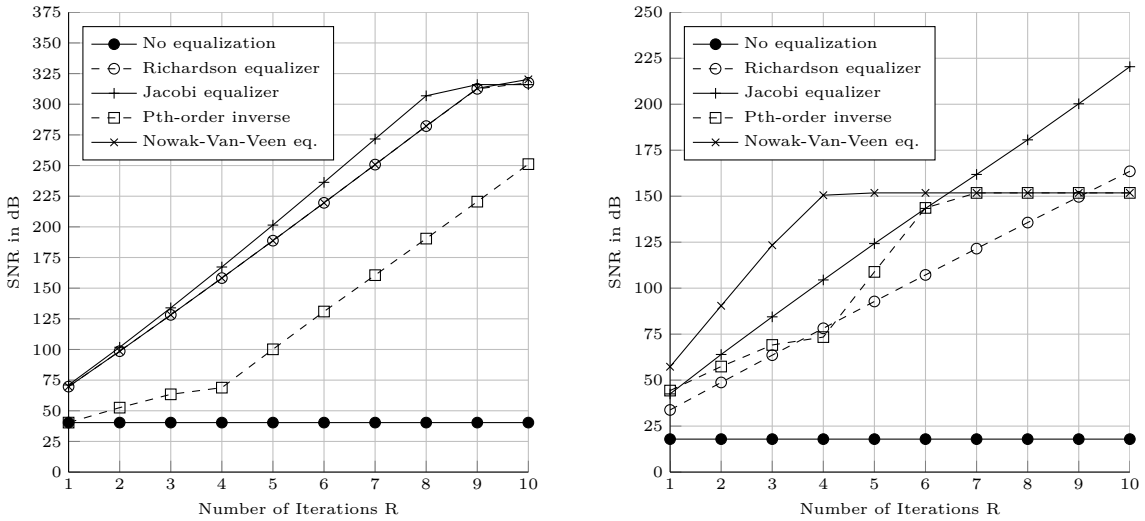


(a) $Q = 2$ ($L = 28$) (b) $Q = 3$ ($L = 21$)

(c) $Q = 5$ ($L = 19$) (d) $Q = 7$ ($L = 17$)

Figure 3.12: Equalization of a Volterra system with $K = 4$ and $h_{1,n}[0] \sim \mathcal{U}(0.95, 1.05)$, where the equalization performance is shown with respect to the number of iterations R for different maximum orders Q. The samples of the input signal are $x[n] \sim \mathcal{U}(-1, 1)$ and the filter order L of the inverse filter $G_{1,n}$ for the Pth-order inverse and Nowak-Van-Veen equalizer is chosen to provide roughly the same approximation accuracy in the different simulations.

inverse filter $H_{1,n}^{-1}$ is not known for the randomly generated first-order kernel and, therefore, the inverse filter $G_{1,n}$ needs to be designed at every time instant n. This is achieved by stating the design problem as discussed in [1] and performing an optimization in the Chebychev sense (minimax approximation) using `CVX`, a package for solving convex optimization problems in Matlab® [54, 55]. The accuracy of the inverse filter $G_{1,n}$ depends on its filter order L, which is evident in Figure 3.11, where a Volterra system comprising only a first-order kernel is equalized. For Volterra systems with $Q = 1$, the equalization performance of the Pth-order inverse and the Nowak-Van-Veen equalizer is entirely determined by the accuracy of $G_{1,n}$ and independent of the number of iterations R, as in this particular case (3.10) and (3.15) amount to $x[n]^{(r+1)} = G_{1,n}\{y[n]\}$. In contrast, the Richardson and Jacobi equalizer do not require an inverse filter for



(a) $h_{1,n}[k_1] = \delta[k_1]$ (b) $h_{1,n}[0] \sim \mathcal{U}(0.8, 1.2)$

Figure 3.13: Equalization of a Volterra system with $K = 4$ and $Q = 5$, where the equalization performance is shown with respect to the number of iterations R for different first-order Volterra kernels. The samples of the input signal are $x[n] \sim \mathcal{U}(-1, 1)$. In (a), the first-order Volterra kernel is set to the unit impulse sequence, i.e., line 3 in Algorithm 6 is replaced by $h_{1,n}[k_1] = \delta[k_1]$ and lines 5-7 are skipped, and in (b), $h_{1,n}[0]$ is sampled from $\mathcal{U}(0.8, 1.2)$.

the first-order Volterra operator, rather they implement the inverse filter using the underlying fixed-point iteration, where the equalization performance increases with every iteration.

In Figure 3.12, the performance of the equalizers is demonstrated for several different maximum orders Q of the Volterra system. The Nowak-Van-Veen equalizer outperforms the other methods for a small number of iterations, but its performance is ultimately limited by the accuracy of the inverse filter $G_{1,n}$. It can be recognized that its equalization performance increases with an increasing maximum order Q of the Volterra system. This is explained by the fact that the conditions for convergence of the underlying nonlinear fixed-point iteration improve and resembles the situation discussed in Section 3.7.1. Similarly, the performance of the Pth-order inverse is limited by the accuracy of $G_{1,n}$ as well, but it performs rather poorly if its order P is less than the maximum order Q of the Volterra system. For $P \geq Q$, its equalization performance improves considerably, as then one or more of the reconstruction stages exhibit the same structure as the ones of the Nowak-Van-Veen equalizer. The equalization performance of the Richardson and Jacobi equalizer increases exceptionally well with increasing maximum order Q. In part, this can be attributed to the fact that the worst case for convergence becomes more and more improbable and, therefore, the performance improves, cf. Section 3.7.1. However, due to the generation of the Volterra system via Algorithm 6, another important aspect of these equalization methods is revealed. In particular, the coefficients $h_{1,n}[k_1]$ of the first-order Volterra kernel, for $k_1 \neq 0$, become smaller compared to $h_{1,n}[0]$ and, therefore, the first-order Volterra operator becomes easier to invert. This fact is illustrated by the decreasing filter order L of $G_{1,n}$ and also contributes to the improvement of the performance of the Richardson and Jacobi equalizer. Finally, it can be observed in Figure 3.12 that the SNR at the output of the Volterra system (entitled "No equalization") increases with the maximum order Q. The weighting factor $2^p - 1$ in the condition for convergence in (3.77) restricts kernels of order $p \geq 2$ more severly

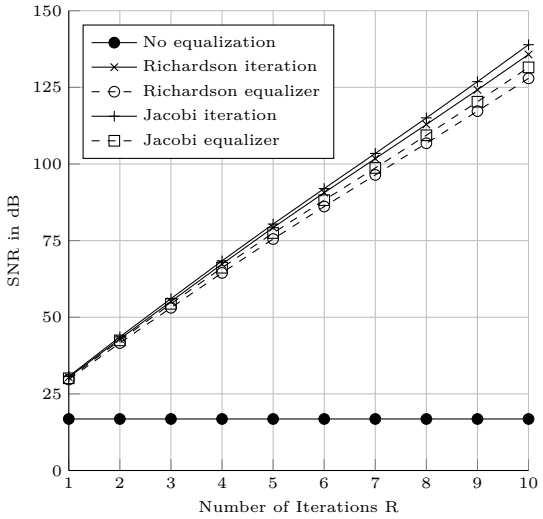


Figure 3.14: Comparison of the equalization performance of the Richardson and Jacobi equalizer to their respective ideal convergence behavior defined by the Richardson and Jacobi iteration for a Volterra system with $K = 4$, $Q = 2$, and $h_{1,n}[0] \sim \mathcal{U}(0.95, 1.05)$. The samples of the input signal are $x[n] \sim \mathcal{U}(-1, 1)$.

and, therefore, allows less overall distortion as the maximum order Q increases.

The influence of the first-order Volterra operator is investigated further in Figure 3.13. Therein, Figure 3.13a illustrates that if the first-order Volterra kernel coincides with the unit impulse sequence, the Richardson equalizer is equivalent to the Nowak-Van-Veen equalizer, which was shown analytically in Section 3.6.1. Considering the results in Figure 3.12 as well, this implies that the Richardson equalizer approaches the equalization performance of the Nowak-Van-Veen equalizer if the first-order Volterra operator approaches the identity function, i.e., $H_{1,n}\{x[n]\} = x[n]$. Furthermore, it is observed that the Jacobi equalizer outperforms the Nowak-Van-Veen equalizer if it is not burdened with the inversion of the first-order Volterra operator. In the contrary situation, where $h_{1,n}[0]$ varies rather severely, the Richardson equalizer turns out to be inferior to the other equalizers, which are immune to this variation as illustrated in Figure 3.13b. This can be traced back to the particular convergence properties of the underlying Richardson iteration, which is highlighted in the following section.

3.7.3 Optimal Convergence of the Richardson and Jacobi Equalizer

As discussed in Section 3.6, the Richardson and Jacobi equalizer rely on an approximation $\mathbf{A}_{\tilde{\mathbf{x}}^{(r)}}$ of the coefficient matrix $\mathbf{A}_{\mathbf{x}}$, which is improved in every iteration step if the condition for convergence is fulfilled. Consequently, if the initial approximation $\tilde{\mathbf{x}}^{(0)} = \mathbf{y}$ is close to the solution $\mathbf{x}$, these equalizers will exhibit approximately the same performance as the Richardson and Jacobi iteration, respectively, which use the exact coefficient matrix and, therefore, exhibit the optimal convergence behavior. The requirement of a decent initial approximation translates to a high SNR at the output of the Volterra system. In order to compare the convergence behavior of the Richardson and Jacobi equalizer to the optimal convergence in a bad case, a situation with a low SNR at the output of the Volterra system is considered. As visible in Figure 3.12, the lowest SNR at the output of the Volterra system is observed for $Q = 2$, which is

about 17 dB.[4] Therefore, the equalization performance of the Richardson and Jacobi equalizer is compared to the Richardson and Jacobi iteration for $Q = 2$ in Figure 3.14. This simulation reveals that the condition for convergence in (3.77) assures a sufficiently high SNR at the output of the Volterra system to enable a nearly optimal convergence behavior of the Richardson and Jacobi equalizer.

3.7.4 Computational Complexity

The preceding discussion was focused on the comparison of the equalization performance of the different methods, whereas the computational complexity was neglected. However, in a practical implementation this is a significant decision criterion. A serious drawback of the Pth-order inverse and the Nowak-Van-Veen equalizer is the requirement of the inverse filter $G_{1,n}$. This implies the need of a computationally costly filter design whenever the first-order Volterra operator $H_{1,n}$ changes. If the Volterra system varies only slowly with time this may be managable, but if the Volterra system varies reasonably fast, the frequent filter design probably becomes infeasible. Regarding the equalization structure itself, the Pth-order inverse exhibits the least computational complexity due to the truncated Volterra systems in the reconstruction stages. However, the price paid is a rather poor performance if its order P is less than the maximum order Q of the Volterra system, and if its performance becomes competitive, i.e., for $P \geq Q$, the computational advantage over the other methods vanishes. The Nowak-Van-Veen, Richardson, and Jacobi equalizer exhibit approximately the same complexity, but the latter requires a division in every reconstruction stage. Altough a division by zero is prevented by the condition for convergence for the Jacobi iteration in (3.49) anyway,[5] the realization of a divider is considerably more complex than the realization of a multiplier.

3.8 Summary

This chapter opened with the introduction of the time-varying Volterra series and an overview of existing equalization methods, where the latter covered the Pth-order inverse and nonlinear fixed-point iteration in more detail. Subsequently, it was shown that a time-varying Volterra series may be transformed to a linear time-varying system, whose impulse response depends on the time index as well as on the input signal. This novel perspective was utilized to derive two equalization methods, the Richardson equalizer and the Jacobi equalizer, where the structure of the former was shown to coincide with the nonlinear Richardson iteration. For the Richardson equalizer, the condition for convergence was investigated and resulted in particularly simple and easily evaluable expression based on the Volterra kernels and the input amplitude range. Finally, the Richardson and Jacobi equalizer were simulated under different conditions and compared to a particular Pth-order inverse and nonlinear fixed-point iteration. It was highlighted that the key advantage of the Richardson and Jacobi equalizer is the absence of the inverse filter for the first-order Volterra operator, which is a fundamental requirement of the existing methods that

[4] Note that the Richardson and Jacobi equalizer is equivalent to the Richardson and Jacobi iteration, respectively, if the maximum order of the Volterra system is $Q = 1$, cf. (3.22) and (3.54).

[5] Note that the condition for convergence for the Jacobi equalizer may only be more restrictive.

limits their performance and adds significant computational complexity if the Volterra system varies reasonably fast with time. Furthermore, it was demonstrated that the Richardson and Jacobi equalizer perform particularly well if the first-order Volterra operator is close to the identity function, and if it coincides with the identity function, the Jacobi equalizer outperforms all other methods.

Chapter 4

Concluding Remarks and Future Research

In this thesis, two particular aspects of reconstruction methods for time-varying systems were investigated, i.e., parallel processing with LTV filters and equalization methods for time-varying Volterra systems. Regarding the former issue, a design equation was introduced, which is a convenient tool to achieve parallel processing with LTV systems. It is readily applicable to convolution-based LTV systems and provides the basis for parallel processing with more complex systems, which was demonstrated for the Farrow filter and iterative correction structures. As this method does not introduce any additional computational effort and even reduces the critical path under certain conditions, it can be regarded as optimal. Concerning the latter issue, a novel view on time-varying Volterra systems was established to derive and analyze equalization methods, which resulted in the Richardson and Jacobi equalizer, where the former coincides with the nonlinear Richardson iteration and the latter is entirely novel. For the Richardson equalizer, the range of applicability, i.e., the condition for convergence, was derived analytically, where the resulting criterion is simple to evaluate for a given time-varying Volterra system. A comparison of these equalizers to existing methods demonstrated that they perform very well under certain conditions and provide a computational advantage if the Volterra system varies reasonably fast with time. Concluding, it is hoped that the results developed in this thesis prove to be a valuable contribution to reconstruction methods for time-varying systems and help to tackle future challenges in engineering.

Future Research

The approach towards Volterra system equalization introduced in Chapter 3 exhibits a serious potential for research and only some of the emerging questions could be addressed in this thesis. Therefore, an overview of future research shall be provided below.

- Due to the absence of the inverse filter, the Richardson and Jacobi equalizer may exhibit a significant advantage over the existing methods if they are utilized in conjunction with identification methods. Therefore, it would be interesting to investigate on identification and the sensitivity of the methods to identification errors.
- As mentioned in Section 3.5.1, the convergence behavior may be improved by using the final reconstruction $\hat{x}[n-k_i] = \tilde{x}[n-k_i]^{(R)}$ of previous samples, i.e., $k_i > 0$, instead of the intermediate reconstruction results $\tilde{x}[n-k_i]^{(r)}$, cf. [6]. Besides improved convergence, this

may as well be beneficial in practical implementations, because some results can be reused in the reconstruction stages, in particular those subsets of the products that involve only previous reconstruction results.

- The condition for convergence, which was derived by considering the worst case, showed to be rather conservative with respect to the average behavior. Consequently, the derivation of a condition for convergence and rate of convergence in the *mean* would provide further insight. Furthermore, the analysis could be complemented by corresponding derivations for the Jacobi equalizer.

- On the basis of simulations, it was discovered that the Richardson and Jacobi equalizer are also suited for predistortion. Consequently, the theoretical investigation on the application for predistortion and the corresponding analysis of the condition for convergence would open up a new field of application.

- Last but not least, potential applications should be investigated. This could include, e.g., the linearization of an analog front end and amplifier predistortion.

Appendix A

Direct and Transposed Form LTV FIR Filters

In the following, the motivation for focusing on direct form LTV FIR filters in Chapter 2 shall be highlighted. Therefore, direct form and transposed form LTV FIR filters are introduced and their polyphase decomposition is discussed.

A.1 Direct Form LTV FIR Filters

An LTV FIR filter of order N implemented in direct form is depicted in Figure A.1 and its output is characterized by

$$y[n] = \sum_{k=0}^{N} h_n[k]x[n-k] \; .$$

According to the notation defined in Chapter 2, its z-transform at time instant n is given by[1]

$$H_n(z) = \sum_{k=-\infty}^{\infty} z^{-k} h_n[k] \; .$$

Following the concept of polyphase decomposition [10], the impulse response is split into M subsequences, i.e.,

$$H_n(z) = \sum_{m=0}^{M-1} \left(\sum_{l=-\infty}^{\infty} z^{-Ml-m} h_n[Ml+m] \right) \; .$$

Instead of using delayed subsequences, the *input* signal may be delayed, which results in

$$H_n(z) = \sum_{m=0}^{M-1} z^{-m} \left(\sum_{l=-\infty}^{\infty} z^{-Ml} h_n[Ml+m] \right) \; .$$

A comparison of this result with (2.3) explains the origin of the polyphase decomposition for direct form LTV FIR filters.

[1] Note that $h_n[k] = 0$ for $k < 0$ and $k > N$.

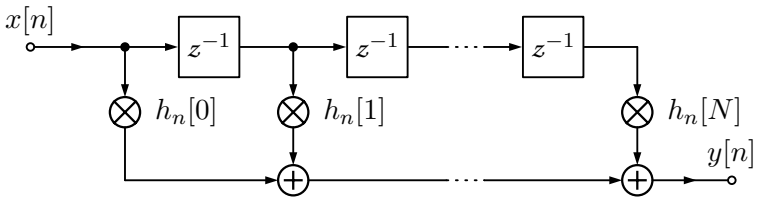


Figure A.1: Time-varying finite impulse response filter of order N in direct form.

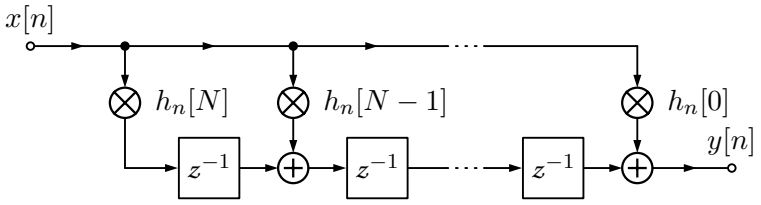


Figure A.2: Time-varying finite impulse response filter of order N in transposed form.

A.2 Transposed Form LTV FIR Filters

An LTV FIR filter of order N implemented in transposed form is depicted in Figure A.2 and its output is characterized by

$$y[n] = \sum_{k=0}^{N} h_{n-k}[k]x[n-k] \; .$$

Again, following the notation defined in Chapter 2, its z-transform at time instant n is given by

$$H_n(z) = \sum_{k=-\infty}^{\infty} h_n[k]z^{-k} \; ,$$

and splitting into M subsequences results in

$$H_n(z) = \sum_{m=0}^{M-1} \left(\sum_{l=-\infty}^{\infty} h_n[Ml+m]z^{-Ml-m} \right) \; .$$

Instead of using delayed subsequences, the *output* signal of the filter may be delayed, i.e.,

$$H_n(z) = \sum_{m=0}^{M-1} \left(\sum_{l=-\infty}^{\infty} h_n[Ml+m]z^{-Ml} \right) z^{-m} \; .$$

If it is required to delay the input signals instead, as it is the case for the derivation in Chapter 2, the delay and the time-varying multipliers need to be interchanged according to the rule described in Figure 2.2, which results in

$$H_n(z) = \sum_{m=0}^{M-1} z^{-m} \left(\sum_{l=-\infty}^{\infty} h_{n-m}[Ml+m]z^{-Ml} \right) \; . \tag{A.1}$$

The reason to choose the direct form LTV FIR filter for the derivation in Chapter 2 was to avoid the additional mathematical complexity due to these time index changes. However, if a

design equation for transposed form LTV FIR filters is required, one can follow the derivation in Chapter 2 using (A.1) as the starting point of the polyphase decomposition.

Bibliography

[1] C. Vogel, M. Hotz, S. Saleem, K. Hausmair, and M. Soudan, "A review on low-complexity structures and algorithms for the correction of mismatch errors in time-interleaved ADCs," *Proc. IEEE Int. Northeast Workshop Circuits and Systems (NEWCAS) Conf.*, Jun. 2012 (invited paper).

[2] M. Soudan and C. Vogel, "Low complexity least-squares filter design for the correction of linear time-varying systems," in *Proc. 20th European Conf. Circuit Theory and Design (ECCTD)*, Aug. 2011, pp. 665–668.

[3] H. Johansson and P. Löwenborg, "A least-squares filter design technique for the compensation of frequency response mismatch errors in time-interleaved A/D converters," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 55, no. 11, pp. 1154–1158, Nov. 2008.

[4] M. Soudan and C. Vogel, "Correction structures for linear weakly time-varying systems," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. PP, no. 99, pp. 1–10, 2012.

[5] K. M. Tsui and S. C. Chan, "Iterative correction of frequency response mismatches in time-interleaved ADCs: A novel framework and case study in ofdm systems," in *Proc. 2010 Int. Conf. Green Circuits and Systems (ICGCS)*, Jun. 2010, pp. 253–258.

[6] M. Soudan, "Low complexity correction structures for time-varying systems," Ph.D. dissertation, Graz University of Technology, Austria, Sep. 2011.

[7] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation.* Wiley, 1999.

[8] S.-M. Phoong and P. P. Vaidyanathan, "Time-varying filters and filter banks: some basic principles," *IEEE Trans. Signal Processing*, vol. 44, no. 12, pp. 2971–2987, Dec. 1996.

[9] R. D. Nowak and B. D. Van Veen, "Volterra filter equalization: a fixed point approach," *IEEE Trans. Signal Processing*, vol. 45, no. 2, pp. 377–388, Feb. 1997.

[10] P. P. Vaidyanathan, *Multirate Systems and Filter Banks.* Prentice Hall, 1993.

[11] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Prentice Hall, 2009.

[12] W. C. Black and D. A. Hodges, "Time interleaved converter arrays," *IEEE J. Solid-State Circuits*, vol. 15, no. 6, pp. 1022–1029, Dec. 1980.

[13] N. Kurosawa, H. Kobayashi, K. Maruyama, H. Sugawara, and K. Kobayashi, "Explicit analysis of channel mismatch effects in time-interleaved ADC systems," *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 3, pp. 261–271, Mar. 2001.

[14] T.-H. Tsai, P. Hurst, and S. Lewis, "Bandwidth mismatch and its correction in time-interleaved analog-to-digital converters," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 53, no. 10, pp. 1133–1137, Oct. 2006.

[15] C. Vogel, "Modeling, identification, and compensation of channel mismatch errors in time-interleaved analog-to-digital converters," Ph.D. dissertation, Graz University of Technology, Austria, Jul. 2005.

[16] S. Tertinek and C. Vogel, "Reconstruction of nonuniformly sampled bandlimited signals using a differentiator–multiplier cascade," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 55, no. 8, pp. 2273–2286, Sep. 2008.

[17] S. Saleem and C. Vogel, "Adaptive compensation of frequency response mismatches in high-resolution time-interleaved ADCs using a low-resolution ADC and a time-varying filter," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2010)*, Jun. 2010, pp. 561–564.

[18] K. M. Tsui and S. C. Chan, "A versatile iterative framework for the reconstruction of bandlimited signals from their nonuniform samples," *J. Signal Processing Systems*, vol. 62, no. 3, pp. 459–468, Mar. 2011.

[19] C. W. Farrow, "A continuously variable digital delay element," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 1988)*, vol. 3, Jun. 1988, pp. 2641–2645.

[20] S. Afsardoost, T. Eriksson, and C. Fager, "Digital predistortion using a vector-switched model," *IEEE Trans. Microwave Theory and Techniques*, vol. 60, no. 4, pp. 1166–1174, Apr. 2012.

[21] J. Liszewski, B. Schubert, W. Keusgen, and A. Kortke, "Low-complexity FPGA implementation of Volterra predistorters for power amplifiers," in *Proc. IEEE Topical Conf. Power Amplifiers for Wireless and Radio Applications (PAWR)*, Jan. 2011, pp. 41–44.

[22] Z. Peng, Z. Qin, and W. Siliang, "A novel adaptive digital predistortion for RF power amplifier linearization based on simplified Volterra series," in *Proc. Int. Symp. Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, Aug. 2007, pp. 327–331.

[23] G. Lazzarin, S. Pupolin, and A. Sarti, "Nonlinearity compensation in digital radio systems," *IEEE Trans. Communications*, vol. 42, no. 234, pp. 988–999, Feb./Mar./Apr. 1994.

[24] A. Stenger, L. Trautmann, and R. Rabenstein, "Nonlinear acoustic echo cancellation with 2nd order adaptive Volterra filters," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 2, Mar. 1999, pp. 877–880.

[25] A. Guerin, G. Faucon, and R. Le Bouquin-Jeannes, "Nonlinear acoustic echo cancellation based on Volterra filters," *IEEE Trans. Speech and Audio Processing*, vol. 11, no. 6, pp. 672–683, Nov. 2003.

[26] J. Fu and W.-P. Zhu, "A simplified structure of second-order Volterra filters for nonlinear acoustic echo cancellation," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2010)*, Jun. 2010, pp. 2366–2369.

[27] H. Furuhashi, Y. Kajikawa, and Y. Nomura, "Realization of nonlinear acoustic echo cancellation by subband parallel cascade Volterra filter," in *Proc. Int. Symp. Intelligent Signal Processing and Communications (ISPACS)*, Dec. 2006, pp. 837–840.

[28] E. Biglieri, S. Barberis, and M. Catena, "Analysis and compensation of nonlinearities in digital transmission systems," *IEEE J. Selected Areas in Communications*, vol. 6, no. 1, pp. 42–51, Jan. 1988.

[29] T. Ogunfunmi and T. Drullinger, "Equalization of non-linear channels using a Volterra-based non-linear adaptive filter," in *Proc. IEEE 54th Int. Midwest Symp. Circuits and Systems (MWSCAS)*, Aug. 2011, pp. 1–4.

[30] D. Kotoulas and G. Maragakis, "XDSL and OFDM systems equalization using Volterra series," in *Proc. 50th FITCE Congress*, Sep. 2011, pp. 1–3.

[31] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*, ser. Wiley Series in Telecommunications and Signal Processing. Wiley, 2000.

[32] W. J. Rugh, *Nonlinear System Theory: The Volterra/Wiener Approach.* Johns Hopkins University Press, 1981.

[33] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems.* Wiley, 1980.

[34] P. Alper, "A consideration of the discrete Volterra series," *IEEE Trans. Automatic Control*, vol. 10, no. 3, pp. 322–327, Jul. 1965.

[35] S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlinear operators with Volterra series," *IEEE Trans. Circuits and Systems*, vol. 32, no. 11, pp. 1150–1161, Nov. 1985.

[36] M. B. Brilliant, "Theory of the analysis of nonlinear systems," Massachusetts Institute of Technology, Research Laboratory of Electronics, Cambridge, MA, Tech. Rep. 345, Mar. 1958.

[37] B. Murmann, C. Vogel, and H. Koeppl, "Digitally enhanced analog circuits: System aspects," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2008)*, May 2008, pp. 560–563.

[38] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Processing Magazine*, vol. 8, no. 3, pp. 10–26, Jul. 1991.

[39] M. Schetzen, "Theory of pth-order inverses of nonlinear systems," *IEEE Trans. Circuits and Systems*, vol. 23, no. 5, pp. 285–291, May 1976.

[40] E. Biglieri, A. Gersho, R. Gitlin, and T. L. Lim, "Adaptive cancellation of nonlinear intersymbol interference for voiceband data transmission," *IEEE J. Selected Areas in Communications*, vol. 2, no. 5, pp. 765–777, Sep. 1984.

[41] A. J. Redfern and G. T. Zhou, "A root method for Volterra system equalization," *IEEE Signal Processing Letters*, vol. 5, no. 11, pp. 285–288, Nov. 1998.

[42] A. Sarti and S. Pupolin, "Recursive techniques for the synthesis of a *p*th-order inverse of a Volterra system," *European Trans. Telecommunications*, vol. 3, no. 4, pp. 315–322, Jul. 1992.

[43] V. S. Kafka, "Rekursive Strukturen auf Volterra-Basis zur aufwandsarmen Darstellung und Entzerrung von nichtlinearen Systemen," Ph.D. dissertation, Universität der Bundeswehr München, Germany, Mar. 2002.

[44] Y.-W. Fang, L.-C. Jiao, X.-D. Zhang, and J. Pan, "On the convergence of Volterra filter equalizers using a *p*th-order inverse approach," *IEEE Trans. Signal Processing*, vol. 49, no. 8, pp. 1734–1744, Aug. 2001.

[45] K. Lashkari, "A novel Volterra-Wiener model for equalization of loudspeaker distortions," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, May 2006.

[46] A. Carini, J. V. Mathews, and G. L. Sicuranza, "Exact and *p*th order equalization and linearization of recursive polynomial systems," in *Proc. Thirty-Second Asilomar Conf. Signals, Systems & Computers*, vol. 1, Nov. 1998, pp. 688–692.

[47] A. Halme, J. Orava, and H. Blomberg, "Polynomial operators in non-linear systems theory," *Int. J. Systems Science*, vol. 2, no. 1, pp. 25–47, 1971.

[48] P. M. Prenter, "On polynomial operators and equations," in *Nonlinear Functional Analysis and Applications*, L. B. Rall, Ed. New York: Academic Press, 1971.

[49] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, ser. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1995. [Online]. Available: http://www.siam.org/books/textbooks/fr16_book.pdf

[50] E. Zeidler, *Nonlinear Functional Analysis and its Applications: Fixed-Point Theorems*. New York: Springer, 1986.

[51] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*. New York: Dover Publications, 1994.

[52] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2003. [Online]. Available: http://www-users.cs.umn.edu/~saad/IterMethBook_2ndEd.pdf

[53] C. L. Byrne, *Applied Iterative Methods*. AK Peters, 2008.

[54] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.22," http://cvxr.com/, May 2012.

[55] ——, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds. Springer-Verlag Limited, 2008, pp. 95–110, http://stanford.edu/~boyd/graph_dcp.html.